IBM SPSS Modeler 18.5 Python Guia de Script e Automação

Nota

Antes de utilizar essas informações e o produto que elas suportam, leia as informações em "Avisos" na página 477.

Informações do produto

Esta edição se aplica à versão 18, release 4, modificação 0 de IBM® SPSS Modeler e a todos os lançamentos e modificações subsequentes até indicado de outra forma em novas edições.

Índice

Capítulo 1. Script e a Linguagem de Script

Visão geral de script

A criação de script no IBM SPSS Modeler é uma ferramenta poderosa para automatizar processos na interface com o usuário. Os scripts podem executar os mesmos tipos de ações que podem ser executadas com o mouse ou teclado e podem ser utilizados para automatizar tarefas que seriam altamente repetitivas ou que demorariam muito tempo para serem executadas manualmente.

É possível utilizar scripts para:

- Impor uma ordem específica às execuções de nó em um fluxo.
- Configurar propriedades para um nó, bem como executar derivações utilizando um subconjunto de CLEM (Linguagem de Controle para Manipulação de Expressão).
- Especificar uma sequência de ações automática que normalmente envolve interação do usuário, por exemplo, é possível construir um modelo e, em seguida, testá-lo.
- Configurar processos complexos que requerem interação com o usuário substancial, por exemplo, procedimentos de validação cruzada que requerem geração e teste de modelo repetitivos.
- Configurar processos que manipulam fluxos, por exemplo, é possível selecionar um fluxo de treinamento do modelo, executá-lo e produzir o fluxo de teste de modelo correspondente automaticamente.

Este capítulo fornece descrições de alto nível e exemplos de scripts de nível de fluxo, scripts independentes e scripts em SuperNodes na interface do IBM SPSS Modeler. Mais informações sobre a linguagem de script, sintaxe e comandos são fornecidas nos capítulos que se seguem.

Nota:

Não é possível importar e executar scripts criados em IBM SPSS Statistics dentro de IBM SPSS Modeler

Tipos de Scripts

O IBM SPSS Modeler utiliza três tipos de scripts:

- Os **Scripts de fluxo** são armazenados como uma propriedade de fluxo e, portanto, são salvos e carregados com um fluxo específico. Por exemplo, é possível gravar um script de fluxo que automatiza o processo de treinamento e aplicação de um nugget do modelo. Também é possível especificar que, sempre que um fluxo específico for executado, o script deverá ser executado ao invés do conteúdo da tela do fluxo.
- Os **Scripts independentes** não estão associados a nenhum fluxo específico e são salvos em arquivos de texto externos. É possível utilizar um script independente, por exemplo, para manipular diversos fluxos juntos.
- Os **scripts de SuperNode** são armazenados como uma propriedade fluxo de SuperNode. Os scripts de SuperNode estão disponíveis apenas nos SuperNodes de terminal. É possível utilizar um script SuperNode para controlar a sequência de execução do conteúdo do SuperNode. Para SuperNodes de não terminal, (origem ou processo), é possível definir propriedades para o SuperNode ou para os nós que ele contiver em seu script de fluxo diretamente.

Scripts de Fluxo

Os scripts podem ser utilizados para customizar operações dentro de um fluxo específico e são salvos com esse fluxo. Os scripts de fluxo podem ser usados para especificar uma ordem de execução específica para os nós terminais em um fluxo. Utilize a caixa de diálogo do script de fluxo para editar o script que é salvo com o fluxo atual.

Para acessar a guia de script de fluxo na caixa de diálogo Propriedades do Fluxo:

1. No menu **Ferramentas** , escolha:

   Propriedades do Fluxo de > **Execução**
2. Clique na guia **Execução** para trabalhar com scripts para o fluxo atual.

Use os ícones da barra de ferramentas na parte superior da caixa de diálogo do script de fluxo para as operações a seguir:

- Importe o conteúdo de um script independente pré-existente para a janela
- Salvar um script como um arquivo de texto.
- Imprimir um script.
- Anexar script padrão.
- Editar um script (desfazer, recortar, copiar, colar e outras funções comuns de edição).
- Executar o script atual inteiro.
- Execute as linhas selecionadas a partir de um script.
- Parar um script durante a execução. (Este ícone é ativado apenas quando um script estiver em execução).
- Verifique a sintaxe do script e, se quaisquer erros forem localizados, exiba-os para revisão no painel inferior da caixa de diálogo.

Nota: A partir da versão 16.0, o SPSS Modeler utiliza a linguagem de script Python. Todas as versões anteriores a 16.0 usavam uma linguagem de script exclusiva para SPSS Modeler, agora referida como script anterior Dependendo do tipo de script com o qual você está trabalhando, na guia **Execução** selecione o modo de execução **Padrão (script opcional)** e, em seguida, selecione **Python** ou **Anterior**.

É possível especificar se um script é ou não executado quando o fluxo é executado Para executar o script cada vez que o fluxo for executado, respeitando a ordem de execução do script, selecione **Executar este script** Essa configuração fornece automação no nível do fluxo para uma construção de modelo mais rápida. No entanto, a configuração padrão é ignorar este script durante a execução de fluxo. Mesmo se você selecionar a opção **Ignorar este script**, o script sempre poderá ser executado diretamente a partir dessa caixa de diálogo.

O editor de script inclui os recursos a seguir que ajudam na criação de script:

- Destaque da sintaxe; palavras-chave, valores literais (como sequências e números) e comentários são destacados.
- Numeração de linha.
- Correspondência do bloco; quando o cursor é colocado no início de um bloco do programa, o bloco final correspondente também é destacado.
- Conclusão automática sugerida.

As cores e estilos de texto que são usados pelo marcador de sintaxe podem ser customizados usando as preferências de exibição IBM SPSS Modeler . Para acessar as preferências de exibição, escolha **Ferramentas** > **Opções** > **Opções do usuário** e selecione a guia **Sintaxe**

Uma lista de preenchimentos de sintaxe sugeridos pode ser acessada selecionando **Sugestão Automática** no menu de contexto ou pressionando Ctrl + Espaço. Utilize as teclas de cursor para mover a lista para cima e para baixo e, em seguida, pressione Enter para inserir o texto selecionado. Para sair do modo de sugestão automática sem modificar o texto existente, pressione Esc.

A guia **Depuração** exibe mensagens de depuração e pode ser usada para avaliar o estado do script quando o script for executado. A guia **Depurar** consiste em uma área de texto somente leitura e um campo de texto de entrada de linha única A área de texto exibe o texto que é enviado para a saída padrão ou para o erro padrão pelos scripts, por exemplo, por meio do texto da mensagem de erro. O campo de texto de entrada aceita entrada do usuário. Esta entrada é então avaliada dentro do contexto do script que foi executado mais recentemente no diálogo (conhecido como *contexto de script*). A área de texto

contém o comando e a saída resultante para que o usuário possa ver um rastreio de comandos. O campo de entrada de texto sempre contém o prompt de comando (- -> para script anterior).

Um novo contexto de script é criado nas circunstâncias a seguir:

- Um script é executado usando **Executar este script** ou **Executar linhas selecionadas**..
- A linguagem de script é alterada.

Se um novo contexto de script for criado, a área de texto será limpa.

Nota: Executar um fluxo fora da área de janela de script não modifica o contexto de script da área de janela de script. Os valores de quaisquer variáveis criadas como parte dessa execução não são visíveis na caixa de diálogo do script.

Exemplo de Script de Fluxo: Treinando uma Rede Neural

Um fluxo pode ser utilizado para treinar um modelo de rede neural quando executado. Normalmente, para testar o modelo, é possível executar o nó de modelagem para incluir o modelo no fluxo, fazer as conexões apropriadas e executar um nó Análise.

Usando um script IBM SPSS Modeler, é possível automatizar o processo de teste do nugget do modelo após criá-lo. Por exemplo, o script de fluxo a seguir para testar o fluxo demo `druglearn.str` (disponível na pasta `/Demos/streams/` sob a instalação do IBM SPSS Modeler) poderia ser executado a partir do diálogo Propriedades do Fluxo (**Ferramentas> Propriedades do Fluxo> Script**):

```
stream = modeler.script.stream()
neuralnetnode = stream.findByType("neuralnetwork", None)
results = []
neuralnetnode.run(results)
appliernode = stream.createModelApplierAt(results[0], "Drug", 594, 187)
analysisnode = stream.createAt("analysis", "Drug", 688, 187)
typenode = stream.findByType("type", None)
stream.linkBetween(appliernode, typenode, analysisnode)
analysisnode.run([])
```

Os marcadores a seguir descrevem cada linha neste exemplo de script.

- A primeira linha define uma variável que aponta para o fluxo atual.
- Na linha 2, o script localiza o nó construtor Rede Neural.
- Na linha 3, o script cria uma lista onde os resultados da execução podem ser armazenados.
- Na linha 4, o nugget do modelo de Rede Neural é criado. Ele é armazenado na lista definida na linha 3.
- Na linha 5, um nó de aplicação de modelo é criado para o nugget do modelo e colocado na tela do fluxo.
- Na linha 6, um nó de análise chamado `Drug` é criado.
- Na linha 7, o script localiza o nó Tipo.
- Na linha 8, o script conecta o nó de aplicação de modelo criado na linha 5 entre o nó Tipo e o nó Análise.
- Finalmente, o nó Análise é executado para produzir o relatório Análise.

É possível utilizar um script para construir e executar um fluxo desde o início, começando com uma tela em branco.

Limites de tamanho do código Jython

Jython compila cada script para bytecode Java, que é então executado pela Java Virtual Machine (JVM). No entanto, Java impõe um limite no tamanho de um único arquivo bytecode. Assim, quando Jython tenta carregar o bytecode, ele pode fazer com que a JVM trave. IBM SPSS Modeler é incapaz de evitar que isso aconteça.

Certifique-se de gravar seus scripts Jython usando boas práticas de codificação (como minimizar o código duplicado usando variáveis ou funções para calcular valores intermediários comuns). Se necessário,

talvez você precise dividir seu código sobre vários arquivos de origem ou defini-lo usando módulos, pois estes são compilados em arquivos bytecode separados.

Scripts Independentes

A caixa de diálogo Script Independente é utilizada para criar ou editar um script que é salvo como um arquivo de texto. Ela exibe o nome do arquivo e fornece recursos para carregar, salvar, importar e executar os scripts.

Para acessar a caixa de diálogo de script independente:

No menu principal, escolha:

Ferramentas > **Script Independente**

As mesmas opções de verificação de barra de ferramentas e de sintaxe de script estão disponíveis tanto para scripts independentes quanto para scripts de fluxo. Consulte o tópico "Scripts de Fluxo" na página 1 para obter informações adicionais.

Exemplo de Script Independente: Salvando e Carregando um Modelo

Os scripts independentes são úteis para manipulação de fluxo. Suponha que você tenha dois fluxos – um que cria um modelo e outro que utiliza gráficos para explorar o conjunto de regras gerado a partir do primeiro fluxo com campos de dados existentes. Um script independente para este cenário pode ser semelhante a este:

```
taskrunner = modeler.script.session().getTaskRunner()

# Modify this to the correct Modeler installation Demos folder.
# Note use of forward slash and trailing slash.
installation = "C:/Program Files/IBM/SPSS/Modeler/19/Demos/"

# First load the model builder stream from file and build a model
druglearn_stream = taskrunner.openStreamFromFile(installation + "streams/druglearn.str", True)
results = []
druglearn_stream.findByType("c50", None).run(results)

# Save the model to file
taskrunner.saveModelToFile(results[0], "rule.gm")

# Now load the plot stream, read the model from file and insert it into the stream
drugplot_stream = taskrunner.openStreamFromFile(installation + "streams/drugplot.str", True)
model = taskrunner.openModelFromFile("rule.gm", True)
modelapplier = drugplot_stream.createModelApplier(model, "Drug")

# Now find the plot node, disconnect it and connect the
# model applier node between the derive node and the plot node
derivenode = drugplot_stream.findByType("derive", None)
plotnode = drugplot_stream.findByType("plot", None)
drugplot_stream.disconnect(plotnode)
modelapplier.setPositionBetween(derivenode, plotnode)
drugplot_stream.linkBetween(modelapplier, derivenode, plotnode)
plotnode.setPropertyValue("color_field", "$C-Drug")
plotnode.run([])
```

Nota: Para aprender mais sobre a linguagem de script em geral, consulte "Visão Geral de Linguagem de Script" na página 15.

Exemplo de Script Independente: Gerando um Modelo de Seleção de Variável

Iniciando com uma tela em branco, esse exemplo constrói um fluxo que gera um modelo de Seleção de Variável, aplica o modelo e cria uma tabela que lista os 15 campos mais importantes com relação ao destino especificado.

```
stream = modeler.script.session().createProcessorStream("featureselection",
True)
```

```
statisticsimportnode = stream.createAt("statisticsimport", "Statistics
File", 150, 97)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/
customer_dbase.sav")

typenode = stream.createAt("type", "Type", 258, 97)
typenode.setKeyedPropertyValue("direction", "response_01", "Target")

featureselectionnode = stream.createAt("featureselection", "Feature
Selection", 366, 97)
featureselectionnode.setPropertyValue("top_n", 15)
featureselectionnode.setPropertyValue("max_missing_values", 80.0)
featureselectionnode.setPropertyValue("selection_mode", "TopN")
featureselectionnode.setPropertyValue("important_label", "Check Me Out!")
featureselectionnode.setPropertyValue("criteria", "Likelihood")

stream.link(statisticsimportnode, typenode)
stream.link(typenode, featureselectionnode)
models = []
featureselectionnode.run(models)

# Assumes the stream automatically places model apply nodes in the stream
applynode = stream.findByType("applyfeatureselection", None)
tablenode = stream.createAt("table", "Table", applynode.getXPosition() + 96,
applynode.getYPosition())
stream.link(applynode, tablenode)
tablenode.run([])
```

O script cria um nó de origem para ler nos dados, utiliza um nó Tipo para configurar a função (direção) para o campo `response_01` para `Target` e, em seguida, cria e executa um Nó de Variável. O script também conecta os nós e posiciona cada um na tela do fluxo para produzir um layout legível. O nugget do modelo resultante é então conectado a um nó Tabela que lista os 15 campos mais importantes, conforme determinado pelas propriedades `selection_mode` e `top_n`. Veja o tópico "propriedades de featureselectionnode" na página 255 para obter mais informações.

Scripts SuperNode

É possível criar e salvar scripts dentro de qualquer terminal SuperNodes usando a linguagem de script IBM SPSS Modeler Esses scripts estão disponíveis apenas para os SuperNodes de terminal e são geralmente utilizados quando criar fluxos de modelo ou para impor uma ordem de execução especial ao conteúdo do SuperNode. Os scripts SuperNode também permitem ter mais de um script em execução dentro de um fluxo.

Por exemplo, suponha que você precise especificar a ordem de execução para um fluxo complexo e seu SuperNode contém vários nós que incluem um nó SetGlobals, que precisa ser executado antes de obter um novo campo utilizado em um nó Gráfico. Neste caso, é possível criar um script SuperNode que executa o nó SetGlobals primeiro. Em seguida, os valores calculados por este nó, como o desvio médio ou padrão, poderão ser utilizados quando o nó Gráfico for executado.

Em um script SuperNode, é possível especificar propriedades do nó da mesma maneira que outros scripts. Como alternativa, é possível alterar e definir as propriedades para qualquer SuperNode ou seus nós encapsulados diretamente a partir de um script de fluxo. Consulte o tópico Capítulo 21, "Propriedades do Supernó", na página 459 para obter informações adicionais. Este método funciona para SuperNodes de origem e de processo, bem como para SuperNodes de terminal.

Nota: Como apenas o terminal SuperNodes pode executar seus próprios scripts, a guia Scripts da caixa de diálogo SuperNode está disponível apenas para o terminal SuperNodes

Para abrir a caixa de diálogo do script SuperNode a partir da tela principal:

Selecione um SuperNode de terminal na tela de fluxo e, a partir do menu SuperNode, escolha:

Script SuperNode...

Para abrir a caixa de diálogo do script SuperNode a partir da tela do SuperNode com zoom aumentado:

Clique com o botão direito na tela SuperNode e, no menu de contexto, escolha:

Script SuperNode...

Exemplo de Script SuperNode

O script SuperNode a seguir declara a ordem na qual os nós terminais devem ser executados dentro do SuperNode. Essa ordem assegura que o nó Configurar Globais seja executado primeiro para que os valores calculados por este nó possam então ser utilizados quando outro nó for executado.

```
execute 'Set Globals'
execute 'gains'
execute 'profit'
execute 'age v. $CC-pep'
execute 'Table'
```

Bloqueando e desbloqueando SuperNodes

O exemplo a seguir ilustra como é possível bloquear e desbloquear um SuperNode:

```
stream = modeler.script.stream()
superNode=stream.findByID('id854RNTSD5MB')
# unlock one super node
print 'unlock the super node with password abcd'
if superNode.unlock('abcd'):
    print 'unlocked.'
else:
    print 'invalid password.'
# lock one super node
print 'lock the super node with password abcd'
superNode.lock('abcd')
```

Executando loop e execução condicional em fluxos

A partir da versão 16.0, o SPSS Modeler permite criar alguns scripts básicos a partir de dentro de um fluxo ao selecionar valores em várias caixas de diálogo ao invés de ter que gravar instruções diretamente na linguagem de script. Os dois tipos principais de scripts que podem ser criados dessa forma são loops simples e uma maneira de executar nós se uma condição tiver sido atendida.

É possível combinar ambas as regras de execução de loop e condicional dentro de um fluxo. Por exemplo, é possível ter dados relativos a vendas de carros de fabricantes do mundo todo. É possível configurar um loop para processar os dados em um fluxo, identificar detalhes pelo país do fabricante e gerar os dados para diferentes gráficos mostrando detalhes como o volume de vendas por modelo, níveis de emissões por fabricante e tamanho do motor, e assim por diante. Se você desejar analisar apenas informações europeias, também é possível incluir condições no loop para impedir que sejam criados gráficos de fabricantes baseados nas Américas e Ásia.

Nota: Como as execuções de loop e condicional baseiam-se em scripts de histórico, elas se aplicarão a um fluxo inteiro somente quando ele for executado.

- **Loop** É possível utilizar loop para automatizar tarefas repetitivas. Por exemplo, isso pode significar incluir um determinado número de nós em um fluxo e alterar um parâmetro de nó todas as vezes. Como alternativa, é possível controlar a execução de um fluxo ou ramificação muitas outras vezes mais, como nos exemplos a seguir:
  - Executar o fluxo em um determinado número de vezes e alterar a origem todas as vezes.
  - Executar o fluxo em um determinado número de vezes, alterando o valor de uma variável todas as vezes.
  - Executar o fluxo em um determinado número de vezes, inserindo um campo extra em cada execução.

  - Construir um modelo em um determinado número de vezes e alterar a configuração do modelo todas as vezes.
- **Execução Condicional** É possível utilizar esta opção para controlar como os nós terminais são executados com base nas condições que você predefinir, como nos exemplos a seguir:
  - Com base em se um determinado valor é true ou false, controlar se um nó será executado.
  - Defina se um loop de nós será executado em paralelo ou sequencialmente.

Ambas as execuções de loop e condicional são configuradas na guia Execução dentro da caixa de diálogo Propriedades do Fluxo. Todos os nós que forem utilizados nos requisitos condicional ou de loop são mostrados com um símbolo adicional anexado a eles na tela do fluxo para indicar que fazem parte das execuções de loop e condicional.

É possível acessar a guia Execução de uma das 3 maneiras:

- Utilizando os menus na parte superior da caixa de diálogo principal:
  1. No menu Ferramentas, escolha:

     Propriedades do Fluxo > **Execução**
  2. Clique na guia Execução para trabalhar com scripts para o fluxo atual.
- De dentro de um fluxo:
  1. Clique com o botão direito em um nó e escolha **Execução de Loop/Condicional**.
  2. Selecione a opção de submenu relevante.
- Na barra de ferramentas gráficas na parte superior da caixa de diálogo principal, clique no ícone de propriedades do fluxo.

Se estiver configurando pela primeira vez os detalhes de execução de loop ou condicional, selecione o modo de execução **Execução de Loop/Condicional** na guia Execução e, em seguida, selecione a subguia **Condicional** ou **Loop**.

Executando loop em fluxos

Com o loop, é possível automatizar tarefas repetitivas em fluxos; exemplos podem incluir o seguinte:

- Executar o fluxo em um determinado número de vezes e alterar a origem todas as vezes.
- Executar o fluxo em um determinado número de vezes, alterando o valor de uma variável todas as vezes.
- Executar o fluxo em um determinado número de vezes, inserindo um campo extra em cada execução.
- Construir um modelo em um determinado número de vezes e alterar a configuração do modelo todas as vezes.

Configure as condições a serem atendidas na subguia **Loop** da guia Execução do fluxo. Para exibir a subguia, selecione o modo de execução **Execução de Loop/Condicional**.

Quaisquer requisitos de execução de loop que você definir entrarão em vigor ao executar o fluxo, se o modo de execução **Execução de Loop/Condicional** foi configurado. Opcionalmente, é possível gerar o código de script para seus requisitos de loop e colá-lo no editor de script clicando em **Colar ...** . no canto inferior direito da subguia Looping; a guia Execução principal exibe mudanças para mostrar o modo de execução **Padrão (script opcional)** com o script na parte superior da guia. Isso significa que é possível definir uma estrutura de execução de loop utilizando as várias opções da caixa de diálogo de loop antes de gerar um script que poderá ser customizado posteriormente no editor de script. Observe que quando você clica em **Colar ...** quaisquer requisitos de execução condicionais definidos também serão exibidos no script gerado.

Importante: As variáveis de loop que você configurar em um fluxo do SPSS Modeler poderão ser substituídas se executar o fluxo em uma tarefa do IBM SPSS Collaboration and Deployment Services. Isso ocorre porque a entrada do editor de tarefas do IBM SPSS Collaboration and Deployment Services substitui a entrada do SPSS Modeler. Por exemplo, se você configurar uma variável de loop no fluxo para criar um nome de arquivo de saída diferente para cada loop, os arquivos serão nomeados corretamente

no SPSS Modeler, porém serão substituídos pela entrada fixa inserida na guia Resultado do IBM SPSS Collaboration and Deployment Services Deployment Manager.

Para configurar um loop:

1. Crie uma chave de iteração para definir a estrutura de loop principal a ser executada em um fluxo. Consulte Crie uma chave de iteração para obter mais informações.
2. Onde necessário, defina uma ou mais variáveis de iteração. Consulte Criar uma variável de iteração para obter mais informações.
3. As iterações e quaisquer variáveis criadas são mostradas no corpo principal da subguia. Por padrão, as iterações são executadas na ordem em que aparecem; para mover uma iteração para cima ou para baixo na lista, clique nela para selecioná-la e, em seguida, utilize a seta para cima ou para baixo na coluna à direita da subguia para alterar a ordem.

Criando uma chave de iteração para loop em fluxos

Utilize uma chave de iteração para definir a estrutura de loop principal a ser realizada em um fluxo. Por exemplo, se estiver analisando vendas de automóveis, será possível criar um parâmetro de fluxo *País de fabricação* e utilizar isso como a chave de iteração; quando o fluxo é executado, essa chave é configurada para cada valor de país diferente em seus dados durante cada iteração. Utilize a caixa de diálogo Definir Chave de Iteração para configurar a chave.

Para abrir a caixa de diálogo, selecione a **Chave de Iteração ...** no canto inferior esquerdo da subguia Looping ou clique com o botão direito do mouse em qualquer nó no fluxo e selecione **Looping / Execução Condicional** > **Define Iteration Key (Fields)** ou **Looping / Execução Condicional** > **Define Iteration Key (Values)**. Se você abrir a caixa de diálogo a partir do fluxo, alguns dos campos poderão ser preenchidos automaticamente, como o nome do nó.

Para configurar uma chave de iteração, preencha os campos a seguir:

Iterar em. É possível selecionar uma das opções a seguir:

- **Parâmetro do Fluxo - Campos**. Utilize esta opção para criar um loop que configura o valor de um parâmetro de fluxo existente para cada campo especificado por vez.
- **Parâmetro de Fluxo – Valores**. Utilize esta opção para criar um loop que configura o valor de um parâmetro de fluxo existente para cada valor especificado por vez.
- **Propriedade do Nó – Campos**. Utilize esta opção para criar um loop que configura o valor de uma propriedade do nó para cada campo especificado por vez.
- **Propriedade do Nó – Valores**. Utilize esta opção para criar um loop que configura o valor de uma propriedade do nó para cada valor especificado por vez.

O Que Configurar. Escolha o item que terá seu valor configurado toda vez que o loop for executado. É possível selecionar uma das opções a seguir:

- **Parâmetro**. Disponível apenas se selecionar **Parâmetro de Fluxo – Campos** ou **Parâmetro de Fluxo – Valores**. Selecione o parâmetro necessário na lista disponível.
- **Nó**. Disponível apenas se selecionar **Propriedade do Nó – Campos** ou **Propriedade do Nó – Valores**. Selecione o nó para o qual deseja configurar um loop. Clique no botão Navegar para abrir o diálogo Selecionar Nó e escolha o nó desejado; se houver muitos nós listados, também será possível filtrar a exibição para mostrar apenas determinados tipos de nós selecionando uma das seguintes categorias: nós de Origem, Processo, Gráfico, Modelagem, Saída, Exportar ou Aplicar Modelo de nós.
- **Propriedade**. Disponível apenas se selecionar **Propriedade do Nó – Campos** ou **Propriedade do Nó – Valores**. Selecione a propriedade do nó na lista disponível.

Campos para Usar. Disponível apenas se selecionar **Parâmetro de Fluxo – Campos** ou **Parâmetro de Fluxo – Valores**. Escolha o campo, ou campos, dentro de um nó a serem utilizados para fornecer os valores de iteração. É possível selecionar uma das opções a seguir:

- **Nó**. Disponível apenas se selecionar **Parâmetro de Fluxo – Campos**. Selecione o nó que contém os detalhes para os quais deseja configurar um loop. Clique no botão Navegar para abrir o diálogo Selecionar Nó e escolha o nó desejado; se houver muitos nós listados, também será possível filtrar a exibição para mostrar apenas determinados tipos de nós selecionando uma das seguintes categorias: nós de Origem, Processo, Gráfico, Modelagem, Saída, Exportar ou Aplicar Modelo de nós.
- **Lista de Campo**. Clique no botão de lista na coluna direita para exibir a caixa de diálogo Selecionar Campos, na qual você seleciona os campos no nó para fornecer os dados de iteração. Consulte o "Selecionando campos para iterações" na página 10 para obter informações adicionais.

Valores para Usar. Disponível apenas se selecionar **Parâmetro de Fluxo – Valores** ou **Propriedade do Nó – Valores**. Escolha o valor, ou valores, dentro do campo selecionado a serem utilizados como valores de iteração. É possível selecionar uma das opções a seguir:

- **Nó**. Disponível apenas se selecionar **Parâmetro de Fluxo – Valores**. Selecione o nó que contém os detalhes para os quais deseja configurar um loop. Clique no botão Navegar para abrir o diálogo Selecionar Nó e escolha o nó desejado; se houver muitos nós listados, também será possível filtrar a exibição para mostrar apenas determinados tipos de nós selecionando uma das seguintes categorias: nós de Origem, Processo, Gráfico, Modelagem, Saída, Exportar ou Aplicar Modelo de nós.
- **Lista de Campo**. Selecione o campo no nó para fornecer os dados de iteração.
- **Lista de Valores**. Clique no botão de lista na coluna direita para exibir a caixa de diálogo Selecionar Valores, na qual você seleciona os valores no campo para fornecer os dados de iteração.

Criando uma variável de iteração para loop em fluxos

É possível utilizar variáveis de iteração para alterar os valores de parâmetros ou de propriedades de fluxos de nós selecionados em um fluxo sempre que um loop for executado. Por exemplo, se seu loop de fluxo estiver analisando os dados de vendas de automóveis e utilizando *País de fabricação* como a chave de iteração, você poderá ter uma saída de gráfico mostrando as vendas por modelo e outra saída de gráfico mostrando informações sobre emissões de gases de escape. Nesses casos, é possível criar variáveis de iteração que criar novos títulos para os gráficos resultantes, como *Emissões de veículos suecos* e *Vendas de automóveis japoneses por modelo*. Utilize a caixa de diálogo Definir Variável de Iteração para configurar todas as variáveis que precisar.

Para abrir a caixa de diálogo, selecione **Incluir Variável ...** no canto inferior esquerdo da subguia Looping ou clique com o botão direito em qualquer nó no fluxo e selecione:**Looping / Execução Condicional** > **Definir Variável de Iteração**.

Para configurar uma variável de iteração, preencha os campos a seguir:

Alterar. Selecione o tipo de atributo que deseja corrigir. É possível escolher a partir de **Parâmetro de Fluxo** ou **Propriedade do Nó**.

- Se você selecionar **Parâmetro de Fluxo**, escolha o parâmetro necessário e, em seguida, utilizando uma das opções a seguir, se disponível em seu fluxo, defina o valor para o qual esse parâmetro deverá ser configurado com cada iteração do loop:
  - **Variável global**. Selecione a variável global para a qual o parâmetro de fluxo deverá ser configurado.
  - **Célula de saída de tabela**. Para configurar um parâmetro de fluxo para ser o valor em uma célula de saída de tabela, selecione a tabela na lista e insira a **Linha** e a **Coluna** a serem utilizadas.
  - **Inserir manualmente**. Selecione esta opção se desejar inserir manualmente um valor para esse parâmetro a ser usado em cada iteração. Ao retornar para a subguia Loop, uma nova coluna é criada na qual você insere o texto necessário.
- Se você selecionar **Propriedade do Nó**, escolha o nó necessário e uma de suas propriedades e, em seguida, configure o valor que deseja utilizar para essa propriedade. Configure o novo valor da propriedade usando uma das seguintes opções:
  - **Independentes**. O valor da propriedade usará o valor da chave de iteração. Consulte o "Criando uma chave de iteração para loop em fluxos" na página 8 para obter informações adicionais.

– **Como prefixo para raiz**. Utiliza o valor da chave de iteração como um prefixo para o que você inserir no campo **Raiz**.
– **Como sufixo para raiz**. Utiliza o valor da chave de iteração como um sufixo para o que você inserir no campo **Raiz**.

Se você selecionar qualquer uma das opções de prefixo ou de sufixo, será solicitado a incluir o texto adicional no campo **Raiz**. Por exemplo, se seu valor da chave de iteração for *País de fabricação* e você selecionar **Como prefixo para raiz**, será possível inserir – *vendas por modelo* neste campo.

Selecionando campos para iterações

Ao criar iterações, é possível selecionar um ou mais campos utilizando a caixa de diálogo Selecionar Campos.

Classificar por É possível classificar campos disponíveis para visualização selecionando uma das opções a seguir:

- **Natural** Visualize a ordem dos campos conforme eles foram transmitidos para o fluxo de dados no nó atual.
- **Nome** Utilize a ordem alfabética para classificar campos para visualização.
- **Tipo** Visualize campos classificados pelo seu nível de medição. Essa opção é útil quando selecionar campos com um nível de medição específico.

Selecione os campos da lista um por vez ou utilize os métodos Shift-clique e Ctrl-clique para selecionar vários campos. Também é possível utilizar os botões abaixo da lista para selecionar grupos de campos com base em seu nível de medição, ou para selecionar ou cancelar seleção de todos os campos na tabela.

Observe que os campos disponíveis para seleção são filtrados para mostrar apenas os campos que forem apropriados para o parâmetro de fluxo ou propriedade do nó que estiver utilizando. Por exemplo, se estiver utilizando um parâmetro de fluxo que tenha um tipo de armazenamento String, apenas os campos que possuírem um tipo de armazenamento String serão mostrados.

Execução condicional em fluxos

A execução condicional permite controlar como os nós terminais são executados com base no conteúdo do fluxo correspondente às condições que você definir; exemplos podem incluir o seguinte:

- Com base em se um determinado valor é true ou false, controlar se um nó será executado.
- Defina se um loop de nós será executado em paralelo ou sequencialmente.

Configure as condições a serem atendidas na subguia **Condicional** da guia Execução do fluxo. Para exibir a subguia, selecione o modo de execução **Execução de Loop/Condicional**.

Quaisquer requisitos de execução condicional que você definir entrarão em vigor ao executar o fluxo, se o modo de execução **Execução de Loop/Condicional** foi configurado. Opcionalmente, é possível gerar o código de script para seus requisitos de execução condicionais e colá-lo no editor de script clicando em **Colar ...** no canto inferior direito da subguia Condicional; a guia Execução principal exibe mudanças para mostrar o modo de execução **Padrão (script opcional)** com o script na parte superior da guia. Isso significa que é possível definir condições utilizando as várias opções da caixa de diálogo de loop antes de gerar um script que poderá ser customizado posteriormente no editor de script. Observe que quando você clica em **Colar ...** quaisquer requisitos de loop definidos também serão exibidos no script gerado.

Para configurar uma condição:

1. Na coluna à direita da subguia Condicional, clique no botão Incluir Nova Condição  para abrir a caixa de diálogo **Incluir Instrução de Execução Condicional**. Neste diálogo você especifica a condição que deve ser atendida para que o nó seja executado.
2. Na caixa de diálogo **Incluir Instrução de Execução Condicional**, especifique o seguinte:
   a. **Nó**. Selecione o nó para o qual deseja configurar a execução condicional. Clique no botão Navegar para abrir o diálogo Selecionar Nó e escolha o nó desejado; se houver muitos nós listados, será

possível filtrar a exibição para mostrar nós por uma das seguintes categorias: nó Exportar, Gráfico, Modelagem ou Saída.

b. **Condição baseada em**. Especifique a condição que deve ser atendida para o nó a ser executado. É possível escolher a partir de uma de quatro opções: **Parâmetro de fluxo**, **Variável global**, **Célula de saída de tabela** ou **Sempre true**. Os detalhes que forem inseridos na metade inferior da caixa de diálogo são controlados pela condição que você escolher.

- **Parâmetro de fluxo**. Selecione o parâmetro na lista disponível e, em seguida, escolha o **Operador** para esse parâmetro; por exemplo, o operador pode ser More than, Equals, Less than, Between, e assim por diante. Em seguida, insira o **Valor**, ou valores mínimo e máximo, dependendo do operador.
- **Variável global**. Selecione a variável na lista disponível; por exemplo, isso pode incluir: Média, Soma, valor Mínimo, valor Máximo ou Desvio padrão. Em seguida, selecione o **Operador** e os valores necessários.
- **Célula de saída de tabela**. Selecione o nó de tabela da lista disponível e, em seguida, escolha a **Linha** e a **Coluna** na tabela. Em seguida, selecione o **Operador** e os valores necessários.
- **Sempre true**. Selecione esta opção se o nó tiver que sempre ser executado. Se você selecionar essa opção, não haverá parâmetros adicionais para selecionar.

3. Repita as etapas 1 e 2, sempre que necessário, até configurar todas as condições que precisar. O nó que você selecionou e a condição a ser atendida antes que o nó seja executado são mostrados no corpo principal da subguia nas colunas **Executar Nó** e **Se esta condição for true**, respectivamente.
4. Por padrão, os nós e as condições são executados na ordem em que eles aparecem; para mover um nó e uma condição para cima ou para baixo na lista, clique nele(a) para selecioná-lo(a) e, em seguida, utilize a seta para cima ou para baixo na coluna à direita da subguia para alterar a ordem.

Além disso, é possível configurar as seguintes opções no final da subguia Condicional:

- **Avaliar tudo na ordem**. Selecione esta opção para avaliar cada condição na ordem em que elas são mostradas na subguia. Todos os nós para os quais condições foram localizadas para ser "True" serão executados quando todas as condições tiverem sido avaliadas.
- **Executar um por vez**. Disponível apenas se **Avaliar tudo na ordem** for selecionado. Selecionar isso significa que se uma condição for avaliada como "True", o nó associado a essa condição será executado antes da próxima condição ser avaliada.
- **Avaliar até a primeira ocorrência**. Selecionar isto significa que apenas o primeiro nó que retornar uma avaliação "True" das condições especificadas será executado.

Executando e Interrompendo Scripts

Diversas formas de executar scripts estão disponíveis. Por exemplo, no diálogo de script de fluxo ou de script independente, o botão "Executar este script" executa o script completo:

Figura 1. Botão Executar Este Script

O botão "Executar as linhas selecionadas" executa uma única linha, ou um bloco de linhas adjacentes, que você selecionou no script:

Figura 2. Botão Executar Linhas Selecionadas

É possível executar um script utilizando qualquer um dos métodos a seguir:

- Clique no botão "Executar este script" ou "Executar linhas selecionadas" em uma caixa de diálogo de script de fluxo ou de script independente.

- Execute um fluxo em que **Executar este script** esteja configurado como o método de execução padrão.
- Use o sinalizador `-execute` na inicialização no modo interativo. Veja o tópico "Utilizando Argumentos de Linha de Comandos" na página 65 para obter mais informações.

Nota: Um script SuperNode é executado quando o SuperNode é executado enquanto você selecionou **Executar este script** na caixa de diálogo do script SuperNode .

Interrompendo Execução do Script

Na caixa de diálogo do script de fluxo, o botão vermelho de parada é ativado durante a execução do script. Utilizando este botão, é possível abandonar a execução do script e de qualquer fluxo atual.

Localizar e substituir

A caixa de diálogo Localizar/Substituir está disponível em locais onde você edita script ou texto de expressão, incluindo o editor de script, construtor de expressões do CLEM ou quando você define um modelo no nó de Relatório. Durante a edição de texto em qualquer uma dessas áreas, pressione `Ctrl`+F para acessar a caixa de diálogo, assegurando que o foco do cursor esteja na área de texto. Se estiver trabalhando em um nó de Preenchimento, por exemplo, é possível acessar a caixa de diálogo a partir de qualquer uma das áreas de texto na guia Configurações ou do campo de texto no construtor de expressões.

1. Com o cursor em uma área de texto, pressione Ctrl+F para acessar a caixa de diálogo Localizar/Substituir.
2. Insira o texto que deseja procurar ou escolha na lista suspensa de itens procurados recentemente.
3. Insira o texto de substituição, se houver.
4. Clique em **Localizar Próximo** para iniciar a procura.
5. Clique em **Substituir** para substituir a seleção atual ou em **Substituir Todos** para atualizar todas as instâncias ou as selecionadas.
6. A caixa de diálogo fecha após cada operação. Pressione F3 em qualquer área de texto para repetir a última operação Localizar ou pressione Ctrl+F para acessar a caixa de diálogo novamente.

Opções de Procura

Match case. Especifica se a operação Localizar faz distinção entre maiúsculas e minúsculas; por exemplo, se *myvar* corresponde a *myVar*. O texto de substituição é sempre inserido exatamente como digitado, independentemente dessa configuração.

Somente palavras inteiras. Especifica se a operação Localizar corresponde ao texto integrado às palavras. Se for selecionada, por exemplo, uma procura por *spider* não corresponderá a *spiderman* ou *spider-man*.

Expressões regulares. Especifica se a sintaxe de expressão regular é usada (consulte a próxima seção). Quando selecionada, a opção **Somente palavras inteiras** é desativada e seu valor é ignorado.

Somente texto selecionado. Controla o escopo da procura durante o uso da opção **Substituir Todos**.

sintaxe de expressão regular

Expressões regulares permitem procurar caracteres especiais, como guias ou caracteres de nova linha, classes ou intervalos de caracteres, como de *a* a *d*, qualquer dígito ou não dígito e limites, como início ou fim de uma linha. Os tipos de expressões a seguir são suportados.

Tabela 1. Correspondências de caractere

Caracteres	Correspondências
x	O caractere x
\\	O caractere barra invertida

Tabela 1. Correspondências de caractere (continuação)	
Caracteres	**Correspondências**
\0n	O caractere com valor octal 0n (0 <= n <= 7)
\0nn	O caractere com valor octal 0nn (0 <= n <= 7)
\0mnn	O caractere com valor octal 0mnn (0 <= m <= 3, 0 <= n <= 7)
\xhh	O caractere com valor hexadecimal 0xhh
\uhhhh	O caractere com valor hexadecimal 0xhhhh
\t	O caractere de tabulação ('\u0009')
\n	O caractere de nova linha (feed de linha) ('\u000A')
\r	O caractere de retorno de linha ('\u000D')
\f	O caractere de alimentação de formulário ('\u000C')
\a	O caractere de alerta (sino) ('\u0007')
\e	O caractere de escape ('\u001B')
\cx	O caractere de controle correspondente a x

Tabela 2. Classes de caracteres correspondentes	
Classes de caracteres	**Correspondências**
[abc]	a, b ou c (classe simples)
[^abc]	Qualquer caractere, exceto a, b ou c (subtração)
[a-zA-Z]	a a z ou A a Z, inclusivo (intervalo)
[a-d[m-p]]	a a d ou m a p (união). Alternativamente, poderia ser especificado como [a-dm-p]
[a-z&&[def]]	a a z e d, e ou f (intersecção)
[a-z&&[^bc]]	a a z, exceto para b e c (subtração). Alternativamente, poderia ser especificado como [ad-z]
[a-z&&[^m-p]]	a a z, e não m a p (subtração). Alternativamente, poderia ser especificado como [a-lq-z]

Tabela 3. Classes de caractere predefinidas	
Classes de caractere predefinidas	**Correspondências**
.	Qualquer caractere (pode ou não corresponder aos terminadores de linha)
\d	Qualquer dígito: [0-9]
\D	Um não dígito: [^0-9]
\s	Um caractere de espaço em branco: [\t\n\x0B\f\r]
\S	Um caractere de espaço não em branco: [^\s]
\w	Um caractere de palavra: [a-zA-Z_0-9]
\W	Um caractere não de palavra: [^\w]

Tabela 4. Correspondências de limite	
Correspondentes de limite	**Correspondências**
^	O início de uma linha
$	O final de uma linha
\b	Um limite de palavra
\B	Um limite que não seja de palavra
\A	O início da entrada
\Z	O fim da entrada, mas para o terminador final, se houver
\z	O fim da entrada

Capítulo 2. A Linguagem de Script

Visão Geral de Linguagem de Script

O recurso de script para o IBM SPSS Modeler permite criar scripts que operam na interface com o usuário do SPSS Modeler, manipular objetos de saída e executar sintaxe de comando. É possível executar scripts diretamente de dentro do SPSS Modeler.

Os scripts no IBM SPSS Modeler são gravados na linguagem de script Python. A implementação baseada em Java de Python que é utilizada pelo IBM SPSS Modeler é chamada Jython. A linguagem de script consiste nos recursos a seguir:

- Um formato para fazer referência a nós, fluxos, projetos, saída e outros objetos do IBM SPSS Modeler.
- Um conjunto de instruções de scripts ou comandos que podem ser utilizados para manipular esses objetos.
- Uma linguagem de expressão de script para configurar os valores de variáveis, parâmetros e outros objetos.
- Suporte para comentários, continuações e blocos de texto literal.

As seções a seguir descrevem a linguagem de script Python, a implementação Jython de Python e a sintaxe básica para iniciar com o script dentro do IBM SPSS Modeler. Informações sobre propriedades e comandos específicos são fornecidas nas seções seguintes.

Python e Jython

O Jython é uma implementação da linguagem de script Python que é escrita na linguagem Java e integrada com a plataforma Java. O Python é uma linguagem de script poderosa orientada a objetos. O Jython é útil porque fornece os recursos de produtividade de uma linguagem de script madura e, ao contrário de Python, é executado em qualquer ambiente que suportar uma Java virtual machine (JVM). Isso significa que as bibliotecas Java na JVM estão disponíveis para uso quando você estiver gravando programas. Com o Jython, é possível aproveitar esta diferença e utilizar a sintaxe e a maioria dos recursos da linguagem Python.

Como uma linguagem de script, o Python (e sua implementação Jython) é fácil de aprender, eficiente de codificar e tem uma estrutura mínima necessária para criar um programa em execução. Um código pode ser inserido interativamente, ou seja, uma linha por vez. O Python é uma linguagem de script interpretada e não há etapa de pré-compilação como há em Java. Os programas Python são simplesmente arquivos de texto que são interpretados conforme são inseridos (após a análise de erros de sintaxe). Expressões simples, como valores definidos, bem como as ações mais complexas, como definições de função, são imediatamente executadas e disponibilizadas para uso. Todas as mudanças que forem feitas no código podem ser testadas rapidamente. No entanto, a interpretação de script tem algumas desvantagens. Por exemplo, como o uso de uma variável não definida não é um erro do compilador, ela será detectada apenas se (e quando) a instrução na qual a variável é utilizada for executada. Neste caso, o programa pode ser editado e executado para depurar o erro.

O Python vê tudo, incluindo todos os dados e o código, como um objeto. Portanto, é possível manipular esses objetos com as linhas de código. Alguns tipos de seleção, como números e sequências, são mais convenientemente considerados valores e não objetos, e isso é suportado pelo Python. Há um valor nulo que é suportado. Esse valor nulo tem o nome reservado `None`.

Para obter uma introdução mais detalhada para script Python e Jython e também obter alguns scripts de exemplo, consulte http://www.ibm.com/developerworks/java/tutorials/j-jython1/j-jython1.html e http://www.ibm.com/developerworks/java/tutorials/j-jython2/j-jython2.html.

Script Python

Este guia para a linguagem de script Python é uma introdução aos componentes que mais provavelmente serão utilizados ao executar scripts no IBM SPSS Modeler, incluindo conceitos e princípios básicos de programação. Isso fornecerá um conhecimento suficiente para começar a desenvolver seus próprios scripts Python para uso no IBM SPSS Modeler.

Operações

A designação é feita usando um sinal de igual (=). Por exemplo, para designar o valor "3" para uma variável chamada "x", você usaria a seguinte instrução:

```
x = 3
```

O sinal de igual é utilizado também para designar dados de tipo de sequência para uma variável. Por exemplo, para designar o valor "a string value" para a variável "y", você utiliza a seguinte instrução:

```
y = "a string value"
```

A tabela a seguir lista algumas comparações e operações numéricas normalmente utilizadas e suas descrições.

Tabela 5. Comparação e operações numéricas comuns

Operação	**Descrição**
`x < y`	x é menor que y?
`x > y`	x é maior que y?
`x <= y`	x é menor ou igual a y?
`x >= y`	x é maior ou igual a y?
`x == y`	x é igual a y?
`x != y`	x não é igual a y?
`x <> y`	x não é igual a y?
`x + y`	Incluir y em x
`x - y`	Subtrair y de x
`x * y`	Multiplicar x por y
`x / y`	Dividir x por y
`x ** y`	Elevar x à potência y

Listas

Listas são sequências de elementos. Uma lista pode conter qualquer número de elementos e os elementos da lista podem ser qualquer tipo de objeto. As listas também podem ser consideradas como matrizes. O número de elementos na lista pode aumentar ou diminuir conforme os elementos são incluídos, removidos ou substituídos.

Exemplos

`[]`	Qualquer lista vazia.
`[1]`	Uma lista com um único elemento, um número inteiro.

`["Mike", 10, "Don", 20]`	Uma lista com quatro elementos, dois elementos de sequência e dois elementos de número inteiro.
`[[],[7],[8,9]]`	Uma lista de listas. Cada sublista é uma lista vazia ou uma lista de elementos de número inteiro.
`x = 7; y = 2; z = 3;` `[1, x, y, x + y]`	Uma lista de números inteiros. Este exemplo demonstra o uso de variáveis e expressões.

É possível designar uma lista para uma variável, por exemplo:

```
mylist1 = ["one", "two", "three"]
```

Em seguida, é possível acessar elementos específicos da lista, por exemplo:

```
mylist[0]
```

Isso resulta na saída a seguir:

```
one
```

O número entre os suportes (`[]`) é conhecido como um *índice* e refere-se a um determinado elemento da lista. Os elementos de uma lista são indexados iniciando a partir de 0.

Também é possível selecionar um intervalo de elementos de uma lista; isso é denominado *fatiamento*. Por exemplo, `x[1:3]` seleciona o segundo e o terceiro elementos de `x`. O índice final é um após a seleção.

Sequências

Uma *sequência* é uma sequência imutável que é manipulada como um valor. As sequências suportam todas as funções e operadores de sequência imutáveis que resultam em uma nova sequência. Por exemplo, `"abcdef"[1:4]` resulta na saída `"bcd"`.

No Python, os caracteres são representados por sequências de comprimento um.

As sequências literais são definidas utilizando aspas simples ou triplas. As sequências que são definidas utilizando aspas simples não podem abranger outras linhas, enquanto que as sequências que são definidas utilizando aspas triplas podem. Uma sequência pode ser colocada entre aspas simples (`'`) ou aspas duplas (`"`). Um caractere de aspas pode conter outro caractere de aspas sem escape ou o caractere de aspas com escape, que é precedido pelo caractere de barra invertida (`\`).

Exemplos

```
"This is a string"
'This is also a string'
"It's a string"
'This book is called "Python Scripting and Automation Guide".'
"This is an escape quote (\") in a quoted string"
```

Diversas sequências separadas por espaço em branco são automaticamente concatenadas pelo analisador Python. Isso facilita inserir sequências longas e combinar tipos de aspas em uma única sequência, por exemplo:

```
"This string uses ' and " 'that string uses ".'
```

Isso resulta na saída a seguir:

```
This string uses ' and that string uses ".
```

As sequências suportam vários métodos úteis. Alguns desses métodos são fornecidos na tabela a seguir.

Tabela 6. Métodos de sequência

Método	**Uso**
`s.capitalize()`	Capitalizar inicial s
`s.count(ss {,start {,end}})`	Contar as ocorrências de ss em `s[start:end]`
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Testar para ver se s começa com `str` Testar para ver se s termina com `str`
`s.expandtabs({size})`	Substituir guias por espaços; o padrão `size` é 8
`s.find(str {, start {, end}})` `s.rfind(str {, start {, end}})`	Localiza o primeiro índice de `str` em s; se não for localizado, o resultado será -1. `rfind` procura da direita à esquerda.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Localiza primeiro o índice de `str` em s; se não localizado, aumento de `ValueError`. `rindex` procura da direita à esquerda.
`s.isalnum`	Testa se a sequência é alfanumérica
`s.isalpha`	Testa se a sequência é alfabética
`s.isnum`	Testa se a sequência é numérica
`s.isupper`	Testa se a sequência está toda em letras maiúsculas
`s.islower`	Testa se a sequência está toda em letras minúsculas
`s.isspace`	Testa se a sequência está toda em espaço em branco
`s.istitle`	Testa se a cadeia é uma sequência de cadeias alfanuméricas com iniciais maiúsculas

`s.lower()`
`s.upper()`
`s.swapcase()`
`s.title()`
	Converte tudo em letras minúsculas
	Converte tudo em letras maiúsculas
	Converte tudo em letras maiúsculas e minúsculas opostas
	Converte todas as maiúsculas e minúsculas do título

`s.join(seq)`	Une as sequências em `seq` com s como o separador
`s.splitlines({keep})`	Divida s em linhas; se manter, for `true`, mantenha as novas linhas
`s.split({sep {, max}})`	Divida s em "palavras" usando `sep` (o padrão `sep` é um espaço em branco) para até `max` vezes

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Justificação à esquerda da sequência em um campo de `width` de largura
	Justificação à direita da sequência em um campo de `width` de largura
	Justificação ao centro de uma sequência em um campo de `width` de largura
	Preenche com 0.

Tabela 6. Métodos de sequência (continuação)	
Método	**Uso**

`s.lstrip()`
`s.rstrip()`
`s.strip()`
	Remove espaço em branco à direita
	Remove espaço em branco à esquerda
	Remove espaço em branco à direita e à esquerda

`s.translate(str {,delc})`	Traduzir `s` usando tabela, depois de remover quaisquer caracteres em `delc`. `str` deve ser uma sequência com comprimento == 256.
`s.replace(old, new {, max})`	Substitui todas ou `max` ocorrências de sequência `old` pela sequência `new`

Comentários

Comentários são comentários introduzidos pelo sinal de libra (ou hash) (#). Todo texto que segue o sinal de sustenido na mesma linha é considerado parte do comentário e é ignorado. Um comentário pode iniciar em qualquer coluna. O exemplo a seguir demonstra o uso de comentários:

```
#The HelloWorld application is one of the most simple
print 'Hello World' # print the Hello World line
```

Sintaxe da Instrução

A sintaxe da instrução para Python é muito simples. Em geral, cada linha de origem é uma instrução única. Exceto as instruções `expression` e `assignment`, cada instrução é introduzida por um nome de palavra-chave, como `if` ou `for`. Linhas em branco ou linhas de comentário podem ser inseridas em qualquer lugar entre quaisquer instruções no código. Se houver mais de uma instrução em uma linha, cada instrução deverá ser separada por um ponto e vírgula (`;`).

Instruções muito longas podem continuar em mais de uma linha. Neste caso, a instrução que precisar continuar na próxima linha deverá terminar com uma barra invertida (\), por exemplo:

```
x = "A looooooooooooooooooong string" + \
     "another looooooooooooooooooong string"
```

Quando uma estrutura é colocada entre parênteses (`()`), colchetes (`[]`) ou chaves (`{}`), a instrução poderá continuar em uma nova linha após qualquer vírgula, sem a necessidade de inserir uma barra invertida, por exemplo:

```
x = (1, 2, 3, "hello",
       "goodbye", 4, 5, 6)
```

Identificadores

Identificadores são utilizados para nomear variáveis, funções, classes e palavras-chave. Os identificadores podem ter qualquer comprimento, mas devem começar com um caractere alfabético de maiúscula ou minúscula ou o caractere sublinhado (_). Os nomes que começam com um sublinhado geralmente são reservados para nomes internos ou privados. Após o primeiro caractere, o identificador pode conter qualquer número e combinação de caracteres alfabéticos, números de 0 a 9 e o caractere de sublinhado.

Há algumas palavras reservadas no Jython que não podem ser utilizadas para nomear variáveis, funções ou classes. Elas se enquadram nas categorias a seguir:

- **Introdutores de instrução:** `assert`, `break`, `class`, `continue`, `def`, `del`, `elif`, `else`, `except`, `exec`, `finally`, `for`, `from`, `global`, `if`, `import`, `pass`, `print`, `raise`, `return`, `try` e `while`
- **Introdutores de parâmetros:** `as`, `import` e `in`

- **Operadores:** and, in, is, lambda, not e or

O uso de palavra-chave inadequada geralmente resulta em um SyntaxError.

Blocos de Código

Os blocos de código são grupos de instruções que são utilizados onde instruções únicas são esperadas. Blocos de código podem seguir qualquer uma das instruções a seguir: if, elif, else, for, while, try, except, def e class. Essas instruções introduzem um código de cloco com o caractere dois pontos (:), por exemplo:

```
if x == 1:
    y = 2
    z = 3
elif:
    y = 4
    z = 5
```

A indentação é utilizada para delimitar os blocos de código (ao invés de chaves que são utilizadas em Java). Todas as linhas em um bloco devem ser indentadas para a mesma posição. Isso ocorre porque uma mudança na indentação indica o final de um bloco de códigos. É comum recuar por quatro espaços por nível. Recomenda-se que espaços sejam usados para indentar as linhas, ao invés de usar tabulações. Espaços e tabulações não devem ser misturados. As linhas no bloco mais afastado de um módulo devem iniciar na coluna um, ou um SyntaxError ocorrerá.

As instruções que compõem um bloco de códigos (e após os dois pontos) também podem estar em uma única linha, separadas por ponto e vírgula, por exemplo:

```
if x == 1: y = 2; z = 3;
```

Transmitindo Argumentos para um Script

Transmitir argumentos para um script é útil já que isso significa que um script pode ser utilizado repetidamente sem modificação. Os argumentos que são transmitidos na linha de comandos são transmitidos como valores na lista sys.argv. O número de valores transmitidos pode ser obtido utilizando o comando len(sys.argv). Por exemplo:

```
import sys
print "test1"
print sys.argv[0]
print sys.argv[1]
print len(sys.argv)
```

Neste exemplo, o comando import importa toda a classe sys para que os métodos existentes para essa classe, como argv, possam ser utilizados.

O script nesse exemplo pode ser chamado usando a linha a seguir:

```
/u/mjloos/test1 mike don
```

O resultado é a saída a seguir:

```
/u/mjloos/test1 mike don
test1
mike
don
3
```

Exemplos

A palavra-chave `print` imprime os argumentos imediatamente em seguida. Se a instrução for seguida por uma vírgula, uma nova linha não será incluída na saída. Por exemplo:

```
print "This demonstrates the use of a",
print " comma at the end of a print statement."
```

Isso resulta na saída a seguir:

```
This demonstrates the use of a comma at the end of a print statement.
```

A instrução `for` é utilizada para iterar através de um bloco de código. Por exemplo:

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    print lv
    continue
```

Neste exemplo, três sequências estão designadas à lista `mylist1`. Em seguida, os elementos da lista são impressos, com um elemento de cada linha. Isso resulta na saída a seguir:

```
one
two
three
```

Neste exemplo, o agente iterativo `lv` utiliza o valor de cada elemento na lista `mylist1` sucessivamente conforme o loop 'for' implementa o bloco de códigos de cada elemento. Um agente iterativo pode ser qualquer identificador válido de qualquer comprimento.

A instrução `if` é uma instrução condicional. Ela avalia a condição e retorna true ou false, dependendo do resultado da avaliação. Por exemplo:

```
mylist1 = ["one", "two", "three"]
for lv in mylist1:
    if lv == "two"
        print "The value of lv is ", lv
    else
        print "The value of lv is not two, but ", lv
    continue
```

Neste exemplo, o valor do iterador `lv` é avaliado. Se o valor de `lv` for `two`, uma sequência diferente será retornada para a sequência que for retornada se o valor de `lv` não for `two`. Isso resulta na saída a seguir:

```
The value of lv is not two, but one
The value of lv is two
The value of lv is not two, but three
```

Métodos Matemáticos

Do módulo `math`, é possível acessar métodos matemáticos úteis. Alguns desses métodos são fornecidos na tabela a seguir. A menos que seja especificado o contrário, todos os valores são retornados como flutuantes.

Tabela 7. Métodos matemáticos

Método	**Uso**
`math.ceil(x)`	Retorne o teto de x como um valor flutuante, que é o menor número inteiro maior ou igual a x
`math.copysign(x, y)`	Retorne x com o sinal de y. `copysign(1, -0.0)` retorna `-1`
`math.fabs(x)`	Retorne o valor absoluto de x

Tabela 7. Métodos matemáticos (continuação)	
Método	**Uso**
`math.factorial(x)`	Retorne x fatorial. Se x for negativo ou não um número inteiro, um `ValueError` será levantad.
`math.floor(x)`	Retorne o piso de x como um valor flutuante, que é o maior número inteiro menor ou igual a x
`math.frexp(x)`	Retorne a mantissa (`m`) e o expoente (e) de x como o par `(m, e)`. `m` é um valor flutuante e e é um número inteiro, tal qual `x == m * 2**e` exatamente. Se x for zero, retornará `(0.0, 0)`, caso contrário, `0.5 <= abs(m) < 1`.
`math.fsum(iterable)`	Retorne uma soma de ponto flutuante precisa de valores em `iterable`
`math.isinf(x)`	Verifique se o valor flutuante x é positivo ou negativo infinitivo
`math.isnan(x)`	Verifique se o valor flutuante x é `NaN` (não um número)
`math.ldexp(x, i)`	Retorne `x * (2**i)`. Este é essencialmente o inverso da função `frexp`.
`math.modf(x)`	Retorne as partes fracionadas e de número inteiro de x. Ambos os resultados carregam o sinal de x e são valores flutuantes.
`math.trunc(x)`	Retorne o `Real` valor x, que foi truncado para um `Integral`.
`math.exp(x)`	Retornar `e**x`
`math.log(x[, base])`	Retorne o logaritmo de x para o valor fornecido de `base`. Se `base` não for especificado, o logaritmo natural de x será retornado.
`math.log1p(x)`	Retorne o logaritmo natural de `1+x (base e)`
`math.log10(x)`	Retorne o logaritmo base-10 de x
`math.pow(x, y)`	Retorno x elevado à potência y. `pow(1.0, x)` e `pow(x, 0.0)` sempre retornam 1, mesmo quando x é zero ou NaN.
`math.sqrt(x)`	Retorne a raiz quadrada de x

Além das funções matemáticas, há alguns métodos trigonométricos úteis. Esses métodos são mostrados na tabela a seguir.

Tabela 8. Métodos trigonométricos	
Método	**Uso**
`math.acos(x)`	Retorne o arco cosseno de x em radianos
`math.asin(x)`	Retorne o arco seno de x em radianos
`math.atan(x)`	Retorne o arco tangente de x em radianos
`math.atan2(y, x)`	Retorne `atan(y / x)` em radianos.

Tabela 8. Métodos trigonométricos (continuação)	
Método	**Uso**
`math.cos(x)`	Retorne o cosseno de x em radianos.
`math.hypot(x, y)`	Retorne a norma Euclidiana `sqrt(x*x + y*y)`. Este é o comprimento do vetor desde a origem até o ponto `(x, y)`.
`math.sin(x)`	Retorne o seno de x em radianos
`math.tan(x)`	Retorne a tangente de x em radianos
`math.degrees(x)`	Converter ângulo x de radianos para graus
`math.radians(x)`	Converter ângulo x de graus para radianos
`math.acosh(x)`	Retorne o cosseno hiperbólico inverso de x
`math.asinh(x)`	Retorne o seno hiperbólico inverso de x
`math.atanh(x)`	Retorne a tangente hiperbólica inversa de x
`math.cosh(x)`	Retorne o cosseno hiperbólico de x
`math.sinh(x)`	Retorne o cosseno hiperbólico de x
`math.tanh(x)`	Retorne a tangente hiperbólica de x

Há também duas constantes matemáticas. O valor de `math.pi` é o constante matemático pi. O valor de `math.e` é o constante matemático e.

Utilizando caracteres não ASCII

Para utilizar caracteres não ASCII, o Python requer codificação e decodificação explícitas das sequências em Unicode. No IBM SPSS Modeler, os scripts Python são assumidos como estando codificados em UTF-8, que é uma codificação Unicode padrão que suporta caracteres não ASCII. O script a seguir realizará compilação porque o compilador Python foi configurado para UTF-8 pelo SPSS Modeler.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "テストノード", 96, 64)
```

No entanto, o nó resultante terá um rótulo incorreto.

Figura 3. Rótulo do nó contendo caracteres não ASCII exibido incorretamente

O rótulo está incorreto porque o literal de sequência em si foi convertido em uma sequência ASCII pelo Python.

O Python permite que os literais de sequência Unicode sejam especificados incluindo um prefixo de caracteres u antes da sequência literal:

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", u"テストノード", 96, 64)
```

Isso criará uma sequência Unicode e o rótulo será exibido corretamente.

Figura 4. Rótulo do nó contendo caracteres não ASCII exibido corretamente

O uso do Python e Unicode é um tópico grande que está além do escopo deste documento. Muitos livros e recursos online estão disponíveis para abordar este tópico em mais detalhes.

Programação Orientada a Objetos

A programação orientada a objetos é baseada na noção de criar um modelo do problema de destino em seus programas. A programação orientada a objetos reduz os erros de programação e promove a reutilização de código. O Python é uma linguagem orientada a objetos. Os objetos definidos em Python possuem os recursos a seguir:

- **Identidade.** Cada objeto deve ser distinto, e isso deve ser testável. Os testes `is` e `is not` existem para esta finalidade.
- **Estado.** Cada objeto deve ser capaz de armazenar estado. Atributos, como campos e variáveis de instância, existem para este propósito.
- **Comportamento.** Cada objeto deve ser capaz de manipular seu estado. Métodos existem para este propósito.

O Python inclui os recursos a seguir para suportar programação orientada a objetos:

- **Criação de objeto baseada em classe.** Classes são modelos para a criação de objetos. Os objetos são estruturas de dados com comportamento associado.
- **Herança com polimorfismo.** O Python suporta herança única e múltipla. Todos os métodos de instância Python são polimórficos e podem ser substituídos por subclasses.
- **Encapsulamento com ocultação de dados.** O Python permite que os atributos sejam ocultados. Quando ocultos, os atributos podem ser acessados a partir de fora da classe apenas por meio de métodos da classe. As classes implementam métodos para modificar os dados.

Definindo uma Classe

Em uma classe Python, variáveis e métodos podem ser definidos. Ao contrário do Java, o Python permite definir qualquer número de classes públicas por arquivo de origem (ou *módulo*). Portanto, um módulo em Python pode ser considerado semelhante a um pacote em Java.

Em Python, as classes são definidas usando a instrução `class`. A instrução `class` tem a forma a seguir:

```
class name (superclasses): statement
```

ou

```
class name (superclasses):
    assignment
    .
    .
    function
    .
    .
```

Ao definir uma classe, você tem a opção de fornecer zero ou mais instruções de *designação*. Isso cria atributos de classe que são compartilhados por todas as instâncias da classe. Também é possível

fornecer zero ou mais definições de *função*. Essas definições de função criam métodos. A lista de superclasses é opcional.

O nome de classe deve ser exclusivo no mesmo escopo, que está dentro de um módulo, função ou classe. É possível definir diversas variáveis para fazer referência à mesma classe.

Criando uma Instância de Classe

Classes são utilizadas para reter atributos de classe (ou compartilhados) ou para criar instâncias de classe. Para criar uma instância de uma classe, você chama a classe como se ela fosse uma função. Por exemplo, considere a classe a seguir:

```
class MyClass:
    pass
```

Aqui, a instrução `pass` é usada porque é necessária uma instrução para completar a classe, mas nenhuma ação é necessária programaticamente.

A instrução a seguir cria uma instância da classe `MyClass`:

```
x = MyClass()
```

Incluindo Atributos em uma Instância de Classe

Ao contrário de Java, nos clientes Python é possível incluir atributos em uma instância de uma classe. Apenas a instância é alterada. Por exemplo, para incluir atributos em uma instância `x`, configure novos valores nessa instância:

```
x.attr1 = 1
x.attr2 = 2
    .
    .
x.attrN = n
```

Definindo Atributos e Métodos de Classe

Qualquer variável que é ligada em uma classe é um *atributo de classe*. Qualquer função definida em uma classe é um *método*. Os métodos recebem uma instância da classe, convencionalmente chamada `self`, como o primeiro argumento. Por exemplo, para definir alguns atributos e métodos de classe, é possível inserir o seguinte código:

```
class MyClass
    attr1 = 10        #class attributes
    attr2 = "hello"

    def method1(self):
        print MyClass.attr1   #reference the class attribute

    def method2(self):
        print MyClass.attr2   #reference the class attribute

    def method3(self, text):
        self.text = text        #instance attribute
        print text, self.text   #print my argument and my attribute

    method4 = method3   #make an alias for method3
```

Dentro de uma classe, deve-se qualificar todas as referências a atributos de classe com o nome de classe; por exemplo, `MyClass.attr1`. Todas as referências a atributos de instância devem ser qualificadas com a variável `self`; por exemplo, `self.text`. Fora da classe, deve-se qualificar todas as referências a atributos de classe com o nome da classe (por exemplo, `MyClass.attr1`) ou com uma instância da classe (por exemplo `x.attr1`, em que `x` é uma instância da classe). Fora da classe, todas as referências a variáveis de instância devem ser qualificadas com uma instância da classe; por exemplo, `x.text`.

Variáveis ocultas

Os dados podem ser ocultados ao criar *Variáveis Privadas*. As variáveis privadas podem ser acessadas apenas pela própria classe. Se você declarar nomes no formato `__xxx` ou `__xxx_yyy`, ou seja, com dois sublinhados precedentes, o analisador Python incluirá automaticamente o nome da classe no nome declarado, criando variáveis ocultas, por exemplo:

```
class MyClass:
    __attr = 10   #private class attribute

    def method1(self):
        pass

    def method2(self, p1, p2):
        pass

    def __privateMethod(self, text):
        self.__text = text    #private attribute
```

Ao contrário do Java, no Python todas as referências às variáveis de instância devem ser qualificadas com `self`, e não há nenhum uso implícito de `this`.

herança

A capacidade de herdar a partir de classes é fundamental para programação orientada a objetos. O Python suporta herança única e também diversas heranças. *Herança única* significa que pode haver apenas uma superclasse. *Diversas heranças* significam que pode haver mais de uma superclasse.

A herança é implementada ao definir outras classes como subclasse. Qualquer número de classes Python pode ser superclasses. Na implementação Jython do Python, apenas uma classe Java pode ser herdada direta ou indiretamente. Ela não é necessária para que uma superclasse seja fornecida.

Qualquer atributo ou método em uma superclasse também está em qualquer subclasse e pode ser utilizado pela própria classe ou por qualquer cliente, desde que o atributo ou método não esteja oculto. Qualquer instância de uma subclasse poderá ser utilizada onde quer que a instância de uma superclasse possa ser utilizada; isso é um exemplo de *polimorfismo*. Esses recursos permitem a reutilização e a facilidade da extensão.

Exemplo

```
class Class1: pass    #no inheritance

class Class2: pass

class Class3(Class1): pass     #single inheritance

class Class4(Class3, Class2): pass     #multiple inheritance
```

Capítulo 3. Criando Script em IBM SPSS Modeler

Tipos de scripts

No IBM SPSS Modeler existem três tipos de script:

- Os *Scripts de fluxo* são utilizados para controlar a execução de um fluxo único e são armazenados no fluxo.
- Os *Scripts de SuperNode* são utilizados para controlar o comportamento dos SuperNodes.
- Os *Scripts independentes ou de sessão* podem ser utilizados para coordenar a execução através de um número de fluxos diferentes.

Vários métodos estão disponíveis para serem utilizados em scripts no IBM SPSS Modeler com a qual é possível acessar uma ampla variedade de funcionalidade do SPSS Modeler. Esses métodos também são utilizados no Capítulo 4, “A API de Script”, na página 39 para criar funções mais avançadas.

Fluxos, fluxos de SuperNode e diagramas

Na maioria das vezes, o termo *stream* significa a mesma coisa, independentemente se for um fluxo que é carregado a partir de um arquivo ou utilizado em um SuperNode. Geralmente significa uma coleção de nós que são conectados entre si e que podem ser executados. No script, no entanto, nem todas as operações são suportadas em todos os locais, significando que um autor de script deverá saber qual variante de fluxo ele está utilizando.

Fluxos

Um fluxo é o tipo de documento principal do IBM SPSS Modeler. Ele pode ser salvo, carregado, editado e executado. Os fluxos também podem ter parâmetros, valores globais, um script e outras informações associadas a ele.

Fluxos de SuperNode

Um *fluxo de SuperNode* é o tipo de fluxo utilizado em um SuperNode. Assim como um fluxo normal, ele contém nós que estão vinculados. Os fluxos de SuperNode possuem várias diferenças de um fluxo normal:

- Os parâmetros e quaisquer scripts são associados ao SuperNode que possui o fluxo de SuperNode e não ao próprio fluxo de SuperNode.
- Os fluxos de SuperNode possuem nós de conector de entrada e de saída adicionais, dependendo do tipo de SuperNode. Esses nós de conector são utilizados para fluir informações para dentro e fora do fluxo do SuperNode e são criados automaticamente quando o SuperNode é criado.

Diagramas

O termo *diagrama* abrange as funções que são suportadas pelos fluxos normal e SuperNode, como incluir e remover nós e modificar conexões entre os nós.

Executando um fluxo

O exemplo a seguir executa todos os nós executáveis no fluxo e é o tipo mais simples de script de fluxo:

```
modeler.script.stream().runAll(None)
```

O exemplo a seguir também executa todos os nós executáveis no fluxo:

```
stream = modeler.script.stream()
stream.runAll(None)
```

Neste exemplo, o fluxo é armazenado em uma variável denominada `stream`. Armazenar o fluxo em uma variável é útil porque um script é normalmente utilizado para modificar o fluxo ou os nós dentro de um fluxo. Criar uma variável que armazena o fluxo resulta em um script mais conciso.

O contexto de script

O módulo `modeler.script` fornece o contexto no qual um script é executado. O módulo é automaticamente importado em um script do SPSS Modeler no tempo de execução. O módulo define quatro funções que fornecem um script com acesso ao seu ambiente de execução.

- A função `session()` retorna a sessão para o script. A sessão define informações como o código do idioma e o SPSS Modeler de backend (um processo local ou um SPSS Modeler Server em rede) que está sendo utilizado para executar quaisquer fluxos.
- A função `stream()` pode ser utilizada com os scripts de fluxo e de SuperNode. Esta função retorna o fluxo que possui ou o script de fluxo ou o script de SuperNode que está sendo executado.
- A função `diagram()` pode ser usada com scripts de Supernó. Esta função retorna o diagrama no SuperNode. Para outros tipos de script, essa função retorna o mesmo que a função `stream()`.
- A função `supernode()` pode ser usada com scripts de Supernó. Esta função retorna o SuperNode que possuir o script que está sendo executado.

As quatro funções e suas saídas são resumidas na tabela a seguir.

Tabela 9. Resumo de funções `modeler.script`

Tipo de script	**`session()`**	**`stream()`**	**`diagram()`**	**`supernode()`**
Independente	Retorna uma sessão	Retorna o fluxo gerenciado atual no momento em que o script foi chamado (por exemplo, o fluxo transmitido por meio da opção `-stream` do modo em lote), ou `None`.	Mesmo que para `stream()`	Não aplicável
Fluxo	Retorna uma sessão	Retorna um fluxo	Mesmo que para `stream()`	Não aplicável
SuperNode	Retorna uma sessão	Retorna um fluxo	Retorna um fluxo de SuperNode	Retorna um SuperNode

O módulo `modeler.script` também define uma maneira de finalizar o script com um código de saída. A função `exit(`*`exit-code`*`)` interrompe a execução do script e retorna o código de saída de número inteiro fornecido.

Um dos métodos definidos para um fluxo é `runAll(List)`. Este método executa todos os nós executáveis. Quaisquer modelos ou saídas que forem gerados executando os nós são incluídos na lista fornecida.

Normalmente uma execução de fluxo gera saídas, como modelos, gráficos e outra saída. Para capturar esta saída, um script pode fornecer uma variável que seja inicializada para uma lista, por exemplo:

```
stream = modeler.script.stream()
results = []
stream.runAll(results)
```

Quando a execução está completa, quaisquer objetos gerados pela execução podem ser acessados pela lista `results`.

Referenciando nós existentes

Um fluxo é geralmente pré-construído com alguns parâmetros que devem ser modificados antes do fluxo ser executado. Modificar esses parâmetros envolve as tarefas a seguir:

1. Localizar os nós no fluxo relevante.
2. Alterando as configurações de nó ou de fluxo (ou ambos).

Localizando nós

Os fluxos fornecem várias maneiras de localizar um nó existente. Esses métodos são resumidos na tabela a seguir.

Tabela 10. Métodos para localizar um nó existente

Método	Tipo de retorno	Descrição
`s.findAll(type, label)`	Coleção	Retorna uma lista de todos os nós com o tipo e rótulo especificados. O tipo ou o rótulo pode ser None, caso em que o outro parâmetro é usado.
`s.findAll(filter, recursive)`	Coleção	Retorna uma coleção de todos os nós que forem aceitos pelo filtro especificado. Se o sinalizador recursivo for `True`, quaisquer SuperNodes no fluxo especificado também serão procurados.
`s.findByID(id)`	Nó	Retorna o nó com o ID fornecido ou None se nenhum desse nó existir. A procura é limitada ao fluxo atual.
`s.findByType(type, label)`	Nó	Retorna o nó com o tipo ou rótulo fornecido, ou ambos. O tipo ou o nome pode ser None, caso em que o outro parâmetro é usado. Se diversos nós resultarem em uma correspondência, uma correspondência arbitrária será escolhida e retornada. Se nenhum nó resultar em uma correspondência, o valor de retorno será None.

Tabela 10. Métodos para localizar um nó existente (continuação)		
Método	**Tipo de retorno**	**Descrição**
`s.findDownstream(fromNodes)`	Coleção	Procura a partir da lista de nós fornecida e retorna o conjunto de nós de recebimento de dados dos nós fornecidos. A lista retornada inclui os nós originalmente fornecidos.
`s.findUpstream(fromNodes)`	Coleção	Procura a partir da lista de nós fornecida e retorna o conjunto de nós de envio de dados dos nós fornecidos. A lista retornada inclui os nós originalmente fornecidos.
`s.findProcessorForID(String id, boolean recursive)`	Nó	Retorna o nó com o ID fornecido ou None se nenhum desse nó existir. Se a sinalização recursiva for `true`, quaisquer nós compostos dentro deste diagrama também serão procurados.

Como um exemplo, se o fluxo continha um nó Filtro único que o script precisava acessar, o nó Filtro poderá ser localizado usando o script a seguir:

```
stream = modeler.script.stream()
node = stream.findByType("filter", None)
...
```

Como alternativa, se o ID do nó (conforme mostrado na guia Anotações da caixa de diálogo do nó) for conhecido, o ID poderá ser utilizado para localizar o nó, por exemplo:

```
stream = modeler.script.stream()
node = stream.findByID("id32FJT71G2")  # the filter node ID
...
```

Configurando propriedades

Os nós, fluxos, modelos e saídas possuem propriedades que podem ser acessadas e, na maioria dos casos, configuradas. As propriedades geralmente são utilizadas para modificar o comportamento ou a aparência do objeto. Os métodos que estão disponíveis para acessar e configurar as propriedades do objeto são resumidos na tabela a seguir.

Tabela 11. Métodos para acessar e configurar propriedades de objeto		
Método	**Tipo de retorno**	**Descrição**
`p.getPropertyValue(propertyName)`	Objeto	Retorna o valor da propriedade nomeada ou None se não existir tal propriedade.
`p.setPropertyValue(propertyName, value)`	Não aplicável	Configura o valor da propriedade nomeada.

Tabela 11. Métodos para acessar e configurar propriedades de objeto (continuação)		
Método	**Tipo de retorno**	**Descrição**
`p.setPropertyValues(properties)`	Não aplicável	Configura os valores das propriedades nomeadas. Cada entrada no mapa de propriedades consiste em uma chave que representa o nome da propriedade e o valor que deve ser designado a essa propriedade.
`p.getKeyedPropertyValue(propertyName, keyName)`	Objeto	Retorna o valor da propriedade nomeada e a chave associada ou None se não existir tal propriedade ou chave.
`p.setKeyedPropertyValue(propertyName, keyName, value)`	Não aplicável	Configura o valor da propriedade nomeada e da chave.

Por exemplo, se desejar configurar o valor de um nó Arquivo da Variável no início de um fluxo, será possível utilizar o seguinte script:

```
stream = modeler.script.stream()
node = stream.findByType("variablefile", None)
node.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
...
```

Como alternativa, você pode querer filtrar um campo a partir de um nó Filtro. Nesse caso, o valor também é chaveado no nome do campo, por exemplo:

```
stream = modeler.script.stream()
# Locate the filter node ...
node = stream.findByType("filter", None)
# ... and filter out the "Na" field
node.setKeyedPropertyValue("include", "Na", False)
```

Criando nós e modificando fluxos

Em algumas situações, você pode querer incluir novos nós em fluxos existentes. Incluir nós em fluxos existentes geralmente envolve as tarefas a seguir:

1. Criando os nós.
2. Vinculando os nós no fluxo existente.

Criando nós

Os fluxos fornecem várias maneiras de criar nós. Esses métodos são resumidos na tabela a seguir.

Tabela 12. Métodos para criação de nós		
Método	**Tipo de retorno**	**Descrição**
`s.create(nodeType, name)`	Nó	Cria um nó do tipo especificado e o inclui no fluxo especificado.
`s.createAt(nodeType, name, x, y)`	Nó	Cria um nó do tipo especificado e o inclui no fluxo especificado no local especificado. Se x < 0 ou y < 0,o local não será configurado.

Tabela 12. Métodos para criação de nós (continuação)		
Método	**Tipo de retorno**	**Descrição**
`s.createModelApplier(modelOutput, name)`	Nó	Cria um nó de aplicador de modelo que é derivado do objeto de saída do modelo fornecido.

Por exemplo, para criar um novo nó Tipo em um fluxo, é possível utilizar o script a seguir:

```
stream = modeler.script.stream()
# Create a new type node
node = stream.create("type", "My Type")
```

Vinculando e desvinculando nós

Quando um novo nó é criado dentro de um fluxo, ele deve ser conectado a uma sequência de nós antes de poder ser utilizado. Os fluxos fornecem vários métodos para vincular e desvincular nós. Esses métodos são resumidos na tabela a seguir.

Tabela 13. Métodos para vincular e desvincular nós		
Método	**Tipo de retorno**	**Descrição**
`s.link(source, target)`	Não aplicável	Cria um novo link entre os nós de origem e de destino.
`s.link(source, targets)`	Não aplicável	Cria novos links entre o nó de origem e cada nó de destino na lista fornecida.
`s.linkBetween(inserted, source, target)`	Não aplicável	Conecta um nó entre duas outras instâncias do nó (os nós de origem e de destino) e configura a posição do nó inserido para que ele esteja entre as instâncias. Qualquer link direto entre os nós de origem e de destino é removido inicialmente.
`s.linkPath(path)`	Não aplicável	Cria um novo caminho entre instâncias do nó. O primeiro nó é vinculado ao segundo, o segundo é vinculado ao terceiro, e assim por diante.
`s.unlink(source, target)`	Não aplicável	Remove qualquer link direto entre nós de origem e de destino.
`s.unlink(source, targets)`	Não aplicável	Remove quaisquer links diretos entre o nó de origem e cada objeto na lista de destinos.
`s.unlinkPath(path)`	Não aplicável	Remove qualquer caminho que existir entre instâncias do nó.
`s.disconnect(node)`	Não aplicável	Remove quaisquer links entre o nó fornecido e quaisquer outros nós no fluxo especificado.

Tabela 13. Métodos para vincular e desvincular nós (continuação)		
Método	**Tipo de retorno**	**Descrição**
`s.isValidLink(source, target)`	*Booleano*	Retorna `True` se for válido para criar um link entre os nós de origem e destino especificados. Esse método verifica se ambos os objetos pertencem ao fluxo especificado, se o nó de origem pode fornecer um link e o nó de destino pode receber um link e se criar esse link não causará uma circularidade no fluxo.

O script de exemplo a seguir executa estas cinco tarefas:

1. Cria um nó de entrada Arquivo Variável, um nó Filtro e um nó de saída Tabela.
2. Conecta os nós entre si.
3. Configura o nome de arquivo no nó de entrada de Arquivo Variável.
4. Filtra o campo "Drug" na saída resultante.
5. Executa o nó Tabela.

```
stream = modeler.script.stream()
filenode = stream.createAt("variablefile", "My File Input ", 96, 64)
filternode = stream.createAt("filter", "Filter", 192, 64)
tablenode = stream.createAt("table", "Table", 288, 64)
stream.link(filenode, filternode)
stream.link(filternode, tablenode)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
filternode.setKeyedPropertyValue("include", "Drug", False)
results = []
tablenode.run(results)
```

Importando, substituindo e excluindo nós

Assim como criar e conectar nós, normalmente é necessário substituir e excluir nós do fluxo. Os métodos que estão disponíveis para importar, substituir e excluir nós são resumidos na tabela a seguir.

Tabela 14. Métodos para importar, substituir e excluir nós		
Método	**Tipo de retorno**	**Descrição**
`s.replace(originalNode, replacementNode, discardOriginal)`	Não aplicável	Substitui o nó especificado a partir do fluxo especificado. Tanto o nó original quanto o nó de substituição devem pertencer ao fluxo especificado.

Tabela 14. Métodos para importar, substituir e excluir nós (continuação)		
Método	**Tipo de retorno**	**Descrição**
`s.insert(source, nodes, newIDs)`	Lista	Insere cópias dos nós na lista fornecida. Assume-se que todos os nós na lista fornecida estejam contidos no fluxo especificado. A sinalização `newIDs` indica se novos IDs devem ser gerados para cada nó ou se o ID existente deve ser copiado e usado. Assume-se que todos os nós em um fluxo tenham um ID exclusivo, portanto, este sinalizador deverá ser configurado como `True` se o fluxo de origem for o mesmo que o fluxo especificado. O método retorna a lista de nós recém-inseridos, em que a ordem dos nós é indefinida (ou seja, a ordenação não é necessariamente a mesma dos nós na lista de entrada).
`s.delete(node)`	Não aplicável	Exclui o nó especificado do fluxo especificado. O nó deve pertencer ao fluxo especificado.
`s.deleteAll(nodes)`	Não aplicável	Exclui todos os nós especificados do fluxo especificado. Todos os nós na coleção devem pertencer ao fluxo especificado.
`s.clear()`	Não aplicável	Exclui todos os nós do fluxo especificado.

Percorrendo os nós em um fluxo

Um requisito comum é identificar os nós que forem envio ou recebimento de dados de um nó específico. O fluxo fornece diversos métodos que podem ser utilizados para identificar esses nós. Esses métodos são resumidos na tabela a seguir.

Tabela 15. Métodos para identificar os nós de envio e de recebimento de dados		
Método	**Tipo de retorno**	**Descrição**
`s.iterator()`	Agente iterativo	Retorna um agente iterativo sobre os objetos de nó que estão contidos no fluxo especificado. Se o fluxo for modificado entre as chamadas da função `next()`, o comportamento do agente iterativo será indefinido.
`s.predecessorAt(node, index)`	Nó	Retorna o predecessor imediato especificado do nó fornecido ou None se o índice estiver fora de limites.

Tabela 15. Métodos para identificar os nós de envio e de recebimento de dados (continuação)		
Método	**Tipo de retorno**	**Descrição**
`s.predecessorCount(node)`	*int*	Retorna o número de predecessores imediatos do nó fornecido.
`s.predecessors(node)`	Lista	Retorna os predecessores imediatos do nó fornecido.
`s.successorAt(node, index)`	Nó	Retorna o sucessor imediato especificado do nó fornecido ou None se o índice estiver fora de limites.
`s.successorCount(node)`	*int*	Retorna o número de sucessores imediatos do nó fornecido.
`s.successors(node)`	Lista	Retorna os sucessores imediatos do nó fornecido.

Limpando ou removendo itens

O script legado suporta vários usos do comando `clear`, por exemplo:

- `clear outputs` Exclui todos os itens de saída da paleta do gerenciador.
- `clear generated palette` Limpa todos os nuggets do modelo a partir da paleta de Modelos.
- `clear stream` Remove o conteúdo de um fluxo.

O script Python suporta um conjunto semelhante de funções; o comando `removeAll()` é utilizado para limpar gerenciadores de Fluxos, Saídas e de Modelos, por exemplo:

- Para limpar o gerenciador de Fluxos:

```
session = modeler.script.session()
session.getStreamManager.removeAll()
```

- Para limpar o gerenciador de Saídas:

```
session = modeler.script.session()
session.getDocumentOutputManager().removeAll()
```

- Para limpar o gerenciador de Modelos:

```
session = modeler.script.session()
session.getModelOutputManager().removeAll()
```

Obtendo informações sobre nós

Os nós se dividem em categorias diferentes, como nós de importação e exportação de dados, nós de construção de modelo, e outros tipos de nós. Cada nó fornece diversos métodos que podem ser utilizados para descobrir informações sobre o nó.

Os métodos que podem ser utilizados para obter o ID, o nome e o rótulo de um nó são resumidos na tabela a seguir.

Tabela 16. Métodos para obter o ID, o nome e o rótulo de um nó		
Método	**Tipo de retorno**	**Descrição**
`n.getLabel()`	*sequência*	Retorna o rótulo de exibição do nó especificado. O rótulo é o valor da propriedade `custom_name` apenas se essa propriedade for uma sequência não vazia e a propriedade `use_custom_name` não for configurada; caso contrário, o rótulo será o valor de `getName()`.
`n.setLabel(label)`	Não aplicável	Configura o rótulo de exibição do nó especificado. Se o novo rótulo for uma sequência não vazia, ele será designada à propriedade `custom_name`, e `False` será designado à propriedade `use_custom_name` para que o rótulo especificado tenha precedência; caso contrário, uma sequência vazia será designada à propriedade `custom_name` e `True` será designado à propriedade `use_custom_name`.
`n.getName()`	*sequência*	Retorna o nome do nó especificado.
`n.getID()`	*sequência*	Retorna o ID do nó especificado. Um novo ID é criado sempre que um novo nó for criado. O ID é persistido com o nó quando ele é salvo como parte de um fluxo, de modo que os IDs de nó sejam preservados quando o fluxo for aberto. Entretanto, se um nó salvo for inserido em um fluxo, o nó inserido será considerado um novo objeto e será alocado um novo ID.

Os métodos que podem ser utilizados para obter outras informações sobre um nó são resumidos na tabela a seguir.

Tabela 17. Métodos para obter informações sobre um nó		
Método	**Tipo de retorno**	**Descrição**
`n.getTypeName()`	*sequência*	Retorna o nome de script deste nó. Este é o mesmo nome que pode ser utilizado para criar uma nova instância deste nó.
`n.isInitial()`	*Booleano*	Retornará `True` se este for um nó *inicial*, que é aquele que ocorre no início de um fluxo.

Tabela 17. Métodos para obter informações sobre um nó (continuação)		
Método	**Tipo de retorno**	**Descrição**
`n.isInline()`	*Booleano*	Retornará `True` se este for um nó *sequencial*, que é aquele que ocorre no meio do fluxo.
`n.isTerminal()`	*Booleano*	Retornará `True` se este for um nó *terminal*, que é aquele que ocorre no término de um fluxo.
`n.getXPosition()`	*int*	Retorna o deslocamento da posição x do nó no fluxo.
`n.getYPosition()`	*int*	Retorna o deslocamento da posição y do nó no fluxo.
`n.setXYPosition(x, y)`	Não aplicável	Configura a posição do nó no fluxo.
`n.setPositionBetween(source, target)`	Não aplicável	Configura a posição do nó no fluxo para que ele seja posicionado entre os nós fornecidos.
`n.isCacheEnabled()`	*Booleano*	Retorna `True` se o cache estiver ativado; retorna `False` caso contrário.
`n.setCacheEnabled(val)`	Não aplicável	Ativa ou desativa o cache para este objeto. Se o cache estiver cheio e o armazenamento em cache estiver desativado, o cache será limpo.
`n.isCacheFull()`	*Booleano*	Retorna `True` se o cache estiver cheio; retorna `False` caso contrário.
`n.flushCache()`	Não aplicável	Limpa o cache desse nó. Não terá efeito se o cache não estiver ativado ou não estiver cheio.

Capítulo 4. A API de Script

Introdução à API de Script

A API de Script fornece acesso a uma ampla variedade de funcionalidade do SPSS Modeler. Todos os métodos descritos até o momento fazem parte da API e podem ser acessados implicitamente no script sem importações adicionais. No entanto, se desejar fazer referência às classes de API, deve-se importar a API explicitamente com a seguinte instrução:

```
import modeler.api
```

Esta instrução de importação é necessária por muitos exemplos da API de Script.

Um guia completo para as classes, métodos e parâmetros que estão disponíveis por meio da API de script pode ser localizado no documento *IBM SPSS Modeler Python Guia de Referência da API de Script*.

Exemplo 1: procurando nós usando um filtro customizado

A seção "Localizando nós" na página 29 inclui um exemplo de procura de um nó em um fluxo utilizando o nome do tipo do nó como o critério de procura. Em algumas situações, uma procura mais genérica é necessária, que poderá ser implementada utilizando a classe `NodeFilter` e o método `findAll()` do fluxo. Este tipo de procura envolve as duas etapas a seguir:

1. Criar uma nova classe que estende `NodeFilter` e que implementa uma versão customizada do método `accept()`.
2. Chamada do método `findAll()` do fluxo com uma instância dessa nova classe. Isso retorna todos os nós que atendem aos critérios definidos no método `accept()`.

O exemplo a seguir mostra como procurar por nós em um fluxo que tiver o cache de nó ativado. A lista de nós retornada pode ser utilizada para limpar ou desativar os caches desses nós.

```
import modeler.api

class CacheFilter(modeler.api.NodeFilter):
    """A node filter for nodes with caching enabled"""
    def accept(this, node):
        return node.isCacheEnabled()

cachingnodes = modeler.script.stream().findAll(CacheFilter(), False)
```

Exemplo 2: permitindo que os usuários obtenham informações de diretório ou arquivo com base em seus privilégios

Para evitar que o PSAPI seja aberto aos usuários, um método chamado `session.getServerFileSystem()` pode ser usado por meio da chamada da função PSAPI para criar um objeto do sistema de arquivos.

O exemplo a seguir mostra como permitir que um usuário obtenha informações de diretório ou de arquivo com base nos privilégios do usuário que se conecta ao IBM SPSS Modeler Server

```
import modeler.api
stream = modeler.script.stream()
sourceNode = stream.findByID('')
session = modeler.script.session()
fileSystem = session.getServerFileSystem()
parameter = stream.getParameterValue('VPATH')
serverDirectory = fileSystem.getServerFile(parameter)
files = fileSystem.getFiles(serverDirectory)
for f in files:
    if f.isDirectory():
```

```
        print 'Directory:'
    else:
        print 'File:'
        sourceNode.setPropertyValue('full_filename',f.getPath())
        break
    print f.getName(),f.getPath()
stream.execute()
```

Metadados: Informações sobre dados

Como os nós são conectados entre si em um fluxo, informações sobre as colunas ou campos disponíveis em cada nó são exibidas. Por exemplo, na IU do Modelador, isto permite selecionar os campos pelos quais classificar ou agregar. Essas informações são chamadas de modelo de dados.

Os scripts também podem acessar o modelo de dados ao consultar os campos que entram ou que saem de um nó. Para alguns nós, os modelos de dados de entrada e de saída são os mesmos, por exemplo, um nó Classificar apenas reordena os registros, mas não altera o modelo de dados. Alguns nós, como Derivar, podem incluir novos campos. Outros, como o nó Filtro, podem renomear ou remover campos.

No exemplo a seguir, o script usa o fluxo padrão do IBM SPSS Modeler `druglearn.str` e, para cada campo, constrói um modelo com um dos campos de entrada descartado Ele faz isto ao:

1. Acessar o modelo de dados de saída do nó Tipo.
2. Executar loop em cada campo no modelo de dados de saída.
3. Modificar o nó Filtro para cada campo de entrada.
4. Alterar o nome do modelo que está sendo construído.
5. Executar o nó de construção de modelo.

Nota: Antes de executar o script no fluxo `druglean.str` , lembre-se de configurar a linguagem de script para Python (o fluxo foi criado em uma versão anterior do IBM SPSS Modeler para que a linguagem de script de fluxo seja configurada como Legado)...

```
import modeler.api

stream = modeler.script.stream()
filternode = stream.findByType("filter", None)
typenode = stream.findByType("type", None)
c50node = stream.findByType("c50", None)
# Always use a custom model name
c50node.setPropertyValue("use_model_name", True)

lastRemoved = None
fields = typenode.getOutputDataModel()
for field in fields:
    # If this is the target field then ignore it
    if field.getModelingRole() == modeler.api.ModelingRole.OUT:
        continue

    # Re-enable the field that was most recently removed
    if lastRemoved != None:
        filternode.setKeyedPropertyValue("include", lastRemoved, True)

    # Remove the field
    lastRemoved = field.getColumnName()
    filternode.setKeyedPropertyValue("include", lastRemoved, False)

    # Set the name of the new model then run the build
    c50node.setPropertyValue("model_name", "Exclude " + lastRemoved)
    c50node.run([])
```

O objeto DataModel fornece vários métodos para acessar informações sobre os campos ou colunas no modelo de dados. Esses métodos são resumidos na tabela a seguir.

Tabela 18. Métodos do objeto DataModel para acessar informações sobre campos ou colunas		
Método	**Tipo de retorno**	**Descrição**
`d.getColumnCount()`	*int*	Retorna o número de colunas no modelo de dados.
`d.columnIterator()`	Agente iterativo	Retorna um agente iterativo que retorna cada coluna na ordem de inserção "natural". O agente iterativo retorna instâncias de Coluna.
`d.nameIterator()`	Agente iterativo	Retorna um agente iterativo que retorna o nome de cada coluna na ordem de inserção "natural".
`d.contains(name)`	*Booleano*	Retorna `True` se uma coluna com o nome fornecido existe neste DataModel; `False`, caso contrário.
`d.getColumn(name)`	Coluna	Retorna a coluna com o nome especificado.
`d.getColumnGroup(name)`	ColumnGroup	Retorna o grupo de colunas nomeado ou None se nenhum grupo de coluna existir.
`d.getColumnGroupCount()`	*int*	Retorna o número de grupos de coluna nesse modelo de dados.
`d.columnGroupIterator()`	Agente iterativo	Retorna um agente iterativo que retorna cada grupo de colunas sucessivamente.
`d.toArray()`	Coluna []	Retorna o modelo de dados como uma matriz de colunas. As colunas são ordenadas em sua ordem de inserção "natural".

Cada campo (objeto da Coluna) inclui vários métodos para acessar informações sobre a coluna. A tabela abaixo mostra uma seleção deles.

Tabela 19. Métodos de objetos Coluna para acessar informações sobre a coluna		
Método	**Tipo de retorno**	**Descrição**
`c.getColumnName()`	*sequência*	Retorna o nome da coluna.
`c.getColumnLabel()`	*sequência*	Retorna o rótulo da coluna ou uma sequência de caracteres vazia se nenhum rótulo estiver associado à coluna.
`c.getMeasureType()`	MeasureType	Retorna o tipo de medida para a coluna.
`c.getStorageType()`	StorageType	Retorna o tipo de armazenamento para a coluna.

Tabela 19. Métodos de objetos Coluna para acessar informações sobre a coluna (continuação)

Método	Tipo de retorno	Descrição
`c.isMeasureDiscrete()`	*Booleano*	Retorna `True` se a coluna for discreta. As colunas que forem um conjunto ou um sinalizador são consideradas discretas.
`c.isModelOutputColumn()`	*Booleano*	Retorna `True` se a coluna for uma coluna de saída de modelo.
`c.isStorageDatetime()`	*Booleano*	Retorna `True` se o armazenamento da coluna for um valor de tempo, data ou registro de data e hora.
`c.isStorageNumeric()`	*Booleano*	Retorna `True` se o armazenamento da coluna for um número inteiro ou um número real.
`c.isValidValue(value)`	*Booleano*	Retorna `True` se o valor especificado é válido para este armazenamento e `valid` quando os valores da coluna válidos são conhecidos.
`c.getModelingRole()`	ModelingRole	Retorna o tipo de modelagem para a coluna.
`c.getSetValues()`	Objeto []	Retorna uma matriz de valores válidos para a coluna ou None se os valores não são conhecidos ou a coluna não é um conjunto.
`c.getValueLabel(value)`	*sequência*	Retorna o rótulo para o valor na coluna ou uma sequência vazia se não houver nenhum rótulo associado ao valor.
`c.getFalseFlag()`	Objeto	Retorna o valor do indicador "false" para a coluna ou None se o valor não é conhecido ou a coluna não é uma sinalização.
`c.getTrueFlag()`	Objeto	Retorna o valor do indicador "true" para a coluna ou None se o valor não é conhecido ou a coluna não é uma sinalização.
`c.getLowerBound()`	Objeto	Retorna o valor de limite inferior para os valores na coluna ou None se o valor não é conhecido ou a coluna não é contínua.
`c.getUpperBound()`	Objeto	Retorna o valor de limite superior para os valores na coluna ou None se o valor não é conhecido ou a coluna não é contínua.

Observe que a maioria dos métodos que acessam informações sobre uma coluna possui métodos equivalentes definidos no próprio objeto DataModel. Por exemplo, as duas instruções a seguir são equivalentes:

```
dataModel.getColumn("someName").getModelingRole()
dataModel.getModelingRole("someName")
```

Acessando Objetos Gerados

Executar um fluxo geralmente envolve produzir objetos de saída adicionais. Esses objetos adicionais podem ser um novo modelo ou a uma parte da saída que fornece informações a serem utilizadas em execuções subsequentes.

No exemplo abaixo, o fluxo `druglearn.str` é utilizado novamente como o ponto de início para o fluxo. Neste exemplo, todos os nós no fluxo são executados e os resultados são armazenados em uma lista. Em seguida, o script efetua loop por meio dos resultados e quaisquer saídas de modelo que resultem da execução são salvas como um arquivo de modelo IBM SPSS Modeler (`.gm`) e o modelo é exportado por PMML

```
import modeler.api

stream = modeler.script.stream()

# Set this to an existing folder on your system.
# Include a trailing directory separator
modelFolder = "C:/temp/models/"

# Execute the stream
models = []
stream.runAll(models)

# Save any models that were created
taskrunner = modeler.script.session().getTaskRunner()
for model in models:
    # If the stream execution built other outputs then ignore them
    if not(isinstance(model, modeler.api.ModelOutput)):
        continue

    label = model.getLabel()
    algorithm = model.getModelDetail().getAlgorithmName()

    # save each model...
    modelFile = modelFolder + label + algorithm + ".gm"
    taskrunner.saveModelToFile(model, modelFile)

    # ...and export each model PMML...
    modelFile = modelFolder + label + algorithm + ".xml"
    taskrunner.exportModelToFile(model, modelFile, modeler.api.FileFormat.XML)
```

A classe executora de tarefa fornece uma maneira conveniente de executar várias tarefas comuns. Os métodos que estão disponíveis nesta classe são resumidos na tabela a seguir.

Tabela 20. Métodos da classe executora de tarefas para executar tarefas comuns

Método	Tipo de retorno	Descrição
`t.createStream(name, autoConnect, autoManage)`	Fluxo	Cria e retorna um novo fluxo. Observe que o código que deve criar fluxos de modo privativo sem torná-los visíveis para o usuário deve configurar o sinalizador `autoManage` para `False`.
`t.exportDocumentToFile( documentOutput, filename, fileFormat)`	Não aplicável	Exporta a descrição do fluxo em um arquivo utilizando o formato de arquivo especificado.

Tabela 20. Métodos da classe executora de tarefas para executar tarefas comuns (continuação)

Método	Tipo de retorno	Descrição
`t.exportModelToFile(modelOutput, filename, fileFormat)`	Não aplicável	Exporta o modelo em um arquivo utilizando o formato de arquivo especificado.
`t.exportStreamToFile(stream, filename, fileFormat)`	Não aplicável	Exporta o fluxo em um arquivo utilizando o formato de arquivo especificado.
`t.insertNodeFromFile(filename, diagram)`	Nó	Lê e retorna um nó a partir do arquivo especificado, inserindo-o no diagrama fornecido. Observe que isso pode ser utilizado para ler ambos os objetos de Nó e SuperNode.
`t.openDocumentFromFile(filename, autoManage)`	DocumentOutput	Lê e retorna um documento a partir do arquivo especificado.
`t.openModelFromFile(filename, autoManage)`	ModelOutput	Lê e retorna um documento a partir do arquivo especificado.
`t.openStreamFromFile(filename, autoManage)`	Fluxo	Lê e retorna um fluxo a partir do arquivo especificado.
`t.saveDocumentToFile(documentOutput, filename)`	Não aplicável	Salva o documento no local do arquivo especificado.
`t.saveModelToFile(modelOutput, filename)`	Não aplicável	Salva o modelo no local do arquivo especificado.
`t.saveStreamToFile(stream, filename)`	Não aplicável	Salva o fluxo no local do arquivo especificado.

Manipulando erros

A linguagem Python fornece manipulação de erros por meio do bloco de código `try...except`. Isso pode ser utilizado dentro de scripts para capturar exceções e manipular problemas que, de outra forma, fazem com que o script seja finalizado.

No script de exemplo abaixo, uma tentativa é feita para recuperar um modelo de um IBM SPSS Collaboration and Deployment Services Repository. Essa operação pode fazer com que uma exceção seja emitida, por exemplo, se as credenciais de login de repositório não tiverem sido configuradas corretamente ou se o caminho do repositório estiver errado. No script, isso pode fazer com que um `ModelerException` seja lançado (todas as exceções que são geradas por IBM SPSS Modeler são derivadas de `modeler.api.ModelerException`.

```
import modeler.api

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
```

Nota: Algumas operações de script podem fazer com que exceções Java padrão sejam emitidas; essas exceções não são derivadas de `ModelerException`. Para capturar essas exceções, um bloco de exceção adicional poderá ser usado para capturar todas as exceções Java, por exemplo:

```
import modeler.api
import java.lang.Exception

session = modeler.script.session()
try:
    repo = session.getRepository()
    m = repo.retrieveModel("/some-non-existent-path", None, None, True)
    # print goes to the Modeler UI script panel Debug tab
    print "Everything OK"
except modeler.api.ModelerException, e:
    print "An error occurred:", e.getMessage()
except java.lang.Exception, e:
    print "A Java exception occurred:", e.getMessage()
```

Parâmetros de Fluxo, Sessão e SuperNode

Os parâmetros fornecem uma maneira útil de transmitir valores no tempo de execução ao invés de codificá-los permanentemente direto em um script. Os parâmetros e seus valores são definidos da mesma maneira dos fluxos, ou seja, como entradas na tabela de parâmetros de um fluxo ou SuperNode ou como parâmetros na linha de comandos. As classes de Fluxo e SuperNode implementam um conjunto de funções definidas pelo objeto ParameterProvider conforme mostrado na tabela a seguir. A sessão fornece uma chamada de `getParameters()` que retorna um objeto que define essas funções.

Tabela 21. Funções definidas pelo objeto ParameterProvider

Método	Tipo de retorno	Descrição
`p.parameterIterator()`	Agente iterativo	Retorna um agente iterativo de nomes de parâmetro para este objeto.
`p.getParameterDefinition(parameterName)`	ParameterDefinition	Retorna a definição de parâmetro para o parâmetro com o nome especificado ou None se nenhum parâmetro desse tipo existir nesse provedor. O resultado poderá ser uma captura instantânea da definição no momento em que o método foi chamado e não precisará refletir nenhuma modificação subsequente feita no parâmetro por meio deste provedor.
`p.getParameterLabel(parameterName)`	*sequência*	Retorna o rótulo do parâmetro nomeado ou None se não existir tal parâmetro.
`p.setParameterLabel(parameterName, label)`	Não aplicável	Configura o rótulo do parâmetro nomeado.
`p.getParameterStorage(parameterName)`	ParameterStorage	Retorna o armazenamento do parâmetro nomeado ou None se não existir tal parâmetro.
`p.setParameterStorage(parameterName, storage)`	Não aplicável	Configura o armazenamento do parâmetro nomeado.

Tabela 21. Funções definidas pelo objeto ParameterProvider (continuação)		
Método	**Tipo de retorno**	**Descrição**
`p.getParameterType(parameterName)`	ParameterType	Retorna o tipo do parâmetro nomeado ou None se não existir tal parâmetro.
`p.setParameterType(parameterName, type)`	Não aplicável	Define o tipo do parâmetro nomeado.
`p.getParameterValue(parameterName)`	Objeto	Retorna o valor do parâmetro nomeado ou None se não existir tal parâmetro.
`p.setParameterValue(parameterName, value)`	Não aplicável	Configura o valor do parâmetro nomeado.

No exemplo a seguir, o script agrega alguns dados do Telco para localizar qual região possui os dados da receita média mais baixa. Um parâmetro de fluxo é então configurado com esta região. Em seguida, esse parâmetro de fluxo é utilizado em um nó Selecionar para excluir essa região dos dados antes que um modelo de rotatividade seja construído no restante.

O exemplo é artificial porque o script gera o nó Selecionar nó em si e, portanto, pode ter gerado o valor correto diretamente na expressão do nó Selecionar. No entanto, como os fluxos são normalmente pré-construídos, configurar os parâmetros dessa forma fornece um exemplo útil.

A primeira parte do script de exemplo cria o parâmetro de fluxo que conterá a região com a receita média mais baixa. O script também cria os nós na ramificação de agregação e na ramificação de construção de modelo e os conecta entre si.

```
import modeler.api

stream = modeler.script.stream()

# Initialize a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)

aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)
```

O script de exemplo cria o fluxo a seguir.

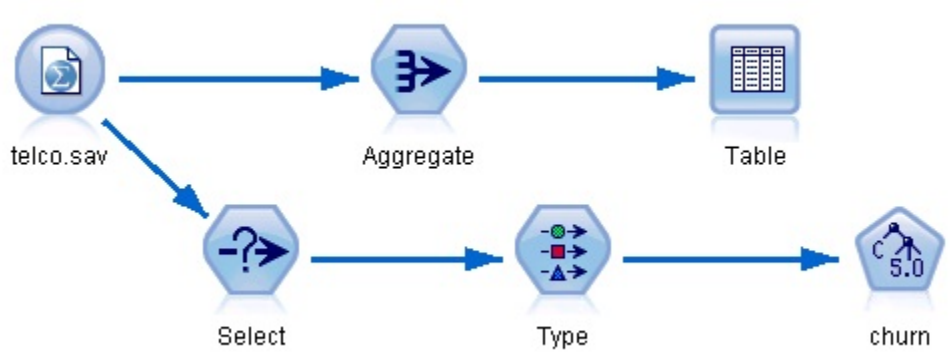

Figura 5. Fluxo resultante do script de exemplo

A parte a seguir do script de exemplo executa o nó Tabela no término da ramificação de agregação.

```
# First execute the table node
results = []
tablenode.run(results)
```

A parte a seguir do script de exemplo acessa a saída da tabela que foi gerada pela execução do nó Tabela. O script então itera através das linhas na tabela, procurando a região com a receita média mais baixa.

```
# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1
```

A parte a seguir do script utiliza a região com a receita média mais baixa para configurar o parâmetro de fluxo "LowestRegion" que foi criado anteriormente. O script então executa o construtor de modelo com a região especificada excluída dos dados de treinamento.

```
# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

O script de exemplo completo é mostrado abaixo.

```
import modeler.api

stream = modeler.script.stream()

# Create a stream parameter
stream.setParameterStorage("LowestRegion", modeler.api.ParameterStorage.INTEGER)

# First create the aggregation branch to compute the average income per region
statisticsimportnode = stream.createAt("statisticsimport", "SPSS File", 114, 142)
statisticsimportnode.setPropertyValue("full_filename", "$CLEO_DEMOS/telco.sav")
statisticsimportnode.setPropertyValue("use_field_format_for_storage", True)
```

```
aggregatenode = modeler.script.stream().createAt("aggregate", "Aggregate", 294, 142)
aggregatenode.setPropertyValue("keys", ["region"])
aggregatenode.setKeyedPropertyValue("aggregates", "income", ["Mean"])

tablenode = modeler.script.stream().createAt("table", "Table", 462, 142)

stream.link(statisticsimportnode, aggregatenode)
stream.link(aggregatenode, tablenode)

selectnode = stream.createAt("select", "Select", 210, 232)
selectnode.setPropertyValue("mode", "Discard")
# Reference the stream parameter in the selection
selectnode.setPropertyValue("condition", "'region' = '$P-LowestRegion'")

typenode = stream.createAt("type", "Type", 366, 232)
typenode.setKeyedPropertyValue("direction", "churn", "Target")

c50node = stream.createAt("c50", "C5.0", 534, 232)

stream.link(statisticsimportnode, selectnode)
stream.link(selectnode, typenode)
stream.link(typenode, c50node)


# First execute the table node
results = []
tablenode.run(results)

# Running the table node should produce a single table as output
table = results[0]

# table output contains a RowSet so we can access values as rows and columns
rowset = table.getRowSet()
min_income = 1000000.0
min_region = None

# From the way the aggregate node is defined, the first column
# contains the region and the second contains the average income
row = 0
rowcount = rowset.getRowCount()
while row < rowcount:
    if rowset.getValueAt(row, 1) < min_income:
        min_income = rowset.getValueAt(row, 1)
        min_region = rowset.getValueAt(row, 0)
    row += 1

# Check that a value was assigned
if min_region != None:
    stream.setParameterValue("LowestRegion", min_region)
else:
    stream.setParameterValue("LowestRegion", -1)

# Finally run the model builder with the selection criteria
c50node.run([])
```

Valores Globais

Os valores globais são utilizados para calcular várias estatísticas de resumo para campos especificados. Esses valores de resumo podem ser acessados em qualquer lugar dentro do fluxo. Os valores globais são semelhantes aos parâmetros de fluxo por eles serem acessados por nome através do fluxo. Eles diferem dos parâmetros de fluxo pelo fato de os valores associados serem atualizados automaticamente quando um nó Configurar Globais é executado, ao invés de serem designados pelo script ou a partir da linha de comandos. Os valores globais para um fluxo são acessados ao chamar o método `getGlobalValues()` do fluxo.

O objeto GlobalValues define as funções que são mostradas na tabela a seguir.

Tabela 22. Funções que são definidas pelo objeto GlobalValues		
Método	**Tipo de retorno**	**Descrição**
`g.fieldNameIterator()`	Agente iterativo	Retorna um agente iterativo para cada nome de campo com pelo menos um valor global.
`g.getValue(type, fieldName)`	Objeto	Retorna o valor global para o tipo e o nome de campo especificado ou None se nenhum valor puder ser localizado. Geralmente espera-se que o valor retornado seja um número, embora uma funcionalidade futura possa retornar diferentes tipos de valores.
`g.getValues(fieldName)`	Mapa	Retorna um mapa que contém as entradas conhecidas para o nome de campo especificado ou None se não houver entradas existentes para o campo.

`GlobalValues.Type` define o tipo de estatísticas básicas que estão disponíveis. As estatísticas de resumo a seguir estão disponíveis:

- `MAX`: o valor máximo do campo.
- `MEAN`: o valor médio do campo.
- `MIN`: o valor mínimo do campo.
- `STDDEV`: o desvio padrão do campo.
- `SUM`: a soma dos valores no campo.

Por exemplo, o script a seguir acessa o valor médio do campo "income", que é calculado por um nó Configurar Globais:

```
import modeler.api

globals = modeler.script.stream().getGlobalValues()
mean_income = globals.getValue(modeler.api.GlobalValues.Type.MEAN, "income")
```

Trabalhando com Diversos Fluxos: Scripts Independentes

Para trabalhar com diversos fluxos, um script independente deve ser utilizado. O script independente pode ser editado e executado dentro da IU do IBM SPSS Modeler ou transmitido como um parâmetro da linha de comandos no modo em lote.

O script independente a seguir abre dois fluxos. Um destes fluxos constrói um modelo, ao passo que o segundo fluxo representa a distribuição dos valores previstos.

```
# Change to the appropriate location for your system
demosDir = "C:/Program Files/IBM/SPSS/Modeler/18.5.0/DEMOS/streams/"

session = modeler.script.session()
tasks = session.getTaskRunner()

# Open the model build stream, locate the C5.0 node and run it
buildstream = tasks.openStreamFromFile(demosDir + "druglearn.str", True)
c50node = buildstream.findByType("c50", None)
results = []
c50node.run(results)

# Now open the plot stream, find the Na_to_K derive and the histogram
plotstream = tasks.openStreamFromFile(demosDir + "drugplot.str", True)
```

```
derivenode = plotstream.findByType("derive", None)
histogramnode = plotstream.findByType("histogram", None)

# Create a model applier node, insert it between the derive and histogram nodes
# then run the histgram
applyc50 = plotstream.createModelApplier(results[0], results[0].getName())
applyc50.setPositionBetween(derivenode, histogramnode)
plotstream.linkBetween(applyc50, derivenode, histogramnode)
histogramnode.setPropertyValue("color_field", "$C-Drug")
histogramnode.run([])

# Finally, tidy up the streams
buildstream.close()
plotstream.close()
```

O exemplo a seguir mostra como também é possível iterar sobre os fluxos abertos (todos os fluxos abertos na guia Fluxos). Observe que isso é suportado apenas em scripts independentes

```
for stream in modeler.script.streams():
    print stream.getName()
```

Capítulo 5. Dicas de script

Esta seção fornece uma visão geral de dicas e técnicas para utilizar scripts, incluindo modificação da execução de fluxo, uso de uma senha codificada em um script e acesso aos objetos no IBM SPSS Repositório do Collaboration and Deployment Services.

Modificando a Execução de Fluxo

Quando um fluxo é executado, seus nós de terminal são executados em uma ordem otimizada para a situação padrão. Em alguns casos, você pode preferir uma ordem de execução diferente. Para modificar a ordem de execução de um fluxo, conclua as seguintes etapas a partir da guia Execução da caixa de diálogo de propriedades do fluxo:

1. Comece com um script vazio.
2. Clique no botão **Anexar script padrão** na barra de ferramentas para incluir o script de fluxo padrão.
3. Altere a ordem das instruções no script de fluxo padrão para a ordem em que deseja que as instruções sejam executadas.

Executando loop pelos nós

É possível utilizar um loop `for` para executar loop em todos os nós em um fluxo. Por exemplo, os dois exemplos de script a seguir executam loop em todos os nós e alteram os nomes de campo em quaisquer nós Filtro para letras maiúsculas.

Este script poderá ser utilizado em qualquer fluxo que possuir um nó Filtro, mesmo se nenhum campo estiver realmente filtrado. Basta incluir um nó Filtro que transmita todos os campos para alterar os nomes de campo para letras maiúsculas em todo o quadro.

```
# Alternative 1: using the data model nameIterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # nameIterator() returns the field names
        for field in node.getInputDataModel().nameIterator():
            newname = field.upper()
            node.setKeyedPropertyValue("new_name", field, newname)
```

```
# Alternative 2: using the data model iterator() function
stream = modeler.script.stream()
for node in stream.iterator():
    if (node.getTypeName() == "filter"):
        # iterator() returns the field objects so we need
        # to call getColumnName() to get the name
        for field in node.getInputDataModel().iterator():
            newname = field.getColumnName().upper()
            node.setKeyedPropertyValue("new_name", field.getColumnName(), newname)
```

O script executa loop em todos os nós no fluxo atual e verifica se cada nó é um Filtro. Se sim, o script faz loop por cada campo no nó e usa a função `field.upper()` ou `field.getColumnName().upper()` para mudar o nome para maiúsculas.

Acessando Objetos no IBM SPSS Repositório do Collaboration and Deployment Services

Se você tiver uma licença para o IBM SPSS Repositório do Collaboration and Deployment Services, será possível armazenar e recuperar objetos do repositório usando comandos de script. Use o repositório para gerenciar o ciclo de vida de modelos de mineração de dados e objetos preditivos relacionados no contexto de aplicativos, ferramentas e soluções corporativas

Conectando-se ao IBM SPSS Repositório do Collaboration and Deployment Services

Para acessar o repositório, deve-se primeiro configurar uma conexão válida com ele, seja por meio do menu **Ferramentas** da interface com o usuário do SPSS Modeler ou por meio da linha de comandos Para obter informações adicionais, consulte "Argumentos de Conexão do IBM SPSS Repositório do Collaboration and Deployment Services" na página 70.

Obtendo acesso ao repositório

O repositório pode ser acessado a partir da sessão, por exemplo:

```
repo = modeler.script.session().getRepository()
```

Recuperando objetos do repositório

Em um script, use as funções `retrieve*` para acessar vários objetos, incluindo fluxos, modelos, saída e nós. Um resumo das funções de recuperação é mostrado na tabela a seguir.

Tabela 23. Recuperar funções de script

Tipo de Objeto	Função de Repositório
Fluxo	repo.retrieveStream(Caminho de sequência, Versão de sequência, Etiqueta de sequência, Booleano autoManage).
Modelar	repo.retrieveModel(Caminho de sequência, Versão de sequência, Rótulo de sequência, Booleano autoManage)
Saída	repo.retrieveDocument(Caminho de sequência, versão de sequência, rótulo de sequência, Booleano autoManage)
Nó	repo.retrieveProcessor(caminho de sequência, versão de sequência, rótulo de sequência, diagrama ProcessorDiagram)

Por exemplo, é possível recuperar um fluxo do repositório com a seguinte função:

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", None, "production", True)
```

Este exemplo recupera o fluxo `risk_score.str` da pasta especificada. O rótulo `production` identifica qual versão do fluxo recuperar e o último parâmetro especifica que SPSS Modeler deve gerenciar o fluxo (por exemplo, para que o fluxo apareça na guia **Fluxos** se a interface com o usuário SPSS Modeler estiver visível). Como alternativa, para usar uma versão específica não rotulada:

```
stream = repo.retrieveStream("/projects/retention/risk_score.str", "0:2015-10-12 14:15:41.281",
None, True)
```

Nota: Se os parâmetros de versão e rótulo forem `None`, a versão mais recente será retornada.

Armazenando objetos no repositório

Para usar o script para armazenar objetos no repositório, use as funções `store*` Um resumo das funções de loja é mostrado na tabela a seguir.

Tabela 24. Armazenar funções de script

Tipo de Objeto	Função de Repositório
Fluxo	repo.storeStream(fluxoProcessorStream , caminho da sequência, rótulo da sequência)
Modelar	repo.storeModel(ModelOutput modelOutput, caminho da sequência, rótulo da cadeia)

Tabela 24. Armazenar funções de script (continuação)	
Tipo de Objeto	**Função de Repositório**
Saída	repo.storeDocument(DocumentOutput documentOutput, Caminho da sequência, Rótulo da sequência
Nó	repo.storeProcessor(nó de Processador, caminho de Sequência, rótulo de Sequência)

Por exemplo, é possível armazenar uma nova versão do fluxo `risk_score.str` com a função a seguir:

```
versionId = repo.storeStream(stream, "/projects/retention/risk_score.str", "test")
```

Este exemplo armazena uma nova versão do fluxo, associa o rótulo `"test"` a ele, e retorna o marcador de versão para a versão recém-criada

Nota: Se não desejar associar um rótulo à nova versão, passe None para o rótulo.

Gerenciando Pastas de Repositório

Usando pastas no repositório, é possível organizar objetos em grupos lógicos e facilitar a visualização de quais objetos estão relacionados. Crie pastas usando a função `createFolder()` , como no exemplo a seguir:

```
newpath = repo.createFolder("/projects", "cross-sell")
```

Este exemplo cria uma nova pasta chamada `"cross-sell"` na pasta `"/projects"` . A função retorna o caminho completo para a nova Pasta.

Para renomear uma pasta, use a função `renameFolder()` :

```
repo.renameFolder("/projects/cross-sell", "cross-sell-Q1")
```

O primeiro parâmetro é o caminho completo para a pasta a ser renomeada e o segundo é o novo nome para fornecer essa pasta.

Para excluir uma pasta vazia, use a função `deleteFolder()` :

```
repo.deleteFolder("/projects/cross-sell")
```

Bloqueando e desbloqueando objetos

A partir de um script, é possível bloquear um objeto para evitar que outros usuários atualizem qualquer uma de suas versões existentes ou criem novas versões. Também é possível desbloquear um objeto que você bloqueou.

A sintaxe para bloquear e desbloquear um objeto é:

```
repo.lockFile(REPOSITORY_PATH)
repo.lockFile(URI)

repo.unlockFile(REPOSITORY_PATH)
repo.unlockFile(URI)
```

Assim como ocorre com armazenamento e recuperação de objetos, o REPOSITORY_PATH fornece o local do objeto no repositório. Deve-se colocar o caminho entre aspas e utilizar barras como delimitadores. Ele não faz distinção entre maiúsculas e minúsculas.

```
repo.lockFile("/myfolder/Stream1.str")
repo.unlockFile("/myfolder/Stream1.str")
```

Como alternativa, é possível utilizar um Identificador Uniforme de Recursos (URI) ao invés de um caminho de repositório para fornecer o local do objeto. O URI deve incluir o prefixo `spsscr:` e ser totalmente colocado entre aspas. Apenas barras são permitidas como delimitadores de caminho e os espaços devem ser codificados. Ou seja, utilize `%20` ao invés de um espaço no caminho. O URI não faz distinção entre maiúsculas e minúsculas. A seguir estão alguns exemplos:

```
repo.lockFile("spsscr:///myfolder/Stream1.str")
repo.unlockFile("spsscr:///myfolder/Stream1.str")
```

Note que o bloqueio do objeto é aplicado a todas as versões de um objeto - não é possível bloquear ou desbloquear versões individuais.

Gerando uma Senha Codificada

Em determinados casos, poderá ser necessário incluir uma senha em um script; por exemplo, você pode querer acessar uma origem de dados protegida por senha. As senhas codificadas podem ser utilizadas em:

- Propriedades do Nó para nós de Origem e de Saída do Banco de Dados
- Argumentos da linha de comandos para efetuar login no servidor
- Propriedades de conexão com o banco de dados armazenadas em um arquivo *.par* (o arquivo de parâmetro gerado na guia Publicação de um nó de exportação)

Através da interface com o usuário, uma ferramenta está disponível para gerar senhas codificadas com base no algoritmo Blowfish (consulte http://www.schneier.com/blowfish.html para obter mais informações). Depois codificado, é possível copiar e armazenar a senha para os arquivos de script e argumentos de linha de comandos. O nó de propriedade `epassword` usado para `databasenode` e `databaseexportnode` armazena a senha codificada.

1. Para gerar uma senha codificada, no menu Ferramentas, escolha:

   Codificar Senha...
2. Especifique uma senha na caixa de texto Senha.
3. Clique em **Codificar** para gerar uma codificação aleatória de sua senha.
4. Clique no botão Copiar para copiar a senha codificada para a Área de Transferência.
5. Cole a senha no script ou parâmetro desejado.

Verificação de Script

É possível verificar rapidamente a sintaxe de todos os tipos de scripts clicando no botão de verificação vermelho na barra de ferramentas da caixa de diálogo Script Independente.

Figura 6. Ícones da barra de ferramentas de script do fluxo

A verificação de script alerta para quaisquer erros em seu código e faz recomendações para melhoria. Para visualizar a linha com erros, clique no feedback na metade inferior da caixa de diálogo. Isso destacará o erro em vermelho.

Script a partir da Linha de Comandos

O script permite executar operações que geralmente são executadas na interface com o usuário. Basta especificar e executar um script independente na linha de comandos ao ativar o IBM SPSS Modeler Por exemplo:

```
client -script scores.txt -execute
```

O sinalizador `-script` carrega o script especificado, ao passo que o sinalizador `-execute` executa todos os comandos no arquivo de script.

Compatibilidade com Liberações Anteriores

Os scripts criados em liberações anteriores do IBM SPSS Modeler geralmente devem funcionar inalterados na liberação atual. No entanto, os nuggets do modelo agora podem ser inseridos no fluxo automaticamente (essa é a configuração padrão) e podem substituir ou complementar um nugget existente desse tipo no fluxo. Se isso realmente acontece depende das configurações das opções **Incluir modelo para fluxo** e **Substituir modelo anterior** (**Ferramentas> Opções> Opções do usuário> Notificações**). Poderá ser necessário, por exemplo, modificar um script a partir de uma liberação anterior em que uma substituição de nugget é manipulada ao excluir o nugget existente e inserir o novo nugget.

Os scripts criados na liberação atual podem não funcionar em liberações anteriores.

Se um script criado em uma liberação anterior utilizar um comando que foi substituído (ou descontinuado), o formato antigo ainda será suportado, mas uma mensagem de aviso será exibida. Por exemplo, a palavra-chave `generated` antiga foi substituída por `model` e o comando `clear generated` foi substituído por `clear generated palette`. Os scripts que usam os formatos antigos ainda serão executados, mas um aviso será exibido.

acessando resultados da execução de fluxo

Muitos nós do IBM SPSS Modeler produzem objetos de saída como modelos, gráficos e dados tabulares. Muitas dessas saídas contêm valores úteis que podem ser usados por scripts para guiar execuções subsequentes. Esses valores são agrupados em contêineres de conteúdo (referidos simplesmente como contêineres) que podem ser acessados utilizando tags ou IDs que identificam cada contêiner. A maneira como esses valores são acessados depende do formato ou do "modelo de conteúdo" utilizado por esse contêiner.

Por exemplo, muitas saídas de modelo preditivo utilizam uma variante do XML chamada PMML para representar informações sobre o modelo, como quais campos uma árvore de decisão utiliza em cada divisão ou como os neurônios em uma rede neural são conectados e com que intensidade. As saídas de modelo que utilizam o PMML fornecem um Modelo de Conteúdo XML que pode ser utilizado para acessar essas informações. Por exemplo:

```
stream = modeler.script.stream()
# Assume the stream contains a single C5.0 model builder node
# and that the datasource, predictors and targets have already been
# set up
modelbuilder = stream.findByType("c50", None)
results = []
modelbuilder.run(results)
modeloutput = results[0]

# check how many contents are there and what are their names
tags = modeloutput.getContentModelTags()
print "Content Model Tags :" , tags

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")
if (cm != None) :
    # The PMML content model is a generic XML-based content model that
    # uses XPath syntax. Use that to find the names of the data fields.
    # The call returns a list of strings match the XPath values
    dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField",
"name")
    print "Data Field Names:" , dataFieldNames
```

O IBM SPSS Modeler suporta os modelos de conteúdo a seguir no script:

- O **Modelo de conteúdo de tabela** fornece acesso aos dados tabulares simples representados como linhas e colunas.
- O **Modelo de conteúdo XML** fornece acesso ao conteúdo armazenado no formato XML.
- O **Modelo de conteúdo JSON** fornece acesso ao conteúdo armazenado no formato JSON.
- O **Modelo de conteúdo de estatísticas** fornece acesso a estatísticas de resumo sobre um campo específico.
- O **Modelo de conteúdo de estatísticas de coluna pairwise** fornece acesso a estatísticas de resumo entre dois campos ou valores entre dois campos separados.

Observe que os nós a seguir não contêm esses modelos de conteúdo:

- Séries temporais
- Discriminante
- SLRM
- TCM
- Todos os nós Python
- Todos os nós Spark
- Todos os nós de Modelagem de banco de dados
- Modelo de Extensão
- STP

Modelo de conteúdo de tabela

O modelo de conteúdo da tabela fornece um modelo simples para acessar dados de linha e da coluna simples. Todos os valores em uma coluna específica devem ter o mesmo tipo de armazenamento (por exemplo, sequências ou números inteiros).

API do

Tabela 25. API do

Voltar	Método	Descrição
`int`	`getRowCount()`	Retorna o número de linhas nesta tabela.
`int`	`getColumnCount()`	Retorna o número de colunas nesta tabela.
`String`	`getColumnName(int columnIndex)`	Retorna o nome da coluna no índice da coluna especificado. O índice da coluna inicia em 0.
`StorageType`	`getStorageType(int columnIndex)`	Retorna o tipo de armazenamento da coluna no índice especificado. O índice da coluna inicia em 0.
`Object`	`getValueAt(int rowIndex, int columnIndex)`	Retorna o valor no índice de linha e de coluna especificado. Os índices de linha e de coluna iniciam em 0.
`void`	`reset()`	Limpa qualquer armazenamento interno associado a este modelo de conteúdo.

Nós e saídas

Esta tabela lista os nós que constroem saídas que incluem esse tipo de modelo de conteúdo.

Tabela 26. Nós e saídas

Nome do nó	Nome de saída	ID do contêiner
table	table	"table"

Script de exemplo

```
stream = modeler.script.stream()
from modeler.api import StorageType

# Set up the variable file import node
varfilenode = stream.createAt("variablefile", "DRUG Data", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")

# Next create the aggregate node and connect it to the variable file node
aggregatenode = stream.createAt("aggregate", "Aggregate", 192, 96)
stream.link(varfilenode, aggregatenode)

# Configure the aggregate node
aggregatenode.setPropertyValue("keys", ["Drug"])
aggregatenode.setKeyedPropertyValue("aggregates", "Age", ["Min", "Max"])
aggregatenode.setKeyedPropertyValue("aggregates", "Na", ["Mean", "SDev"])

# Then create the table output node and connect it to the aggregate node
tablenode = stream.createAt("table", "Table", 288, 96)
stream.link(aggregatenode, tablenode)

# Execute the table node and capture the resulting table output object
results = []
tablenode.run(results)
tableoutput = results[0]

# Access the table output's content model
tablecontent = tableoutput.getContentModel("table")

# For each column, print column name, type and the first row
# of values from the table content
col = 0
while col < tablecontent.getColumnCount():
     print tablecontent.getColumnName(col), \
     tablecontent.getStorageType(col), \
     tablecontent.getValueAt(0, col)
     col = col + 1
```

A saída na guia Depuração de script será semelhante a esta:

```
Age_Min Integer 15
Age_Max Integer 74
Na_Mean Real 0.730851098901
Na_SDev Real 0.116669731242
Drug String drugY
Record_Count Integer 91
```

Modelo de Conteúdo XML

O Modelo de Conteúdo XML fornece acesso ao conteúdo baseado em XML.

O Modelo de Conteúdo XML suporta a capacidade de acessar componentes com base em expressões XPath. As expressões XPath são sequências que definem quais elementos ou atributos são exigidos pelo responsável pela chamada. O Modelo de Conteúdo XML oculta os detalhes de vários objetos de construção e expressões de compilação que geralmente são necessários pelo suporte ao XPath. Isso simplifica a chamada do script Python.

O Modelo de Conteúdo XML inclui uma função que retorna o documento XML como uma sequência. Isso permite que usuários do script Python utilizem sua biblioteca Python preferencial para analisar o XML.

API do

Tabela 27. API do

Voltar	Método	Descrição
`String`	`getXMLAsString()`	Retorna o XML como uma sequência.
`number`	`getNumericValue(String xpath)`	Retorna o resultado da avaliação do caminho com tipo de retorno numérico (por exemplo, contagem do número de elementos que correspondem à expressão de caminho).
`boolean`	`getBooleanValue(String xpath)`	Retorna o resultado booleano da avaliação da expressão de caminho especificada.
`String`	`getStringValue(String xpath, String attribute)`	Retorna o valor de atributo ou o valor do nó XML que corresponde ao caminho especificado.
`List of strings`	`getStringValues(String xpath, String attribute)`	Retorna uma lista de todos os valores de atributos ou valores do nó XML que correspondem ao caminho especificado.
`List of lists of strings`	`getValuesList(String xpath, <List of strings> attributes, boolean includeValue)`	Retorna uma lista de todos os valores de atributos que correspondem ao caminho especificado junto com o valor do nó XML, se necessário.
`Hash table (key:string, value:list of string)`	`getValuesMap(String xpath, String keyAttribute, <List of strings> attributes, boolean includeValue)`	Retorna uma hashtable que utiliza o atributo-chave ou o valor do nó XML como chave, e a lista de valores de atributos especificados como valores da tabela.
`boolean`	`isNamespaceAware()`	Retorna se os analisadores XML devem reconhecer namespaces. O padrão é `False`.
`void`	`setNamespaceAware(boolean value)`	Configura se os analisadores XML devem reconhecer namespaces. Isso também chama `reset()` para garantir que as mudanças sejam captadas por chamadas subsequentes.

Tabela 27. API do (continuação)		
Voltar	**Método**	**Descrição**
`void`	`reset()`	Esvazia qualquer armazenamento interno associado a este modelo de conteúdo (por exemplo, um objeto DOM em cache).

Nós e saídas

Esta tabela lista os nós que constroem saídas que incluem esse tipo de modelo de conteúdo.

Tabela 28. Nós e saídas		
Nome do nó	**Nome de saída**	**ID do contêiner**
`Most model builders`	`Most generated models`	`"PMML"`
`"autodataprep"`	`n/a`	`"PMML"`

Script de exemplo

O código de script Python para acessar o conteúdo pode ser semelhante a este:

```
results = []
modelbuilder.run(results)
modeloutput = results[0]
cm = modeloutput.getContentModel("PMML")

dataFieldNames = cm.getStringValues("/PMML/DataDictionary/DataField", "name")
predictedNames = cm.getStringValues("//MiningSchema/
MiningField[@usageType='predicted']", "name")
```

Modelo de Conteúdo JSON

O Modelo de Conteúdo JSON é utilizado para fornecer suporte para o conteúdo do formato JSON. Isso fornece uma API básica para permitir que os responsáveis pela chamada extraiam valores supondo que eles saibam quais valores devem ser acessados.

API do

Tabela 29. API do		
Voltar	**Método**	**Descrição**
`String`	`getJSONAsString()`	Retorna o conteúdo JSON como uma sequência.
`Object`	`getObjectAt(<List of cbjecta> path, JSONArtifact artifact) throws Exception`	Retorna o objeto no caminho especificado. O artefato raiz fornecido pode ser nulo no caso em que a raiz do conteúdo é utilizada. O valor retornado pode ser uma sequência de caracteres literal, um número inteiro, real ou booleano ou um artefato JSON (um objeto JSON ou uma matriz JSON).

Tabela 29. API do (continuação)		
Voltar	**Método**	**Descrição**
`Hash table (key:object, value:object>`	`getChildValuesAt(<List of object> path, JSONArtifact artifact) throws Exception`	Retorna os valores-filhos do caminho especificado se o caminho levar a um objeto JSON, caso contrário, retorna nulo. As chaves na tabela são sequências, ao passo que o valor associado pode ser uma sequência de caracteres literal, um número inteiro, real ou booleano ou um artefato JSON (um objeto JSON ou uma matriz JSON).
`List of objects`	`getChildrenAt(<List of object> path path, JSONArtifact artifact) throws Exception`	Retorna a lista de objetos no caminho especificado se o caminho levar a uma matriz JSON, caso contrário, retorna nulo. Os valores retornados podem ser uma sequência de caracteres literal, um número inteiro, real ou booleano ou um artefato JSON (um objeto JSON ou uma matriz JSON).
`void`	`reset()`	Esvazia qualquer armazenamento interno associado a este modelo de conteúdo (por exemplo, um objeto DOM em cache).

Nós e saídas

Esta tabela lista os nós que constroem saídas que incluem esse tipo de modelo de conteúdo.

Tabela 30. Nós e saídas		
Nome do nó	**Nome de saída**	**ID do contêiner**
`"applykmeansas"`	`n/a`	`"JSON_MV"`
`"applyxgboosttree"`	`n/a`	`"JSON_MV"`

Scripts de exemplo

Os seguintes scripts recuperam arquivos JSON:

```
applykmeansas = stream.findByType("applykmeansas",None)
mvjson = applykmeansas.getContentModel("JSON_MV")
print(mvjson.getJSONAsString())
```

```
applyxgboosttree = stream.findByType("applyxgboosttree",None)
mvjson = applyxgboosttree.getContentModel("JSON_MV")
print(mvjson.getJSONAsString())
```

Modelo de conteúdo de estatísticas de coluna e modelo de conteúdo de estatísticas pairwise

O modelo de conteúdo de estatísticas de coluna fornece acesso às estatísticas que podem ser calculadas para cada campo (estatísticas univariadas). O modelo de conteúdo de estatísticas de pares fornece acesso às estatísticas que podem ser calculadas entre pares de campos ou de valores em um campo.

As medidas de estatísticas possíveis são:

- `Count`
- `UniqueCount`
- `ValidCount`
- `Mean`
- `Sum`
- `Min`
- `Max`
- `Range`
- `Variance`
- `StandardDeviation`
- `StandardErrorOfMean`
- `Skewness`
- `SkewnessStandardError`
- `Kurtosis`
- `KurtosisStandardError`
- `Median`
- `Mode`
- `Pearson`
- `Covariance`
- `TTest`
- `FTest`

Alguns valores são apropriados apenas a partir de estatísticas de coluna única, ao passo que outros são apropriados apenas para estatísticas de pares.

Os nós que produzirão esses são:

- O **Nó de estatísticas** produz estatísticas de coluna e pode produzir estatísticas de pares quando os campos de correlação são especificados
- O **Nó Auditoria de Dados** produz estatísticas de coluna e pode produzir estatísticas de pares quando um campo de sobreposição é especificado.
- O **Nó Médias** produz estatísticas de pares quando compara pares de campos ou compara os valores de um campo com outros resumos de campo.

As capacidades de um nó específico e também as configurações no nó determinarão quais modelos e estatísticas de conteúdo estarão disponíveis.

API de ColumnStatsContentModel

Tabela 31. API de ColumnStatsContentModel

Voltar	Método	Descrição
`List<StatisticType>`	`getAvailableStatistics()`	Retorna as estatísticas disponíveis nesse modelo. Nem todos os campos terão necessariamente valores para todas as estatísticas.
`List<String>`	`getAvailableColumns()`	Retorna os nomes das colunas para as quais as estatísticas foram calculadas.
`Number`	`getStatistic(String column, StatisticType statistic)`	Retorna os valores estatísticos associados à coluna.
`void`	`reset()`	Limpa qualquer armazenamento interno associado a este modelo de conteúdo.

API PairwiseStatsContentModel

Tabela 32. API PairwiseStatsContentModel

Voltar	Método	Descrição
`List<StatisticType>`	`getAvailableStatistics()`	Retorna as estatísticas disponíveis nesse modelo. Nem todos os campos terão necessariamente valores para todas as estatísticas.
`List<String>`	`getAvailablePrimaryColumns()`	Retorna os nomes de colunas primárias para as quais as estatísticas foram calculadas.
`List<Object>`	`getAvailablePrimaryValues()`	Retorna os valores de colunas primárias para as quais as estatísticas foram calculadas.
`List<String>`	`getAvailableSecondaryColumns()`	Retorna os nomes de colunas secundárias para as quais as estatísticas foram calculadas.
`Number`	`getStatistic(String primaryColumn, String secondaryColumn, StatisticType statistic)`	Retorna os valores estatísticos associados às colunas.
`Number`	`getStatistic(String primaryColumn, Object primaryValue, String secondaryColumn, StatisticType statistic)`	Retorna os valores de estatística associados ao valor da coluna primária e da coluna secundária.
`void`	`reset()`	Limpa qualquer armazenamento interno associado a este modelo de conteúdo.

Nós e saídas

Esta tabela lista os nós que constroem saídas que incluem esse tipo de modelo de conteúdo.

Tabela 33. Nós e saídas

Nome do nó	Nome de saída	ID do contêiner	Notes
"means" (nó Médias)	"means"	"columnStatistics"	
"means" (nó Médias)	"means"	"pairwiseStatistics"	
"dataaudit" (nó Auditoria de Dados)	"means"	"columnStatistics"	
"statistics" (nó Estatísticas)	"statistics"	"columnStatistics"	Gerado apenas quando campos específicos são examinados.
"statistics" (nó Estatísticas)	"statistics"	"pairwiseStatistics"	Gerado apenas quando campos específicos são correlacionados.

Script de exemplo

```
from modeler.api import StatisticType
stream = modeler.script.stream()

# Set up the input data
varfile = stream.createAt("variablefile", "File", 96, 96)
varfile.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")

# Now create the statistics node. This can produce both
# column statistics and pairwise statistics
statisticsnode = stream.createAt("statistics", "Stats", 192, 96)
statisticsnode.setPropertyValue("examine", ["Age", "Na", "K"])
statisticsnode.setPropertyValue("correlate", ["Age", "Na", "K"])
stream.link(varfile, statisticsnode)

results = []
statisticsnode.run(results)
statsoutput = results[0]
statscm = statsoutput.getContentModel("columnStatistics")
if (statscm != None):
    cols = statscm.getAvailableColumns()
    stats = statscm.getAvailableStatistics()
    print "Column stats:", cols[0], str(stats[0]), " = ",
statscm.getStatistic(cols[0], stats[0])

statscm = statsoutput.getContentModel("pairwiseStatistics")
if (statscm != None):
    pcols = statscm.getAvailablePrimaryColumns()
    scols = statscm.getAvailableSecondaryColumns()
    stats = statscm.getAvailableStatistics()
    corr = statscm.getStatistic(pcols[0], scols[0], StatisticType.Pearson)
```

```
    print "Pairwise stats:", pcols[0], scols[0], " Pearson = ", corr
```

Capítulo 6. Argumentos de Linha de Comandos

Chamando o Software

É possível usar a linha de comandos de seu sistema operacional para iniciar IBM SPSS Modeler da seguinte forma.

Microsoft Windows

1. Em um computador no qual o IBM SPSS Modeler está instalado, abra um DOS, um prompt de comandos ou uma janela.
2. Alterne para o caminho da instalação para IBM SPSS Modeler (por exemplo, `[Installpath]\Program Files\IBM\SPSS\Modeler\18.5\bin`)
3. Para iniciar a interface IBM SPSS Modeler no modo interativo, digite o comando `modelerclient` seguido pelos argumentos necessários; por exemplo:

```
modelerclient -stream report.str -execute
```

É possível usar os argumentos disponíveis (sinalizadores) para se conectar a um servidor, carregar fluxos, executar scripts ou especificar outros parâmetros conforme necessário.

SO Mac

1. Localize o caminho de comando do Mac OS para IBM SPSS Modeler (por exemplo, `[Installpath]/Applications/IBM/SPSS/Modeler/18.5/IBM SPSS Modeler.app/Contents/MacOS`).
2. Para iniciar a interface IBM SPSS Modeler no modo interativo, execute o comando modeler seguido pelos argumentos necessários; por exemplo:

```
./modeler -stream report.str -execute
```

Utilizando Argumentos de Linha de Comandos

É possível anexar argumentos de linha de comandos (também referidos como *sinalizadores*) ao comando inicial `modelerclient` para alterar a chamada do IBM SPSS Modeler.

Vários tipos de argumentos de linha de comandos estão disponíveis e são descritos posteriormente nesta seção.

Tabela 34. Tipos de argumentos de linha de comandos

Tipo de argumento	**Onde descrito**
Argumentos do sistema	Consulte o tópico “Argumentos do sistema” na página 66 para obter mais informações.
Argumentos de parâmetro	Consulte o tópico “Argumentos de parâmetro” na página 68 para obter mais informações.
Argumentos de conexão do servidor	Consulte o tópico “Argumentos de conexão do servidor” na página 69 para obter mais informações.

Tabela 34. Tipos de argumentos de linha de comandos (continuação)	
Tipo de argumento	**Onde descrito**
IBM SPSS Repositório do Collaboration and Deployment Services argumentos de conexão	Consulte o tópico "Argumentos de Conexão do IBM SPSS Repositório do Collaboration and Deployment Services" na página 70 para obter informações adicionais.
IBM SPSS Analytic Server argumentos de conexão	Consulte o tópico "IBM SPSS Analytic Server argumentos de conexão" na página 71 para obter informações adicionais.

Por exemplo, é possível utilizar os sinalizadores `-server`, `-stream` e `-execute` para se conectar a um servidor e, em seguida, carregar e executar um fluxo, conforme a seguir:

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Observe que ao executar com relação a uma instalação do cliente local, os argumentos de conexão do servidor não são necessários.

Os valores de parâmetros que contiverem espaços podem ser colocados entre aspas duplas, por exemplo:

```
modelerclient -stream mystream.str -Pusername="Joe User" -execute
```

Você também pode executar IBM SPSS Modeler estados e scripts desta maneira, usando as sinalizadores `-state` e `-script` , respectivamente.

Nota: Se você utilizar um parâmetro estruturado em um comando, deve-se preceder as aspas com uma barra invertida. Isso evita que as aspas sejam removidas durante a interpretação da sequência.

Depurando Argumentos de Linha de Comandos

Para depurar uma linha de comandos, utilize o comando `modelerclient` para ativar o IBM SPSS Modeler com os argumentos desejados. Isso permite verificar se os comandos serão executados conforme o esperado. Também é possível confirmar os valores de quaisquer parâmetros transmitidos a partir da linha de comandos na caixa de diálogo Parâmetros da Sessão (menu Ferramentas, Configurar Parâmetros da Sessão).

Argumentos do sistema

A tabela a seguir descreve os argumentos do sistema disponíveis para chamada da linha de comandos da interface com o usuário.

Tabela 35. Argumentos do sistema	
Argumento	**Comportamento/Descrição**
`@` <commandFile>	O caractere @ seguido por um nome de arquivo especifica uma lista de comandos. Quando `modelerclient` encontra um argumento que começa com @, ele opera nos comandos nesse arquivo como se estivesse na linha de comandos. Veja o tópico "Combinando Diversos Argumentos" na página 71 para obter mais informações.
`-directory` < dir>	Configura o diretório ativo padrão. No modo local, esse diretório é utilizado para ambos dados e também para saída. Exemplo: `-directory c:/` ou `-directory c:\\`

Tabela 35. Argumentos do sistema (continuação)	
Argumento	**Comportamento/Descrição**
`-server_directory` < dir>	Configura o diretório do servidor padrão para dados. O diretório ativo especificado usando o sinalizador `-directory` é utilizado para saída.
`-execute`	Depois de iniciar, executa qualquer fluxo, estado ou script carregado na inicialização. Se um script estiver carregado além de um fluxo ou estado, apenas o script será executado.
`-stream` < stream>	Na inicialização, carregue o fluxo especificado. Diversos fluxos podem ser especificados, no entanto, o último fluxo especificado será configurado como o fluxo atual.
`-stream_password` <password>	Na inicialização, carregue um fluxo criptografado por senha. Por exemplo, você pode executar um comando como: `clemb.exe -stream enc-stream1.str -stream_password Pass1234 -execute`
`-script` <script>	Na inicialização, carrega o script independente especificado. Isso pode ser especificado além de um fluxo ou estado conforme descrito abaixo, no entanto, apenas um script pode ser carregado na inicialização.
`-model` < model>	Na inicialização, carrega o modelo gerado (formato de arquivo `.gm`) especificado.
`-state` < state>	Na inicialização, carrega o estado salvo especificado.
`-project` < projeto>	Carrega o projeto especificado. Somente um projeto pode ser carregado na inicialização.
`-output` < output>	Na inicialização, carrega o objeto de saída salvo (formato de arquivo `.cou`).
`-help`	Exibe uma lista de argumentos de linha de comandos. Quando essa opção é especificada, todos os outros argumentos são ignorados e a tela Ajuda é exibida.
`-P` < name> = < value>	Utilizado para configurar um parâmetro de inicialização. Também pode ser utilizado para configurar propriedades do nó (parâmetros do slot).

Nota: Diretórios padrão também podem ser configurados na interface do usuário. Para acessar as opções, no menu Arquivo, escolha **Configurar Diretório Ativo** ou **Configurar Diretório do Servidor**.

Carregando Diversos Arquivos

Na linha de comandos, é possível carregar diversos fluxos, estados e saídas na inicialização ao repetir o argumento relevante para cada objeto carregado. Por exemplo, para carregar e executar dois fluxos chamados `report.str` e `train.str`, utilize o comando a seguir:

```
modelerclient -stream report.str -stream train.str -execute
```

Carregando objetos do IBM SPSS Repositório do Collaboration and Deployment Services

Como é possível carregar determinados objetos a partir de um arquivo ou a partir do IBM SPSS Repositório do Collaboration and Deployment Services (se licenciado), o prefixo de nome de arquivo `spsscr:`, e opcionalmente `file:` (para objetos no disco), informam ao IBM SPSS Modeler onde procurar pelo objeto. O prefixo funciona com os seguintes sinalizadores:

- `-stream`
- `-script`

- `-output`
- `-model`
- `-project`

Use o prefixo para criar um URI que especifica o local do objeto, por exemplo, `-stream "spsscr:///folder_1/scoring_stream.str"`. A presença do prefixo `spsscr:` requer que uma conexão válida com o IBM SPSS Repositório do Collaboration and Deployment Services seja especificada no mesmo comando. Portanto, por exemplo, o comando completo será semelhante ao seguinte:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

Observe que *deve-se* utilizar um URI na linha de comandos. O `REPOSITORY_PATH` mais simples não é suportado. (Ele funciona apenas dentro de scripts). Para obter mais detalhes sobre URIs para objetos no IBM SPSS Repositório do Collaboration and Deployment Services, consulte o tópico "Acessando Objetos no IBM SPSS Repositório do Collaboration and Deployment Services" na página 51.

Argumentos de parâmetro

Os parâmetros podem ser utilizados como sinalizadores durante a execução da linha de comandos do IBM SPSS Modeler. Em argumentos de linha de comando, o flag `-P` é usado para denotar um parâmetro do formulário `-P` <*nome*> = <*value*>.

Os parâmetros podem ser qualquer um dos seguintes:

- **Parâmetros simples** (ou parâmetros utilizados diretamente em expressões do CLEM).
- **Parâmetros de Slot**, também chamados de propriedades do nó. Esses parâmetros são utilizados para modificar as configurações de nós no fluxo. Consulte o tópico "Visão geral de propriedades do nó" na página 75 para obter mais informações.
- **Parâmetros da linha de comandos**, utilizados para alterar a chamada do IBM SPSS Modeler.

Por exemplo, é possível fornecer nomes de usuário e senhas de origem de dados como um sinalizador da linha de comandos, conforme a seguir:

```
modelerclient -stream response.str -P:databasenode.datasource="{\"ORA
10gR2\",user1,mypsw,false}"
```

O formato é o mesmo que o parâmetro `datasource` da propriedade do nó `databasenode`. Para obter mais informações, consulte: "Propriedades de databasenode" na página 93.

O último parâmetro deve ser configurado como `true` se você estiver passando uma senha codificada. Observe também que nenhum espaço de liderança deve ser usado na frente do nome de usuário e senha do banco de dados (a menos que, claro, seu nome de usuário ou senha realmente contenha um espaço líder).

Nota: Se o nó for nomeado, deve-se colocar o nome do nó entre aspas duplas e escapar as aspas com uma barra invertida. Por exemplo, se o nó da origem de dados no exemplo anterior possuir o nome `Source_ABC`, a entrada será a seguinte:

```
modelerclient -stream response.str
-P:databasenode.\"Source_ABC\".datasource="{\"ORA 10gR2\",
 user1,mypsw,true}"
```

Uma barra invertida também é necessária na frente das aspas que identificam um parâmetro estruturado, como no exemplo de origem de dados do TM1 a seguir:

```
clemb -server -hostname 9.115.21.169 -port 28053 -username administrator
    -execute -stream C:\Share\TM1_Script.str -P:tm1import.pm_host="http://9.115.21.163:9510/
pmhub/pm"
```

```
    -P:tm1import.tm1_connection={\"SData\",\"\",\"admin\",\"apple\"}
    -P:tm1import.selected_view={\"SalesPriorCube\",\"salesmargin%\"}
```

Nota: Se o nome do banco de dados (na propriedade `datasource`) contiver um ou mais espaços, pontos (também conhecidos como um "ponto final") ou sublinhados, será possível utilizar o formato de "barra invertida aspas duplas" para tratá-la como uma sequência. Por exemplo: `"{\"db2v9.7.6_linux\"}"` ou `"{\"TDATA 131\"}"`. Além disso, sempre coloque abaixo os valores de string `datasource` em cotações duplas e curly, como no exemplo a seguir: `"{\"SQL Server\",spssuser,abcd1234,false}"`.

Argumentos de conexão do servidor

O sinalizador `-server` informa ao IBM SPSS Modeler que ele deve se conectar a um servidor público e os sinalizadores `-hostname`, `-use_ssl`, `-port`, `-username`, `-password` e `-domain` são utilizados para informar ao IBM SPSS Modeler como conectar-se ao servidor público. Se nenhum argumento -server for especificado, o servidor padrão ou local será utilizado.

Exemplos

Para conectar-se a um servidor público:

```
modelerclient -server -hostname myserver -port 80 -username dminer
-password 1234 -stream mystream.str -execute
```

Para conectar-se a um cluster de servidores:

```
modelerclient -server -cluster "QA Machines" \
-spsscr_hostname pes_host -spsscr_port 8080 \
-spsscr_username asmith -spsscr_epassword xyz
```

Note que conectar a um cluster de servidores requer o Coordenador de Processos por meio de IBM SPSS Collaboration and Deployment Services, portanto, o argumento `-cluster` deve ser usado em combinação com as opções de conexão do repositório (`spsscr_*`). Veja o tópico “Argumentos de Conexão do IBM SPSS Repositório do Collaboration and Deployment Services” na página 70 para obter mais informações.

Tabela 36. Argumentos de conexão do servidor

Argumento	Comportamento/Descrição
`-server`	Executa o IBM SPSS Modeler no modo de servidor, conectando-se a um servidor público utilizando os sinalizadores `-hostname`, `-port`, `-username`, `-password`, e `-domain`.
`-hostname <name>`	O nome do host da máquina servidor. Disponível apenas no modo de servidor.
`-use_ssl`	Especifica que a conexão deve usar SSL (Secure Sockets Layer). Esse sinalizador é opcional; a configuração padrão é *não* usar SSL.
`-port < numer>`	O número da porta do servidor especificado. Disponível apenas no modo de servidor.

Tabela 36. Argumentos de conexão do servidor (continuação)	
Argumento	**Comportamento/Descrição**
`-cluster <name>`	Especifica uma conexão com um cluster de servidores e não com um servidor denominado; esse argumento é uma alternativa para os argumentos `hostname`, `port` e `use_ssl`. O nome é o nome do cluster ou um URI exclusivo que identifica o cluster no IBM SPSS Repositório do Collaboration and Deployment Services. O cluster de servidores é gerenciado pelo Coordenador de Processos por meio de IBM SPSS Collaboration and Deployment Services. Consulte o tópico "Argumentos de Conexão do IBM SPSS Repositório do Collaboration and Deployment Services" na página 70 para obter mais informações.
`-username <name>`	O nome do usuário com o qual efetuar logon no servidor. Disponível apenas no modo de servidor.
`-password <password>`	A senha com a qual efetuar logon no servidor. Disponível apenas no modo de servidor. **Nota:** Se o argumento `-password` não for usado, você será solicitado por uma senha.
`-epassword <` encodedpasswordstring>	A senha codificada com a qual efetuar logon no servidor. Disponível apenas no modo de servidor. **Nota:** Uma senha codificada pode ser gerada a partir do menu Ferramentas do aplicativo IBM SPSS Modeler .
`-domain <name>`	O domínio utilizado para efetuar logon no servidor. Disponível apenas no modo de servidor.
`-P` < name> = < value>	Utilizado para configurar um parâmetro de inicialização. Também pode ser utilizado para configurar propriedades do nó (parâmetros do slot).

Argumentos de Conexão do IBM SPSS Repositório do Collaboration and Deployment Services

Se desejar armazenar ou recuperar objetos a partir do IBM SPSS Collaboration and Deployment Services por meio da linha de comandos, deve-se especificar uma conexão válida com o IBM SPSS Repositório do Collaboration and Deployment Services. Por exemplo:

```
modelerclient -spsscr_hostname myhost -spsscr_port 8080
-spsscr_username myusername -spsscr_password mypassword
-stream "spsscr:///folder_1/scoring_stream.str" -execute
```

A tabela a seguir lista os argumentos que podem ser utilizados para configurar a conexão.

Tabela 37. Argumentos de conexão do IBM SPSS Repositório do Collaboration and Deployment Services	
Argumento	**Comportamento/Descrição**
`-spsscr_hostname` < nome do host ou endereço IP>	O nome do host ou o endereço IP do servidor no qual o IBM SPSS Repositório do Collaboration and Deployment Services está instalado.
`-spsscr_port` < numer>	O número da porta na qual o IBM SPSS Repositório do Collaboration and Deployment Services aceita conexões (geralmente 8080, por padrão).
`-spsscr_use_ssl`	Especifica que a conexão deve usar SSL (Secure Sockets Layer). Esse sinalizador é opcional; a configuração padrão é *não* usar SSL.

Tabela 37. Argumentos de conexão do IBM SPSS Repositório do Collaboration and Deployment Services (continuação)	
Argumento	**Comportamento/Descrição**
`-spsscr_username <name>`	O nome de usuário com o qual efetuar logon no IBM SPSS Repositório do Collaboration and Deployment Services.
`-spsscr_password <password>`	A senha com a qual efetuar logon no IBM SPSS Repositório do Collaboration and Deployment Services.
`-spsscr_epassword < senha codificada>`	A senha codificada com a qual efetuar logon no IBM SPSS Repositório do Collaboration and Deployment Services.
`-spsscr_providername <name>`	O provedor de autenticação usado para registro de logon no IBM SPSS Repositório do Collaboration and Deployment Services (Active Directory ou LDAP). Isso não é necessário se utilizando o provedor nativo (Repositório Local).

IBM SPSS Analytic Server argumentos de conexão

Se desejar armazenar ou recuperar objetos a partir do IBM SPSS Analytic Server por meio da linha de comandos, deve-se especificar uma conexão válida com o IBM SPSS Analytic Server.

Nota: O local padrão de Servidor Analítico é obtido a partir de SPSS Modeler Server. Os usuários também podem definir suas próprias conexões Servidor Analítico via **Ferramentas** > **Conexões do Servidor Analítico**.

A tabela a seguir lista os argumentos que podem ser utilizados para configurar a conexão.

Tabela 38. IBM SPSS Analytic Server argumentos de conexão	
Argumento	**Comportamento/Descrição**
`-analytic_server_username`	O nome de usuário com o qual efetuar logon no IBM SPSS Analytic Server.
`-analytic_server_password`	A senha com a qual efetuar logon no IBM SPSS Analytic Server.
`-analytic_server_epassword`	A senha codificada com o qual efetuar logon no IBM SPSS Analytic Server.
`-analytic_server_credential`	As credenciais utilizadas para efetuar logon no IBM SPSS Analytic Server.

Combinando Diversos Argumentos

Diversos argumentos podem ser combinados em um arquivo de comando único especificado na chamada utilizando o símbolo @ seguido pelo nome do arquivo. Isso permite reduzir a chamada da linha de comandos e superar quaisquer limitações do sistema operacional referentes ao comprimento de comando. Por exemplo, o comando de inicialização a seguir usa argumentos especificados no arquivo referenciado por `<commandFileName>`.

```
modelerclient @<commandFileName>
```

Coloque o nome do arquivo e o caminho para o arquivo de comando entre aspas se espaços forem necessários, conforme a seguir:

```
modelerclient @ "C:\Program
Files\IBM\SPSS\Modeler\nn\scripts\my_command_file.txt"
```

O arquivo de comando pode conter todos os argumentos anteriormente especificados individualmente na inicialização, com um argumento por linha. Por exemplo:

```
-stream report.str
-Porder.full_filename=APR_orders.dat
-Preport.filename=APR_report.txt
-execute
```

Ao gravar e fazer referência a arquivos de comando, assegure-se de seguir estas restrições:

- Utilize apenas um comando por linha.
- Não integre um argumento `@CommandFile` dentro de um arquivo de comando.

Capítulo 7. Referência de Propriedades

Visão geral de referência de propriedades

É possível especificar várias propriedades diferentes para nós, fluxos, projetos e SuperNodes. Algumas propriedades são comuns a todos os nós, como nome, anotação e dica de ferramenta, enquanto que outras são específicas para determinados tipos de nós. Outras propriedades se referem a operações de fluxo de alto nível, como o armazenamento em cache ou comportamento de SuperNode. As propriedades podem ser acessadas por meio da interface com o usuário padrão (por exemplo, quando abrir uma caixa de diálogo para editar opções para um nó) e também pode ser utilizadas de várias maneiras diferentes.

- As propriedades podem ser modificadas por meio de scripts, conforme descrito nesta seção. Para obter informações adicionais, consulte “Sintaxe para propriedades” na página 73.
- As propriedades do nó podem ser utilizadas em parâmetros de SuperNode.
- As propriedades do nó também podem ser usadas como parte de uma opção de linha de comandos (usando a sinalização `-P`) ao iniciar IBM SPSS Modeler.

No contexto de script dentro do IBM SPSS Modeler, as propriedades do nó e do fluxo são geralmente chamadas de **parâmetros de slot**. Neste guia, elas são referidas como propriedades de nó ou de fluxo.

Sintaxe para propriedades

As propriedades podem ser configuradas utilizando a sintaxe a seguir

```
OBJECT.setPropertyValue(PROPERTY, VALUE)
```

ou:

```
OBJECT.setKeyedPropertyValue(PROPERTY, KEY, VALUE)
```

O valor de propriedades pode ser recuperado utilizando a sintaxe a seguir:

```
VARIABLE = OBJECT.getPropertyValue(PROPERTY)
```

ou:

```
VARIABLE = OBJECT.getKeyedPropertyValue(PROPERTY, KEY)
```

em que `OBJECT` é um nó ou saída, `PROPERTY` é o nome da propriedade do nó que sua expressão referencia e `KEY` é o valor da chave para as propriedades definidas como chave. Por exemplo, a sintaxe a seguir é usada para localizar o nó de filtro e, em seguida, configure o padrão para incluir todos os campos e filtrar o campo `Age` a partir dos dados de recebimento de dados:

```
filternode = modeler.script.stream().findByType("filter", None)
filternode.setPropertyValue("default_include", True)
filternode.setKeyedPropertyValue("include", "Age", False)
```

Todos os nós utilizados no IBM SPSS Modeler podem ser localizados utilizando a função `findByType(TYPE, LABEL)` do fluxo. Pelo menos um de `TYPE` ou `LABEL` deve ser especificado.

Propriedades estruturadas

O script usa as propriedades estruturadas de duas maneiras para maior clareza durante a análise:

- Para fornecer estrutura para os nomes de propriedades para nós complexos, como Tipo, Filtro ou Balanceamento.

- Para fornecer um formato para especificar diversas propriedades de uma vez.

Estruturando para Interfaces Complexas

Os scripts para nós com tabelas e outras interfaces complexas (por exemplo, Tipo, Filtro e Balanceamento) devem seguir uma estrutura específica para que a análise seja executada corretamente. Essas propriedades precisam de um nome que seja mais complexo do que o nome para um identificador único; esse nome é chamado de chave. Por exemplo, em um nó Filtro, cada campo disponível (em seu lado de envio de dados) é ativado ou desativado. Para fazer referência a essas informações, o nó Filtro armazena um item de informações por campo (independentemente se cada campo for true ou false). Essa propriedade pode ter (ou receber) o valor `True` ou `False`. Suponha que um nó de Filtro denominado `mynode` tenha (em seu lado de envio de dados) um campo chamado `Age`. Para desativar, configure a propriedade `include`, com a chave `Age`, para o valor `False`, como a seguir:

```
mynode.setKeyedPropertyValue("include", "Age", False)
```

Estruturando para Configurar Diversas Propriedades

Para muitos nós, é possível designar mais de uma propriedade de nó ou de fluxo por vez. Isso é referido como o **comando de multiconjunto** ou **bloco de conjunto**.

Em alguns casos, uma propriedade estruturada pode ser muito complexa. Um exemplo é o seguinte:

```
sortnode.setPropertyValue("keys", [["K", "Descending"], ["Age",
"Ascending"], ["Na", "Descending"]])
```

Outra vantagem que as propriedades estruturadas possuem é a capacidade de configurar várias propriedades em um nó antes de o nó se tornar estável. Por padrão, um multiconjunto configura todas as propriedades no bloco antes de executar qualquer ação com base em uma configuração de propriedade individual. Por exemplo, ao definir um nó Arquivo Fixo, utilizar duas etapas para configurar as propriedades do campo resultaria em erros porque o nó não estará consistente até que ambas as configurações sejam válidas. Definir propriedades como um multiconjunto evita esse problema ao configurar as duas propriedades antes de atualizar o modelo de dados.

Abreviações

Abreviações padrão são utilizadas em toda a sintaxe das propriedades do nó. Aprender as abreviações é útil na construção de scripts.

Tabela 39. Abreviações padrão utilizadas em toda a sintaxe

Abreviação	**Significado**
abs	Valor absoluto
len	Comprimento
min	Mínimo
máx	Máximo
correl	Correlação
covar	Covariância
num	Número ou numérico
pct	Percentual ou porcentagem
transp	Transparência
xval	Validação cruzada

Tabela 39. Abreviações padrão utilizadas em toda a sintaxe (continuação)	
Abreviação	**Significado**
var	Variação ou variável (em nós de origem)

Exemplos de propriedade de nó e de fluxo

As propriedades do nó e de fluxo podem ser utilizadas de várias maneiras com o IBM SPSS Modeler. Elas são utilizadas com mais frequência como parte de um script, seja um **script independente** utilizado para automatizar diversos fluxos ou operações ou um **script de fluxo** utilizado para automatizar os processos dentro de um único fluxo. Também é possível especificar parâmetros do nó usando as propriedades do nó no SuperNode. No nível mais básico, as propriedades também podem ser utilizadas como uma opção de linha de comandos para iniciar o IBM SPSS Modeler. Usando o argumento `-p` como parte da chamada da linha de comandos, é possível utilizar uma propriedade de fluxo para alterar uma configuração no fluxo.

Tabela 40. Exemplos de propriedade de nó e de fluxo	
Propriedade	**Significado**
`s.max_size`	Refere-se à propriedade `max_size` do nó denominado `s`.
`s:samplenode.max_size`	Refere-se à propriedade `max_size` do nó denominado `s`, que deve ser um nó de Amostra.
`:samplenode.max_size`	Refere-se à propriedade `max_size` do nó Amostra no fluxo atual (deve haver apenas um nó Amostra).
`s:sample.max_size`	Refere-se à propriedade `max_size` do nó denominado `s`, que deve ser um nó de Amostra.
`t.direction.Age`	Refere-se à função do campo *Age* no nó Tipo `t`.
`:.max_size`	*** ILEGAL *** É necessário especificar o nome do nó ou o tipo de nó.

O exemplo `s:sample.max_size` ilustra que não é preciso digitar os tipos de nó por completo.

O exemplo `t.direction.Age` ilustra que alguns nomes de slots podem ser estruturados - em casos em que os atributos de um nó são mais complexos do que simplesmente slots individuais com valores individuais. Esses slots são chamados de propriedades **estruturadas** ou **complexas**.

Visão geral de propriedades do nó

Cada tipo de nó possui seu próprio conjunto de propriedades legais e cada propriedade possui um tipo. Este tipo pode ser um tipo geral – número, sinalizador ou sequência – caso em que as configurações da propriedade são forçadas para o tipo correto. Um erro será gerado se elas não puderem ser forçadas. Como alternativa, a referência da propriedade pode especificar o intervalo de valores legais, como `Discard`, `PairAndDiscard` e `IncludeAsText`, caso em que um erro será causado se algum outro valor for usado. As propriedades de sinalização devem ser lidas ou definidas usando valores de `true` e `false`. (Variações incluindo `Off`, `OFF`, `off`, `No`, `NO`, `no`, `n`, `N`, `f`, `F`, `false`, `False`, `FALSE` ou `0` também são reconhecidas ao configurar valores, mas poderão causar erros ao ler os valores de propriedade em alguns casos. Todos os outros valores são considerados como true. Usar `true` e `false` consistentemente evitará qualquer confusão.) Nas tabelas de referência desse guia, as propriedades estruturadas são indicadas dessa forma na coluna **Descrição da propriedade** e seus formatos de uso são fornecidos.

Propriedades Comuns do Nó

Um número propriedades é comum para todos os nós (incluindo SuperNodes) no IBM SPSS Modeler.

Tabela 41. Propriedades comuns do nó		
Nome da propriedade	**Tipo de dados**	**Descrição da propriedade**
use_custom_name	*sinalização*	
name	*sequência*	Propriedade somente leitura que lê o nome (automático ou customizado) para um nó na tela.
custom_name	*sequência*	Especifica um nome customizado para o nó.
tooltip	*sequência*	
annotation	*sequência*	
keywords	*sequência*	Slot estruturado que especifica uma lista de palavras-chave associadas ao objeto (por exemplo, ["Keyword1" "Keyword2"]).
cache_enabled	*sinalização*	
node_type	source_supernode process_supernode terminal_supernode todos os nomes de nó conforme especificado para script	Propriedade somente leitura utilizada para referenciar um nó por tipo. Por exemplo, em vez de se referir a um nó apenas por nome, como real_income, também é possível especificar o tipo, como userinputnode ou filternode.

Propriedades específicas do SuperNode são discutidas separadamente, assim como com todos os outros nós. Consulte o tópico Capítulo 21, “Propriedades do Supernó”, na página 459 para obter informações adicionais.

Capítulo 8. Propriedades do Fluxo

Uma variedade de propriedades de fluxo pode ser controlada por script. Para referenciar as propriedades do fluxo, deve-se configurar o método de execução para utilizar scripts:

```
stream = modeler.script.stream()
stream.setPropertyValue("execute_method", "Script")
```

Exemplo

A propriedade do nó é utilizada para referenciar os nós no fluxo atual. O script de fluxo a seguir fornece um exemplo:

```
stream = modeler.script.stream()
annotation = stream.getPropertyValue("annotation")

annotation = annotation + "\n\nThis stream is called \"" + stream.getLabel() + "\" and
 contains the following nodes:\n"

for node in stream.iterator():
    annotation = annotation + "\n" + node.getTypeName() + " node called \"" + node.getLabel()
 + "\""

stream.setPropertyValue("annotation", annotation)
```

O exemplo acima utiliza a propriedade do nó para criar uma lista de todos os nós no fluxo e gravar essa lista nas anotações de fluxo. A anotação produzida é semelhante a esta:

```
This stream is called "druglearn" and contains the following nodes:

type node called "Define Types"
derive node called "Na_to_K"
variablefile node called "DRUG1n"
neuralnetwork node called "Drug"
c50 node called "Drug"
filter node called "Discard Fields"
```

As propriedades do fluxo são descritas na tabela a seguir.

Tabela 42. Propriedades do Fluxo

Nome da propriedade	Tipo de dados	Descrição da propriedade
execute_method	Normal Script	

Tabela 42. Propriedades do Fluxo (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
date_baseline	*number*	
date_2digit_baseline	*number*	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	
time_rollover	*sinalização*	
import_datetime_as_string	*sinalização*	
decimal_places	*number*	
decimal_symbol	Default Period Comma	
angles_in_radians	*sinalização*	
use_max_set_size	*sinalização*	
max_set_size	*number*	

Tabela 42. Propriedades do Fluxo (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
ruleset_evaluation	Voting FirstHit	
refresh_source_nodes	*sinalização*	Utilize para atualizar os nós de origem automaticamente após a execução do fluxo.
script	*sequência*	
annotation	*sequência*	
name	*sequência*	**Nota:** Esta propriedade é somente leitura. Se desejar alterar o nome de um fluxo, deve-se salvá-lo com um nome diferente.
parameters		Utilize esta propriedade para atualizar os parâmetros de fluxo a partir de dentro de um script independente.
nodes		Consulte as informações detalhadas abaixo.
encoding	SystemDefault "UTF-8"	
stream_rewriting	*Booleano*	
stream_rewriting_maximise_sql	*Booleano*	
stream_rewriting_optimise_clem_execution	*Booleano*	
stream_rewriting_optimise_syntax_execution	*Booleano*	
enable_parallelism	*Booleano*	
sql_generation	*Booleano*	
database_caching	*Booleano*	
sql_logging	*Booleano*	
sql_generation_logging	*Booleano*	
sql_log_native	*Booleano*	
sql_log_prettyprint	*Booleano*	
record_count_suppress_input	*Booleano*	

Tabela 42. Propriedades do Fluxo (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
`record_count_feedback_interval`	*Número inteiro*	
`use_stream_auto_create_node_settings`	*Booleano*	Se true, as configurações específicas do fluxo serão utilizadas, caso contrário, as preferências do usuário são utilizadas.
`create_model_applier_for_new_models`	*Booleano*	Se true, quando um construtor de modelo cria um novo modelo e ele não tiver links de atualização ativos, um novo aplicador de modelo será incluído. **Nota:** Se você estiver utilizando o IBM SPSS Modeler Batch versão 15, deve-se incluir explicitamente o aplicador de modelo em seu script.

`create_model_applier_update_links`	`createEnabled`
	`createDisabled`
	`doNotCreate`
		Define o tipo de link criado quando um nó aplicador de modelo é incluído automaticamente.

`create_source_node_from_builders`	*Booleano*	Se true, quando um construtor de origem cria uma nova saída de origem e ele não tiver links de atualização ativos, um novo nó de origem será incluído.

`create_source_node_update_links`	`createEnabled`
	`createDisabled`
	`doNotCreate`
		Define o tipo de link criado quando um nó de origem é incluído automaticamente.

`has_coordinate_system`	*Booleano*	Se true, aplica um sistema de coordenadas no fluxo inteiro.
`coordinate_system`	*sequência*	O nome do sistema de coordenadas projetadas selecionado.

`deployment_area`	`ModelRefresh`
	`Scoring`
	`None`
		Escolha como deseja implementar o fluxo. Se esse valor for configurado como None, nenhuma outra entrada de implementação será usada.

`scoring_terminal_node_id`	*sequência*	Escolha a ramificação de pontuação do fluxo. Ele pode ser qualquer nó terminal no stream.
`scoring_node_id`	*sequência*	Escolha o nugget na ramificação de pontuação.
`model_build_node_id`	*sequência*	Escolha o nó de modelagem no fluxo

Capítulo 9. Propriedades do Nó de Origem

Propriedades comuns do nó de origem

Propriedades que são comuns a todos os nós de origem são listadas abaixo, com informações sobre os nós específicos nos tópicos a seguir.

Exemplo 1

```
varfilenode = modeler.script.stream().create("variablefile", "Var. File")
varfilenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
varfilenode.setKeyedPropertyValue("check", "Age", "None")
varfilenode.setKeyedPropertyValue("values", "Age", [1, 100])
varfilenode.setKeyedPropertyValue("type", "Age", "Range")
varfilenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Exemplo 2

Este script assume que o arquivo de dados especificado contém um campo chamado Region que representa uma sequência de várias linhas.

```
from modeler.api import StorageType
from modeler.api import MeasureType

# Create a Variable File node that reads the data set containing
# the "Region" field
varfilenode = modeler.script.stream().create("variablefile", "My Geo Data")
varfilenode.setPropertyValue("full_filename", "C:/mydata/mygeodata.csv")
varfilenode.setPropertyValue("treat_square_brackets_as_lists", True)

# Override the storage type to be a list...
varfilenode.setKeyedPropertyValue("custom_storage_type", "Region",
StorageType.LIST)
# ...and specify the type if values in the list and the list depth
varfilenode.setKeyedPropertyValue("custom_list_storage_type", "Region",
StorageType.INTEGER)
varfilenode.setKeyedPropertyValue("custom_list_depth", "Region", 2)

# Now change the measurement to indentify the field as a geospatial value...
varfilenode.setKeyedPropertyValue("measure_type", "Region",
MeasureType.GEOSPATIAL)
# ...and finally specify the necessary information about the specific
# type of geospatial object
varfilenode.setKeyedPropertyValue("geo_type", "Region", "MultiLineString")
varfilenode.setKeyedPropertyValue("geo_coordinates", "Region", "2D")
varfilenode.setKeyedPropertyValue("has_coordinate_system", "Region", True)
varfilenode.setKeyedPropertyValue("coordinate_system", "Region",
 "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabela 43. Propriedades comuns do nó de origem

Nome da propriedade	Tipo de dados	Descrição da propriedade
direction	Input Target Both None Partition Split Frequency RecordID	Propriedade definida como chave para funções de campo. Formato de uso: NODE.direction.FIELDNAME **Nota:** Os valores de In e Out estão agora descontinuados. O suporte para eles poderá ser retirado em uma liberação futura.
type	Range Flag Set Typeless Discrete Ordered Set Default	Tipo de campo. Configurar essa propriedade como *Default* limpará qualquer configuração da propriedade values e, se value_mode for configurado para *Specify*, ele será reconfigurado para *Read*. Se value_mode já estiver configurado para *Pass* ou *Read*, ele não será afetado pela configuração type. Formato de uso: NODE.type.FIELDNAME
storage	Unknown String Integer Real Time Date Timestamp	Propriedade definida como chave somente leitura para tipo de armazenamento de campo. Formato de uso: NODE.storage.FIELDNAME

Tabela 43. Propriedades comuns do nó de origem (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
check	None Nullify Coerce Discard Warn Abort	Propriedade definida como chave para verificação de tipo e de intervalo de campo Formato de uso: NODE.check.FIELDNAME
values	[*value value*]	Para um campo contínuo (intervalo), o primeiro valor é o mínimo e o último valor é o máximo. Para os campos nominais (conjunto), especifique todos os valores. Para campos de sinalização, o primeiro valor representa *false* e o último valor representa *true*. Configurar esta propriedade configura automaticamente a propriedade value_mode para *Specify*. O armazenamento é determinado com base no primeiro valor na lista, por exemplo, se o primeiro valor for uma *string*, então o armazenamento será configurado como Sequência. Formato de uso: NODE.values.FIELDNAME
value_mode	Read Pass Read+ Current Specify	Determina como os valores são configurados para um campo na próxima transmissão de dados. Formato de uso: NODE.value_mode.FIELDNAME Observe que não é possível configurar essa propriedade para *Specify* diretamente; para utilizar valores específicos, configure a propriedade values.
default_value_mode	Read Pass	Especifica o método padrão para configurar valores para todos os campos. Formato de uso: NODE.default_value_mode Esta configuração pode ser substituída para campos específicos utilizando a propriedade value_mode.

Tabela 43. Propriedades comuns do nó de origem (continuação)		
Nome da propriedade	**Tipo de dados**	**Descrição da propriedade**
`extend_values`	*sinalização*	Aplica-se quando `value_mode` for configurado para *Read*. Configure para *T* para incluir valores recém-lidos em quaisquer valores existentes para o campo. Configure para *F* para descartar valores existentes a favor dos valores recém-lidos. Formato de uso: `NODE.extend_values.FIELDNAME`
`value_labels`	*sequência*	Utilizado para especificar um rótulo de valor. Observe que os valores devem ser especificados primeiro.
`enable_missing`	*sinalização*	Quando configurado para *T*, ativa o rastreamento de valores omissos para o campo. Formato de uso: `NODE.enable_missing.FIELDNAME`
`missing_values`	[*value value ...*]	Especifica valores de dados que denotam dados ausentes. Formato de uso: `NODE.missing_values.FIELDNAME`
`range_missing`	*sinalização*	Quando esta propriedade é configurada como *T*, especifica se um intervalo de valores omissos (em branco) é definido para um campo. Formato de uso: `NODE.range_missing.FIELDNAME`
`missing_lower`	*sequência*	Quando `range_missing` é true, especifica o limite inferior do limite de valor ausente. Formato de uso: `NODE.missing_lower.FIELDNAME`
`missing_upper`	*sequência*	Quando `range_missing` é true, especifica o limite superior do intervalo de valor ausente. Formato de uso: `NODE.missing_upper.FIELDNAME`

Tabela 43. Propriedades comuns do nó de origem (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
null_missing	*sinalização*	Quando esta propriedade é configurada como *T*, valores nulos (valores indefinidos que são exibidos como $null$ no software) são considerados valores omissos. Formato de uso: NODE.null_missing.FIELDNAME
whitespace_missing	*sinalização*	Quando esta propriedade é configurada como *T*, valores que contêm apenas espaços em branco (espaços, tabulações e novas linhas) são considerados valores omissos. Formato de uso: NODE.whitespace_missing.FIELDNAME
description	*sequência*	Utilizado para especificar um rótulo ou descrição de campo.
default_include	*sinalização*	Propriedade definida como chave para especificar se o comportamento padrão é transmitir ou filtrar os campos: NODE.default_include Exemplo: set mynode:filternode.default_include = false
include	*sinalização*	Propriedade definida como chave utilizada para determinar se campos individuais são incluídos ou filtrados: NODE.include.FIELDNAME.
new_name	*sequência*	

Tabela 43. Propriedades comuns do nó de origem (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Esta propriedade definida como chave é semelhante a type na medida em que pode ser usada para definir a medição associada ao campo. O que é diferente é que no script Python, a função de configurador também pode ser passada um dos valores MeasureType enquanto a função que obtém sempre retornará nos valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de coleção (listas com uma profundidade 0), essa propriedade definida como chave define o tipo de medição associado aos valores subjacentes.

Tabela 43. Propriedades comuns do nó de origem (continuação)

Nome da propriedade	Tipo de dados	Descrição da propriedade
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Para campos geoespaciais, esta propriedade definida como chave define o tipo de objeto geoespacial representado por este campo. Isso deverá estar consistente com a profundidade da lista dos valores.
has_coordinate_system	*Booleano*	Para campos geoespaciais, essa propriedade define se esse campo tem um sistema de coordenadas
coordinate_system	*sequência*	Para campos geoespaciais, esta propriedade definida como chave define o sistema de coordenadas para este campo.
custom_storage_type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Esta propriedade definida como chave é semelhante a custom_storage na medida que pode ser usada para definir o armazenamento de substituição para o campo. O que é diferente é que no script Python, a função de configurador também pode ser passada um dos valores StorageType enquanto a função que obtém sempre retornará nos valores StorageType.

Tabela 43. Propriedades comuns do nó de origem (continuação)		
Nome da propriedade	**Tipo de dados**	**Descrição da propriedade**

`custom_list_storage_type`	`String` / `MeasureType.STRING`
	
	`Integer` / `MeasureType.INTEGER`
	
	`Real` / `MeasureType.REAL`
	
	`Time` / `MeasureType.TIME`
	
	`Date` / `MeasureType.DATE`
	
	`Timestamp` / `MeasureType.TIMESTAMP`
		Para campos de lista, esta propriedade definida como chave especifica o tipo de armazenamento dos valores subjacentes.

`custom_list_depth`	*Número inteiro*	Para campos de lista, esta propriedade definida como chave especifica a profundidade do campo
`max_list_length`	*Número inteiro*	Disponível apenas para dados com um nível de medição de *Geoespacial* ou *Coleção*. Configure o comprimento máximo da lista ao especificar o número de elementos que a lista pode conter.
`max_string_length`	*Número inteiro*	Apenas disponível para dados *sem tipo* e usado quando você está gerando SQL para criar uma tabela. Digite o valor da maior sequência em seus dados; isso gera uma coluna na tabela que é grande o suficiente para conter a sequência.

Propriedades de asimport

A origem do Servidor Analítico permite executar um fluxo no Hadoop Distributed File System (HDFS).

Exemplo

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false
","","","","",""])
```

Tabela 44. Propriedades de asimport		
propriedades `asimport`	**Tipo de dados**	**Descrição da propriedade**
`data_source`	*sequência*	O nome da origem de dados.

Tabela 44. Propriedades de asimport (continuação)		
propriedades asimport	**Tipo de dados**	**Descrição da propriedade**
use_default_as	*Booleano*	Se configurado como True, usa a conexão Servidor Analítico padrão configurada no arquivo options.cfg do servidor. Se configurado como False, usa a conexão desse nó.
connection	["string","string","string", "string","string","string","string", "string" ,"string","string", "string" ,"string"]	Uma propriedade de lista contendo os detalhes da conexão do Servidor Analítico O formato é: ["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"] Em que: is_secure_connect: indica se a conexão segura é usada e é true ou false. use-kerberos-auth: indica se a autenticação do Kerberos é usada e é true ou false. enable-kerberos-debug: indica se o modo de depuração da autenticação do Kerberos é usado e se é true ou false

Propriedades do Nó cognosimport

O nó de origem IBM Cognos importa dados dos bancos de dados do Cognos Analytics .

Exemplo

```
node = stream.create("cognosimport", "My node")
node.setPropertyValue("cognos_connection", ["http://mycogsrv1:9300/p2pd/
servlet/dispatch",
 True, "", "", ""])
node.setPropertyValue("cognos_package_name", "/Public Folders/GOSALES")
node.setPropertyValue("cognos_items", ["[GreatOutdoors].[BRANCH].
[BRANCH_CODE]", "[GreatOutdoors]
.[BRANCH].[COUNTRY_CODE]"])
```

Tabela 45. Propriedades do nó cognosimport

Propriedades do nó `cognosimport`	**Tipo de dados**	**Descrição da propriedade**

`mode`	`Data`
	
	`Report`
		Especifica se deve importar dados do Cognos (padrão) ou relatórios.

Tabela 45. Propriedades do nó cognosimport (continuação)		
Propriedades do nó `cognosimport`	**Tipo de dados**	**Descrição da propriedade**
`cognos_connection`	*["string",flag,"string", "string","string"]*	Uma propriedade de lista que contém os detalhes de conexão com o servidor Cognos. O formato é: `["Cognos_server_URL", login_mode, "namespace", "username", "password"]` em que: `Cognos_server_URL` é a URL do servidor Cognos que contém a origem. `login_mode` indica se login anônimo é usado e é `true` ou `false`; se configurado para `true`, os campos a seguir deverão ser configurados para `""`. `namespace` especifica o provedor de autenticação de segurança utilizado para efetuar logon no servidor. `username` e `password` são aqueles utilizados para efetuar logon no servidor Cognos. Ao invés de `login_mode`, os modos a seguir também estão disponíveis: • `anonymousMode`. Por exemplo: `['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]` • `credentialMode`. Por exemplo: `['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]` • `storedCredentialMode`. Por exemplo: `['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"]` Em que `stored_credential_name` é o nome de uma credencial do Cognos no repositório.

Tabela 45. Propriedades do nó cognosimport (continuação)

Propriedades do nó cognosimport	**Tipo de dados**	**Descrição da propriedade**
cognos_package_name	*sequência*	O caminho e o nome do pacote Cognos a partir do qual você está importando objetos de dados, por exemplo: `/Public Folders/GOSALES` **Nota:** Apenas barras são válidas.
cognos_items	*["campo", "campo" ..., "campo"]*	O nome de um ou mais objetos de dados a serem importados. O formato do *campo* é [*namespace*], [*query_subject*]. [*query_item*]
cognos_filters	*campo*	O nome de um ou mais filtros para aplicar antes de importar os dados.
cognos_data_parameters	*lista*	Valores para parâmetros de prompt para dados. Os pares de nome e valor são colocados entre colchetes, pares múltiplos são separados por vírgulas e a sequência inteira é colocada entre colchetes. Formato: [["*param1*", "*value*"],...,["*paramN*", "*value*"]]
cognos_report_directory	*campo*	O caminho do Cognos de uma pasta ou pacote a partir do qual importar relatórios, por exemplo: `/Public Folders/GOSALES` **Nota:** Apenas barras são válidas.
cognos_report_name	*campo*	O caminho e o nome no local do relatório de um relatório a ser importado.
cognos_report_parameters	*lista*	Valores para parâmetros de relatório. Os pares de nome e valor são colocados entre colchetes, pares múltiplos são separados por vírgulas e a sequência inteira é colocada entre colchetes. Formato: [["*param1*", "*value*"],...,["*paramN*", "*value*"]]

Propriedades de databasenode

É possível usar o nó de Banco de Dados para importar dados de uma variedade de outros pacotes usando ODBC (Open Database Connectivity), incluindo Microsoft SQL Server, Db2, Oraclee outros.

Exemplo

```
import modeler.api
stream = modeler.script.stream()
node = stream.create("database", "My node")
node.setPropertyValue("mode", "Table")
node.setPropertyValue("query", "SELECT * FROM drug1n")
node.setPropertyValue("datasource", "Drug1n_db")
node.setPropertyValue("username", "spss")
node.setPropertyValue("password", "spss")
node.setPropertyValue("tablename", ".Drug1n")
```

Tabela 46. Propriedades de databasenode

propriedades databasenode	**Tipo de dados**	**Descrição da propriedade**

`mode`	`Table`
	
	`Query`
		Especifique *Table* para conectar-se a uma tabela de banco de dados utilizando os controles de caixa de diálogo ou especifique *Query* para consultar o banco de dados selecionado utilizando SQL.

`datasource`	*sequência*	Nome do banco de dados (consulte também a nota a seguir).
`username`	*sequência*	Detalhes de conexão com o banco de dados (consulte também a nota a seguir).
`password`	*sequência*	
`credential`	*sequência*	Nome da credencial armazenada no IBM SPSS Collaboration and Deployment Services. Isso pode ser utilizado ao invés das propriedades `username` e `password`. O nome do usuário e a senha da credencial devem corresponder ao nome do usuário e à senha necessários para acessar o banco de dados
`use_credential`		Configure para `True` ou `False`.
`epassword`	*sequência*	Especifica uma senha codificada como uma alternativa para codificar permanentemente uma senha em um script. Consulte o tópico “Gerando uma Senha Codificada” na página 54 para obter informações adicionais. Esta propriedade é somente leitura durante a execução.
`tablename`	*sequência*	Nome da tabela que deseja acessar.

Tabela 46. Propriedades de databasenode (continuação)		
propriedades databasenode	**Tipo de dados**	**Descrição da propriedade**
strip_spaces	None Left Right Both	Opções para descartar espaços iniciais e finais nas sequências.
use_quotes	AsNeeded Always Never	Especifique se os nomes de tabelas e de colunas são colocados entre aspas quando as consultas são enviadas ao banco de dados (por exemplo, se contiverem espaços ou pontuação).
query	*sequência*	Especifica o código SQL para a consulta que deseja enviar.

Nota: Se o nome do banco de dados (na propriedade datasource) contiver espaços ao invés de propriedades individuais para datasource, username e password, uma única propriedade origem de dados poderá ser usada no formato a seguir:

Tabela 47. Propriedades de databasenode - específica da origem de dados		
propriedades databasenode	**Tipo de dados**	**Descrição da propriedade**
datasource	*sequência*	Formato: [database_name,username,password[,true | false]] O último parâmetro é para uso com senhas criptografadas. Se este for configurado como true, a senha será decriptografada antes de usar.

Utilize este formato se também estiver alterando a origem de dados, no entanto, se desejar alterar apenas o nome de usuário ou a senha, será possível utilizar as propriedades username ou password.

Propriedades de datacollectionimportnode

O nó de Importação de dados do Coleta de Dados importa dados de pesquisa de opinião com base no Modelo de Dados do Coleta de Dados usado por produtos de pesquisa de mercado O Coleta de Dados Data Library deve ser instalado para usar este nó.

Exemplo

```
node = stream.create("datacollectionimport", "My node")
node.setPropertyValue("metadata_name", "mrQvDsc")
node.setPropertyValue("metadata_file", "C:/Program Files/IBM/SPSS/
DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("casedata_name", "mrQvDsc")
node.setPropertyValue("casedata_source_type", "File")
```

```
node.setPropertyValue("casedata_file", "C:/Program Files/IBM/SPSS/
DataCollection/DDL/Data/
Quanvert/Museum/museum.pkd")
node.setPropertyValue("import_system_variables", "Common")
node.setPropertyValue("import_multi_response", "MultipleFlags")
```

Tabela 48. Propriedades de datacollectionimportnode

propriedades datacollectionimportnode	**Tipo de dados**	**Descrição da propriedade**
metadata_name	*sequência*	O nome do MDSC. O valor especial DimensionsMDD indica que o documento de metadados padrão do Coleta de Dados deve ser utilizado. Outros valores possíveis incluem: mrADODsc mrI2dDsc mrLogDsc mrQdiDrsDsc mrQvDsc mrSampleReportingMDSC mrSavDsc mrSCDsc mrScriptMDSC O valor especial none indica que não há nenhum MDSC.
metadata_file	*sequência*	Nome do arquivo no qual os metadados são armazenados.

Tabela 48. Propriedades de datacollectionimportnode (continuação)

propriedades datacollectionimportnode	**Tipo de dados**	**Descrição da propriedade**
casedata_name	*sequência*	O nome do CDSC. valores possíveis incluem: mrADODsc mrI2dDsc mrLogDsc mrPunchDSC mrQdiDrsDsc mrQvDsc mrRdbDsc2 mrSavDsc mrScDSC mrXmlDsc O valor especial none indica que não há nenhum CDSC.
casedata_source_type	Unknown File Folder UDL DSN	Indica o tipo de origem do CDSC.
casedata_file	*sequência*	Quando casedata_source_type for *File*, especifica o arquivo que contém os dados do caso.
casedata_folder	*sequência*	Quando casedata_source_type for *Folder*, especifica a pasta que contém os dados do caso.
casedata_udl_string	*sequência*	Quando casedata_source_type for *UDL*, especifica a sequência de conexões com o OLD-DB para a origem de dados que contém os dados do caso.

Tabela 48. Propriedades de datacollectionimportnode (continuação)		
propriedades datacollectionimportnode	**Tipo de dados**	**Descrição da propriedade**
casedata_dsn_string	*sequência*	Quando casedata_source_type for *DSN*, especifica a sequência de conexões com o ODBC para a origem de dados.
casedata_project	*sequência*	Ao ler os dados do caso a partir de um banco de dados do Coleta de Dados, é possível inserir o nome do projeto. Para todos os outros tipos de dados do caso, essa configuração deverá ser deixada em branco.
version_import_mode	All Latest Specify	Define como as versões devem ser manipuladas.
specific_version	*sequência*	Quando version_import_mode for *Specify*, define a versão dos dados do caso a serem importados.
use_language	*sequência*	Define se os rótulos de um idioma específico devem ser utilizados.
language	*sequência*	Se use_language for true, define o código de idioma a ser utilizado na importação. O código de idioma deve ser um dos códigos disponíveis nos dados do caso.
use_context	*sequência*	Define se um contexto específico deve ser importado. Os contextos são utilizados para variar a descrição associada às respostas.
context	*sequência*	Se use_context for true, define o contexto a ser importado. O contexto deve ser um dos contextos disponíveis nos dados do caso.
use_label_type	*sequência*	Define se um tipo específico de rótulo deve ser importado.
label_type	*sequência*	Se use_label_type for true, define o tipo de rótulo a ser importado. O tipo de rótulo deve ser um dos tipos disponíveis nos dados do caso.
user_id	*sequência*	Para bancos de dados que requerem um login explícito, é possível fornecer um ID de usuário e senha para acessar a origem de dados.
password	*sequência*	

Tabela 48. Propriedades de datacollectionimportnode (continuação)		
propriedades datacollectionimportnode	**Tipo de dados**	**Descrição da propriedade**
import_system_variables	Common None All	Especifica quais variáveis do sistema são importadas.
import_codes_variables	*sinalização*	
import_sourcefile_variables	*sinalização*	
import_multi_response	MultipleFlags Single	

Propriedades de excelimportnode

O nó Importação do Excel importa dados do Microsoft Excel no formato de arquivo .xlsx. Uma origem de dados ODBC não é necessária.

Exemplos

```
#To use a named range:
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("use_named_range", True)
node.setPropertyValue("named_range", "DRUG")
node.setPropertyValue("read_field_names", True)

#To use an explicit range:
node = stream.create("excelimport", "My node")
node.setPropertyValue("excel_file_type", "Excel2007")
node.setPropertyValue("full_filename", "C:/drug.xlsx")
node.setPropertyValue("worksheet_mode", "Name")
node.setPropertyValue("worksheet_name", "Drug")
node.setPropertyValue("explicit_range_start", "A1")
node.setPropertyValue("explicit_range_end", "F300")
```

Tabela 49. Propriedades de excelimportnode		
propriedades excelimportnode	**Tipo de dados**	**Descrição da propriedade**
excel_file_type	Excel2007	
full_filename	*sequência*	O nome do arquivo completo, incluindo o caminho.
use_named_range	*Booleano*	Especifica se um intervalo nomeado deve ser usado. Se true, a propriedade named_range será utilizada para especificar o intervalo a ser lido e as outras configurações de planilha e de intervalo de dados serão ignoradas.

Tabela 49. Propriedades de excelimportnode (continuação)		
propriedades excelimportnode	**Tipo de dados**	**Descrição da propriedade**
named_range	*sequência*	
worksheet_mode	Index Name	Especifica se a planilha é definida por índice ou nome.
worksheet_index	*Número inteiro*	O índice da planilha a ser lido, começando com 0 para a primeira planilha, 1 para a segunda, e assim por diante.
worksheet_name	*sequência*	O nome da planilha a ser lida.
data_range_mode	FirstNonBlank ExplicitRange	Especifica como o intervalo deve ser determinado.
blank_rows	StopReading ReturnBlankRows	Quando data_range_mode é *FirstNonBlank*, especifica como as linhas em branco devem ser tratadas.
explicit_range_start	*sequência*	Quando data_range_mode é *ExplicitRange*, especifica o ponto de início do intervalo a ser lido.
explicit_range_end	*sequência*	
read_field_names	*Booleano*	Especifica se a primeira linha no intervalo especificado deve ser utilizada como nomes de campo (coluna).
scanLineCount	*Número inteiro*	Especifica o número de linhas para varrer a coluna e o tipo de armazenamento. O padrão é 200.

propriedades extensionimportnode

Com o nó de importação de extensão, é possível executar scripts R ou Python for Spark para importar dados.

Exemplo de Python for Spark

```
###### Script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_importer", "extension_importer")
node.setPropertyValue("syntax_type", "Python")

python_script = """
import spss.pyspark
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

_schema = StructType([StructField('id', LongType(), nullable=False), \
StructField('age', LongType(), nullable=True), \
StructField('Sex', StringType(), nullable=True), \
```

```
StructField('BP', StringType(), nullable=True), \
StructField('Cholesterol', StringType(), nullable=True), \
StructField('K', DoubleType(), nullable=True), \
StructField('Na', DoubleType(), nullable=True), \
StructField('Drug', StringType(), nullable=True)])

if cxt.isComputeDataModelOnly():
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    if df is None:
        drugList=[(1,23,'F','HIGH','HIGH',0.792535,0.031258,'drugY'), \
(2,47,'M','LOW','HIGH',0.739309,0.056468,'drugC'),\
              (3,47,'M','LOW','HIGH',0.697269,0.068944,'drugC'),\
              (4,28,'F','NORMAL','HIGH',0.563682,0.072289,'drugX'),\
              (5,61,'F','LOW','HIGH',0.559294,0.030998,'drugY'),\
              (6,22,'F','NORMAL','HIGH',0.676901,0.078647,'drugX'),\
              (7,49,'F','NORMAL','HIGH',0.789637,0.048518,'drugY'),\
              (8,41,'M','LOW','HIGH',0.766635,0.069461,'drugC'),\
              (9,60,'M','NORMAL','HIGH',0.777205,0.05123,'drugY'),\
              (10,43,'M','LOW','NORMAL',0.526102,0.027164,'drugY')]
        sqlcxt = cxt.getSparkSQLContext()
        rdd = cxt.getSparkContext().parallelize(drugList)
        print 'pyspark read data count = '+str(rdd.count())
        df = sqlcxt.createDataFrame(rdd, _schema)

    cxt.setSparkOutputData(df)
"""

node.setPropertyValue("python_syntax", python_script)
```

Exemplo de R

```
#### Script example for R
node.setPropertyValue("syntax_type", "R")

R_script = """# 'JSON Import' Node v1.0 for IBM SPSS Modeler
# 'RJSONIO' package created by Duncan Temple Lang - http://cran.r-project.org/web/packages/
RJSONIO
# 'plyr' package created by Hadley Wickham http://cran.r-project.org/web/packages/plyr
# Node developer: Danil Savine - IBM Extreme Blue 2014
# Description: This node allows you to import into SPSS a table data from  a JSON.
# Install function for packages
packages <- function(x){
  x <- as.character(match.call()[[2]])
  if (!require(x,character.only=TRUE)){
    install.packages(pkgs=x,repos="http://cran.r-project.org")
    require(x,character.only=TRUE)
  }
}
#  packages
packages(RJSONIO)
packages(plyr)
### This function is used to generate automatically the dataModel
getMetaData <- function (data) {
  if (dim(data)[1]<=0) {

    print("Warning : modelerData has no line, all fieldStorage fields set to strings")
    getStorage <- function(x){return("string")}

  } else {

    getStorage <- function(x) {
      res <- NULL
      #if x is a factor, typeof will return an integer so we treat the case on the side
      if(is.factor(x)) {
        res <- "string"
      } else {
        res <- switch(typeof(unlist(x)),
                      integer = "integer",
                      double = "real",
                      character = "string",
                      "string")
      }
      return (res)
    }
  }

  col = vector("list", dim(data)[2])
```

```
  for (i in 1:dim(data)[2]) {
    col[[i]] <- c(fieldName=names(data[i]),
                  fieldLabel="",
                  fieldStorage=getStorage(data[i]),
                  fieldMeasure="",
                  fieldFormat="",
                  fieldRole="")
  }
  mdm<-do.call(cbind,col)
  mdm<-data.frame(mdm)
  return(mdm)
}
# From JSON to a list
txt <- readLines('C:/test.json')
formatedtxt <- paste(txt, collapse = '')
json.list <- fromJSON(formatedtxt)
  # Apply path to json.list
  if(strsplit(x='true', split='
  ' ,fixed=TRUE)[[1]][1]) {
    path.list <- unlist(strsplit(x='id_array', split=','))
    i = 1
    while(i<length(path.list)+1){
      if(is.null(getElement(json.list, path.list[i]))){
         json.list <- json.list[[1]]
      }else{
      json.list <- getElement(json.list, path.list[i])
      i <- i+1
      }
    }
  }
# From list to dataframe via unlisted json
i <-1
filled <- data.frame()
while(i < length(json.list)+ 1){
  unlisted.json <- unlist(json.list[[i]])
  to.fill <- data.frame(t(as.data.frame(unlisted.json, row.names = names(unlisted.json))),
stringsAsFactors=FALSE)
  filled <- rbind.fill(filled,to.fill)
  i <-  1 + i
}
# Export to SPSS Modeler Data
modelerData <- filled
print(modelerData)
modelerDataModel <- getMetaData(modelerData)
print(modelerDataModel)

"""

node.setPropertyValue("r_syntax", R_script)
```

Tabela 50. propriedades extensionimportnode

propriedades extensionimportnode	**Tipo de dados**	**Descrição da propriedade**
syntax_type	*R* *Python*	Especifique qual script é executado - R ou Python (R é o padrão).
r_syntax	*sequência*	A sintaxe de script do R a ser executada.
python_syntax	*sequência*	A sintaxe de script Python a ser executada.

Propriedades de fixedfilenode

O nó Arquivo Fixo importa dados de arquivos de texto de campo fixo, ou seja, arquivos cujos campos não são delimitados, mas iniciam na mesma posição e têm um comprimento fixo. Dados gerados por máquina ou legados são frequentemente armazenados em formato de campo fixo.

Exemplo

```
node = stream.create("fixedfile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("record_len", 32)
node.setPropertyValue("skip_header", 1)
node.setPropertyValue("fields", [["Age", 1, 3], ["Sex", 5, 7], ["BP", 9,
10], ["Cholesterol",
 12, 22], ["Na", 24, 25], ["K", 27, 27], ["Drug", 29, 32]])
node.setPropertyValue("decimal_symbol", "Period")
node.setPropertyValue("lines_to_scan", 30)
```

Tabela 51. Propriedades de fixedfilenode

propriedades fixedfilenode	**Tipo de dados**	**Descrição da propriedade**
record_len	*number*	Especifica o número de caracteres em cada registro.
line_oriented	*sinalização*	Ignora o caractere de nova linha no término de cada registro.
decimal_symbol	Default Comma Period	O tipo de separador decimal utilizado em sua origem de dados.
skip_header	*number*	Especifica o número de linhas a serem ignoradas no início do primeiro registro. Útil para ignorar cabeçalhos da coluna.
auto_recognize_datetime	*sinalização*	Especifica se datas ou horas são identificadas automaticamente nos dados de origem.
lines_to_scan	*number*	
fields	*lista*	Propriedade estruturada.
full_filename	*sequência*	Nome completo do arquivo a ser lido, incluindo o diretório.
strip_spaces	None Left Right Both	Descarta espaços à direita e à esquerda nas sequências na importação.
invalid_char_mode	Discard Replace	Remove caracteres inválidos (nulo, 0 ou qualquer caractere inexistente na codificação atual) da entrada de dados ou substitui caracteres inválidos pelo símbolo do caractere um especificado.
invalid_char_replacement	*sequência*	
use_custom_values	*sinalização*	

Tabela 51. Propriedades de fixedfilenode (continuação)		
propriedades `fixedfilenode`	**Tipo de dados**	**Descrição da propriedade**

`custom_storage`	`Unknown`
	
	`String`
	
	`Integer`
	
	`Real`
	
	`Time`
	
	`Date`
	
	`Timestamp`	

Tabela 51. Propriedades de fixedfilenode (continuação)

propriedades fixedfilenode	Tipo de dados	Descrição da propriedade
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	Esta propriedade se aplicará apenas se um armazenamento customizado tiver sido especificado.

Tabela 51. Propriedades de fixedfilenode (continuação)		
propriedades fixedfilenode	**Tipo de dados**	**Descrição da propriedade**
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Esta propriedade se aplicará apenas se um armazenamento customizado tiver sido especificado.

Tabela 51. Propriedades de fixedfilenode (continuação)		
propriedades fixedfilenode	**Tipo de dados**	**Descrição da propriedade**
custom_decimal_symbol	*campo*	Aplicável apenas se um armazenamento customizado tiver sido especificado.
encoding	StreamDefault SystemDefault "UTF-8"	Especifica o método de codificação de texto.

Propriedades do Nó gsdata_import

Utilize o nó de origem Geoespacial para exibir dados de mapa ou espaciais na sessão de mineração de dados.

Tabela 52. Propriedades do nó gsdata_import		
Propriedades do nó gsdata_import	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	Insere o caminho de arquivo para o arquivo .shp que deseja carregar.
map_service_URL	*sequência*	Insere a URL de serviço de mapa à qual conectar-se.
map_name	*sequência*	Apenas se map_service_URL for usado, isso contém a estrutura da pasta de nível superior do serviço de mapa.

Propriedades jsonimportnode

O nó de origem JSON importa dados de um arquivo JSON.

Tabela 53. Propriedades jsonimportnode		
propriedades jsonimportnode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	O nome do arquivo completo, incluindo o caminho.
string_format	*registros* *valores*	Especifique o formato da sequência JSON. O padrão é records.
auto_label		Incluído na versão 18.2.1.1..

Propriedades de sasimportnode

O nó Importação SAS importa dados do SAS no IBM SPSS Modeler.

Exemplo

```
node = stream.create("sasimport", "My node")
node.setPropertyValue("format", "Windows")
node.setPropertyValue("full_filename", "C:/data/retail.sas7bdat")
node.setPropertyValue("member_name", "Test")
node.setPropertyValue("read_formats", False)
node.setPropertyValue("full_format_filename", "Test")
node.setPropertyValue("import_names", True)
```

Tabela 54. Propriedades de sasimportnode

propriedades sasimportnode	**Tipo de dados**	**Descrição da propriedade**
format	Windows UNIX Transport SAS7 SAS8 SAS9	O formato do arquivo a ser importado.
full_filename	*sequência*	O nome completo do arquivo que você inserir, incluindo o caminho.
member_name	*sequência*	Especifica o membro a ser importado a partir do arquivo de transporte SAS especificado.
read_formats	*sinalização*	Lê os formatos de dados (como rótulos de variáveis) a partir do arquivo de formato especificado.
full_format_filename	*sequência*	
import_names	NamesAndLabels LabelsasNames	Especifica o método para o mapeamento de nomes de variáveis e de rótulos na importação.

Propriedades de simgennode

O nó Gerar Simulação fornece uma maneira fácil de gerar dados simulados, seja desde o início utilizando distribuições de estatísticas especificadas pelo usuário ou automaticamente utilizando as distribuições obtidas da execução de um nó Ajuste de Simulação em dados históricos existentes. Isso é útil quando se deseja avaliar o resultado de um modelo preditivo na presença de incerteza nos insumos do modelo.

Tabela 55. Propriedades de simgennode		
propriedades simgennode	**Tipo de dados**	**Descrição da propriedade**
fields	Propriedade estruturada	Ver exemplo
correlations	Propriedade estruturada	Ver exemplo
keep_min_max_setting	*Booleano*	
refit_correlations	*Booleano*	
max_cases	*Número inteiro*	O valor mínimo é 1000 e o valor máximo é 2.147.483.647
create_iteration_field	*Booleano*	
iteration_field_name	*sequência*	
replicate_results	*Booleano*	
random_seed	*Número inteiro*	
parameter_xml	*sequência*	Retorna o parâmetro Xml como uma sequência

Exemplo de campos

Este é um parâmetro de slot estruturado com a sintaxe a seguir:

```
simgennode.setPropertyValue("fields", [
    [field1, storage, locked, [distribution1], min, max],
    [field2, storage, locked, [distribution2], min, max],
    [field3, storage, locked, [distribution3], min, max]
])
```

distribution é uma declaração do nome de distribuição seguida de uma lista que contém pares de nomes de atributos e valores. Cada distribuição é definida da seguinte maneira:

```
[distributionname, [[par1], [par2], [par3]]]

simgennode = modeler.script.stream().createAt("simgen", u"Sim Gen", 726, 322)
simgennode.setPropertyValue("fields", [["Age", "integer", False, ["Uniform",[["min","1"],
["max","2"]]], "", ""]])
```

Por exemplo, para criar um nó que gera um campo único com uma distribuição binomial, é possível utilizar o script a seguir:

```
simgen_node1 = modeler.script.stream().createAt("simgen", u"Sim Gen", 200, 200)
simgen_node1.setPropertyValue("fields", [["Education", "Real", False, ["Binomial", [["n", 32],
 ["prob", 0.7]]], "", ""]])
```

A Distribuição binomial leva dois parâmetros: n e prob. Como o binomial não suporta valores mínimos e máximos, eles são fornecidos como uma sequência vazia.

Nota: Não é possível configurar o distribution diretamente porque ele é usado junto com a propriedade fields.

Os exemplos a seguir mostram todos os tipos de distribuição possíveis. Observe que o limite é inserido como thresh em ambos NegativeBinomialFailures e NegativeBinomialTrial.

```
stream = modeler.script.stream()

simgennode = stream.createAt("simgen", u"Sim Gen", 200, 200)

beta_dist = ["Field1", "Real", False, ["Beta",[["shape1","1"],["shape2","2"]]], "", ""]
binomial_dist = ["Field2", "Real", False, ["Binomial",[["n" ,"1"],["prob","1"]]], "", ""]
categorical_dist = ["Field3", "String", False, ["Categorical", [["A",0.3],["B",0.5],["C",0.2]]], "", ""]
```

```
dice_dist = ["Field4", "Real", False, ["Dice", [["1" ,"0.5"],["2","0.5"]]], "", ""]
exponential_dist = ["Field5", "Real", False, ["Exponential", [["scale","1"]]], "", ""]
fixed_dist = ["Field6", "Real", False, ["Fixed", [["value","1" ]]], "", ""]
gamma_dist = ["Field7", "Real", False, ["Gamma", [["scale","1"],["shape"," 1"]]], "", ""]
lognormal_dist = ["Field8", "Real", False, ["Lognormal", [["a","1"],["b","1" ]]], "", ""]
negbinomialfailures_dist = ["Field9", "Real", False, ["NegativeBinomialFailures",[["prob","0.5"],["thresh","1"]]], "", ""]
negbinomialtrial_dist = ["Field10", "Real", False, ["NegativeBinomialTrials",[["prob","0.2"],["thresh","1"]]], "", ""]
normal_dist = ["Field11", "Real", False, ["Normal", [["mean","1"] ,["stddev","2"]]], "", ""]
poisson_dist = ["Field12", "Real", False, ["Poisson", [["mean","1"]]], "", ""]
range_dist = ["Field13", "Real", False, ["Range", [["BEGIN","[1,3]"] ,["END","[2,4]"],["PROB","[[0.5],[0.5]]"]]], "", ""]
triangular_dist = ["Field14", "Real", False, ["Triangular", [["min","0"],["max","1"],["mode","1"]]], "", ""]
uniform_dist = ["Field15", "Real", False, ["Uniform", [["min","1"],["max","2"]]], "", ""]
weibull_dist = ["Field16", "Real", False, ["Weibull", [["a","0"],["b","1 "],["c","1"]]], "", ""]
```

```
simgennode.setPropertyValue("fields", [\
beta_dist, \
binomial_dist, \
categorical_dist, \
dice_dist, \
exponential_dist, \
fixed_dist, \
gamma_dist, \
lognormal_dist, \
negbinomialfailures_dist, \
negbinomialtrial_dist, \
normal_dist, \
poisson_dist, \
range_dist, \
triangular_dist, \
uniform_dist, \
weibull_dist
])
```

exemplo de correlações

Este é um parâmetro de slot estruturado com a sintaxe a seguir:

```
simgennode.setPropertyValue("correlations", [
    [field1, field2, correlation],
    [field1, field3, correlation],
    [field2, field3, correlation]
])
```

A correlação pode ser qualquer número entre +1 e -1. É possível especificar quantas correlações desejar. As correlações não especificadas são configuradas para zero. Se algum campo for desconhecido, o valor da correlação deverá ser configurado na matriz de correlação (ou tabela) e é mostrado em texto vermelho. Quando houver campos desconhecidos, não será possível executar o nó.

Propriedades de statisticsimportnode

O nó do arquivo IBM SPSS Statistics lê dados do formato de arquivo *.sav* usado pelo IBM SPSS Statistics, bem como arquivos de cache salvos em IBM SPSS Modeler que também utilizam o mesmo formato.

As propriedades desse nó são descritas em "Propriedades de statisticsimportnode" na página 431.

Propriedades do Nó tm1odataimport

O nó de origem do IBM Cognos TM1 importa dados a partir de bancos de dados do Cognos TM1.

Tabela 56. Propriedades do nó tm1odataimport

tm1odataimport propriedades do nó	**Tipo de dados**	**Descrição da propriedade**
credential_type	*inputCredential* ou *storedCredential*	Usado para indicar o tipo de credencial

Tabela 56. Propriedades do nó tm1odataimport (continuação)		
tm1odataimport propriedades do nó	**Tipo de dados**	**Descrição da propriedade**
input_credential	*lista*	Quando o credential_type for inputCredential; especifique o domínio, nome de usuário e senha.
stored_credential_name	*sequência*	Quando o credential_type for storedCredential; especifique o nome da credencial no servidor C & DS
selected_view	*["field" "field"]*	Uma propriedade de lista contendo os detalhes do cubo do TM1 selecionado e o nome da visualização de cubo a partir do qual os dados serão importados no SPSS. Por exemplo: TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])
is_private_view	*sinalização*	Especifica se o selected_view é uma visualização privada O valor padrão é false.
selected_columns	*["field"]*	Especifique a coluna selecionada; apenas um item pode ser especificado Por exemplo: setPropertyValue("selected_columns", ["Measures"])
selected_rows	*["field" "field"]*	Especifique as linhas selecionadas. Por exemplo: setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"]).
connection_type	AdminServer TM1Server	Indica o tipo de conexão. O padrão é AdminServer.
admin_host	*sequência*	A URL para o nome do host da API REST. Necessário se o connection_type for AdminServer
server_name	*sequência*	O nome do servidor TM1 selecionado a partir do admin_host. Necessário se o connection_type for AdminServer
server_url	*sequência*	A URL para a API de REST do TM1 Server Necessário se o connection_type for TM1Server

tm1import Propriedades do nó (descontinuado)

O nó de origem do IBM Cognos TM1 importa dados a partir de bancos de dados do Cognos TM1.

Nota: Esse nó foi descontinuado no Modeler 18.0 O nome do script do nó de substituição é *tm1odataimport*

Tabela 57. Propriedades do nó tm1import		
Propriedades do nó `tm1import`	**Tipo de dados**	**Descrição da propriedade**
`pm_host`	*sequência*	**Nota:** Somente para as versões 16.0 e 17.0 O nome do host. Por exemplo: `TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
`tm1_connection`	*["campo", "campo" ..., "campo"]*	**Nota:** Somente para as versões 16.0 e 17.0 Uma propriedade de lista que contém os detalhes da conexão com o servidor TM1. O formato é: `[ "TM1_Server_Name","tm1_username","tm1_ password"]` Por exemplo: `TM1_import.setPropertyValue("tm1_connection", ['Planning Sample', "admin", "apple"])`
`selected_view`	*["field" "field"]*	Uma propriedade de lista contendo os detalhes do cubo do TM1 selecionado e o nome da visualização de cubo a partir do qual os dados serão importados no SPSS. Por exemplo: `TM1_import.setPropertyValue("selected_view", ['plan_BudgetPlan', 'Goal Input'])`
`selected_column`	*["field"]*	Especifique a coluna selecionada; apenas um item pode ser especificado Por exemplo: `setPropertyValue("selected_columns", ["Measures"])`
`selected_rows`	*["field" "field"]*	Especifique as linhas selecionadas Por exemplo: `setPropertyValue("selected_rows", ["Dimension_1_1", "Dimension_2_1", "Dimension_3_1", "Periods"])`

propriedades do nó twcimport

O nó de origem do TWC importa dados do clima da The Weather Company, uma Empresa IBM. É possível usá-lo para obter dados meteorológicos históricos ou de previsão de um local Isso pode ajudá-lo a desenvolver soluções de negócios orientadas por clima para uma melhor tomada de decisão usando os dados meteorológicos mais precisos e precisos disponíveis.

Tabela 58. propriedades do nó twcimport		
twcimport propriedades do nó	**Tipo de dados**	**Descrição da propriedade**
TWCDataImport.latitude	*Real*	Especifica um valor de latitude no formato [-90.0~90.0] .
TWCDataImport.longitude	*Real*	Especifica um valor de longitude no formato [-180.0~180.0].
TWCDataImport.licenseKey	*sequência*	Especifica a chave de licença obtida da The Weather Company.
TWCDataImport.measurmentUnit	English Metric Hybrid	Especifica a unidade de medida. Os valores possíveis são English, Metric ou Hybrid. Metric é o padrão..
TWCDataImport.dataType	Historical Forecast	Especifica o tipo de dados de clima a serem inseridos Os valores possíveis são Historical ou Forecast. Historical é o padrão..
TWCDataImport.startDate	*Número inteiro*	Se Historical for especificado para TWCDataImport.dataType, especifique uma data de início no formato yyyyMMdd
TWCDataImport.endDate	*Número inteiro*	Se Historical for especificado para TWCDataImport.dataType, especifique uma data de encerramento no formato yyyyMMdd
TWCDataImport.forecastHour	6 12 24 48	Se Forecast for especificado para TWCDataImport.dataType, especifique 6, 12, 24ou 48 para a hora..

Propriedades de userinputnode

O nó Entrada do Usuário fornece uma maneira fácil de criar dados sintéticos, seja desde o início ou alterando dados existentes. Isso é útil, por exemplo, quando desejar criar um conjunto de dados de teste para modelagem.

Exemplo

```
node = stream.create("userinput", "My node")
node.setPropertyValue("names", ["test1", "test2"])
node.setKeyedPropertyValue("data", "test1", "2, 4, 8")
node.setKeyedPropertyValue("custom_storage", "test1", "Integer")
node.setPropertyValue("data_mode", "Ordered")
```

Tabela 59. Propriedades de userinputnode		
propriedades userinputnode	**Tipo de dados**	**Descrição da propriedade**
data		
names		Slot estruturado que configura ou retorna uma lista de nomes de campo gerados pelo nó.
custom_storage	Unknown String Integer Real Time Date Timestamp	Slot chaveado que configura ou retorna o armazenamento para um campo.
data_mode	Combined Ordered	Se Combined for especificado, serão gerados registros para cada combinação de valores configurados e valores mín/máx. O número de registros gerados é igual ao produto do número de valores em cada campo. Se Ordered for especificado, um valor será tirado de cada coluna para cada registro a fim de gerar uma linha de dados. O número de registros gerados é igual ao número maior de valores associados a um campo. Quaisquer campos com menos valores de dados serão preenchidos com valores nulos.
values		**Nota:** Essa propriedade foi descontinuada a favor de userinputnode.data e não deve mais ser utilizada.

Propriedades de variablefilenode

O nó Arquivo Variável lê arquivos de texto de campo livre, ou seja, arquivos cujos registros contêm um número constante dos campos, mas um número variado de caracteres. Esse nó também é útil para arquivos com texto de cabeçalho de comprimento fixo e certos tipos de anotações.

Exemplo

```
node = stream.create("variablefile", "My node")
node.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node.setPropertyValue("read_field_names", True)
node.setPropertyValue("delimit_other", True)
node.setPropertyValue("other", ",")
```

```
node.setPropertyValue("quotes_1", "Discard")
node.setPropertyValue("decimal_symbol", "Comma")
node.setPropertyValue("invalid_char_mode", "Replace")
node.setPropertyValue("invalid_char_replacement", "|")
node.setKeyedPropertyValue("use_custom_values", "Age", True)
node.setKeyedPropertyValue("direction", "Age", "Input")
node.setKeyedPropertyValue("type", "Age", "Range")
node.setKeyedPropertyValue("values", "Age", [1, 100])
```

Tabela 60. Propriedades de variablefilenode

propriedades variablefilenode	**Tipo de dados**	**Descrição da propriedade**
skip_header	*number*	Especifica o número de caracteres a serem ignorados no início do primeiro registro.
num_fields_auto	*sinalização*	Determina o número de campos em cada registro automaticamente. Os registros devem ser finalizados com um caractere de nova linha.
num_fields	*number*	Especifica manualmente o número de campos em cada registro.
delimit_space	*sinalização*	Especifica o caractere utilizado para delimitar limites de campo no arquivo.
delimit_tab	*sinalização*	
delimit_new_line	*sinalização*	
delimit_non_printing	*sinalização*	
delimit_comma	*sinalização*	Nos casos em que a vírgula é o delimitador de campo e também o separador decimal dos fluxos, configure delimit_other para *true* e especifique uma vírgula como o delimitador utilizando a propriedade other.
delimit_other	*sinalização*	Permite especificar um delimitador customizado utilizando a propriedade other.
other	*sequência*	Especifica o delimitador utilizado quando delimit_other é *true*.
decimal_symbol	Default Comma Period	Especifica o separador decimal utilizado na origem de dados.
multi_blank	*sinalização*	Trata diversos caracteres delimitadores em branco adjacentes como um delimitador único.
read_field_names	*sinalização*	Trata a primeira linha no arquivo de dados como rótulos para a coluna.

Tabela 60. Propriedades de variablefilenode (continuação)

propriedades variablefilenode	**Tipo de dados**	**Descrição da propriedade**
strip_spaces	None Left Right Both	Descarta espaços à direita e à esquerda nas sequências na importação.
invalid_char_mode	Discard Replace	Remove caracteres inválidos (nulo, 0 ou qualquer caractere inexistente na codificação atual) da entrada de dados ou substitui caracteres inválidos pelo símbolo do caractere um especificado.
invalid_char_replacement	*sequência*	
break_case_by_newline	*sinalização*	Especifica que o delimitador de linha é o caractere de nova linha.
lines_to_scan	*number*	Especifica quantas linhas devem ser varridas para tipos de dados especificados.
auto_recognize_datetime	*sinalização*	Especifica se datas ou horas são identificadas automaticamente nos dados de origem.
quotes_1	Discard PairAndDiscard IncludeAsText	Especifica como as aspas simples são tratadas na importação.
quotes_2	Discard PairAndDiscard IncludeAsText	Especifica como as aspas duplas são tratadas na importação.
full_filename	*sequência*	Nome completo do arquivo a ser lido, incluindo o diretório.
use_custom_values	*sinalização*	

Tabela 60. Propriedades de variablefilenode (continuação)

propriedades variablefilenode	**Tipo de dados**	**Descrição da propriedade**
custom_storage	Unknown String Integer Real Time Date Timestamp	

Tabela 60. Propriedades de variablefilenode (continuação)

propriedades variablefilenode	**Tipo de dados**	**Descrição da propriedade**
custom_date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY"	Aplicável apenas se um armazenamento customizado tiver sido especificado.

Tabela 60. Propriedades de variablefilenode (continuação)

propriedades variablefilenode	**Tipo de dados**	**Descrição da propriedade**
	"DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	

Tabela 60. Propriedades de variablefilenode (continuação)		
propriedades variablefilenode	**Tipo de dados**	**Descrição da propriedade**
custom_time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Aplicável apenas se um armazenamento customizado tiver sido especificado.
custom_decimal_symbol	*campo*	Aplicável apenas se um armazenamento customizado tiver sido especificado.
encoding	StreamDefault SystemDefault "UTF-8"	Especifica o método de codificação de texto.

Propriedades de xmlimportnode

O nó de origem XML importa dados no formato XML para o fluxo. É possível importar um único arquivo ou todos os arquivos em um diretório. É possível, opcionalmente, especificar um arquivo de esquema a partir do qual a estrutura XML é lida.

Exemplo

```
node = stream.create("xmlimport", "My node")
node.setPropertyValue("full_filename", "c:/import/ebooks.xml")
node.setPropertyValue("records", "/author/name")
```

Tabela 61. Propriedades de xmlimportnode

propriedades xmlimportnode	**Tipo de dados**	**Descrição da propriedade**
read	single directory	Lê um arquivo de dados único (padrão) ou todos os arquivos XML em um diretório.
recurse	*sinalização*	Especifica se deve também ler arquivos XML a partir de todos os subdiretórios do diretório especificado.
full_filename	*sequência*	(obrigatório) Caminho completo e nome do arquivo XML a ser importado (se read = single).
directory_name	*sequência*	(obrigatório) Caminho completo e nome do diretório do qual importar arquivos XML (se read = directory).
full_schema_filename	*sequência*	Caminho e nome de arquivo completos do arquivo XSD ou DTD a partir do qual ler a estrutura XML. Se você omitir esse parâmetro, a estrutura será lida a partir do arquivo de origem XML.
records	*sequência*	Expressão XPath (por exemplo, /author/name) para definir o limite do registro. Um novo registro será criado toda vez que este elemento for encontrado.
mode	read specify	Lê todos os dados (padrão) ou especifica quais itens serão lidos.
fields		Lista de itens (elementos e atributos) a serem importados. Cada item na lista é uma expressão XPath.

Capítulo 10. Propriedades do Nó de Operações de Registro

Propriedades appendnode

O nó Anexar concatena conjuntos de registros. Ele é útil para combinar conjuntos de dados com estruturas semelhantes, porém dados diferentes.

Exemplo

```
node = stream.create("append", "My node")
node.setPropertyValue("match_by", "Name")
node.setPropertyValue("match_case", True)
node.setPropertyValue("include_fields_from", "All")
node.setPropertyValue("create_tag_field", True)
node.setPropertyValue("tag_field_name", "Append_Flag")
```

Tabela 62. Propriedades appendnode

propriedades appendnode	Tipo de dados	Descrição da propriedade
match_by	Position Name	É possível anexar conjuntos de dados com base na posição dos campos na origem de dados principal ou no nome dos campos nos conjuntos de dados de entrada.
match_case	*sinalização*	Ativa a sensibilidade de maiúsculas e minúsculas ao corresponder nomes de campo.
include_fields_from	Main All	
create_tag_field	*sinalização*	
tag_field_name	*sequência*	

Propriedades de aggregatenode

O nó Agregado substitui uma sequência de registros de entrada por registros de saída resumidos e agregados.

Exemplo

```
node = stream.create("aggregate", "My node")
# dbnode is a configured database import node
stream.link(dbnode, node)
node.setPropertyValue("contiguous", True)
node.setPropertyValue("keys", ["Drug"])
node.setKeyedPropertyValue("aggregates", "Age", ["Sum", "Mean"])
node.setPropertyValue("inc_record_count", True)
```

```
node.setPropertyValue("count_field", "index")
node.setPropertyValue("extension", "Aggregated_")
node.setPropertyValue("add_as", "Prefix")
```

Tabela 63. Propriedades de aggregatenode

propriedades aggregatenode	**Tipo de dados**	**Descrição da propriedade**
keys	*lista*	Lista campos que podem ser utilizados como chaves para agregação. Por exemplo, se Sex e Region forem seus campos-chave, cada combinação exclusiva de M e F com regiões N e S (quatro combinações exclusivas) terá um registro agregado.
contiguous	*sinalização*	Selecione essa opção se você souber que todos os registros com os mesmos valores da chave são agrupados na entrada (por exemplo, se a entrada for classificada nos campos-chave). Fazer isso poderá melhorar o desempenho.
aggregates		Propriedade estruturada que lista os campos numéricos cujos valores serão agregados, bem como os modos de agregação selecionados.
aggregate_exprs		Propriedade definida como chave que define uma chave para o nome de campo derivado com a expressão agregada usada para calcular essa chave. Por exemplo: `aggregatenode.setKeyedPropertyValue ("aggregate_exprs", "Na_MAX", "MAX('Na')")`
extension	*sequência*	Especifica um prefixo ou sufixo para duplicar campos agregados (amostra abaixo).
add_as	Suffix Prefix	
inc_record_count	*sinalização*	Cria um campo extra que especifica quantos registros de entrada foram agregados para formar cada registro agregado.
count_field	*sequência*	Especifica o nome do campo de contagem de registros.
allow_approximation	*Booleano*	Permite aproximação de estatísticas de ordem quando a agregação é executada em Servidor Analítico
bin_count	*Número inteiro*	Especifica o número de categorias a serem utilizadas na aproximação

propriedades balancenode

O nó Balanceamento corrige desbalanceamentos em um conjunto de dados, para que ele esteja em conformidade com uma condição especificada. A diretiva de balanceamento ajusta a proporção de registros em que uma condição é verdadeira pelo fator especificado.

Exemplo

```
node = stream.create("balance", "My node")
node.setPropertyValue("training_data_only", True)
node.setPropertyValue("directives", [[1.3, "Age > 60"], [1.5, "Na > 0.5"]])
```

Tabela 64. propriedades balancenode

propriedades balancenode	Tipo de dados	Descrição da propriedade
directives		Propriedade estruturada para balancear a proporção dos valores de campo com base em um número especificado (consulte o exemplo a seguir).
training_data_only	*sinalização*	Especifica que apenas os dados de treinamento devem ser balanceados. Se nenhum campo de partição estiver presente no fluxo, essa opção será ignorada.

Esta propriedade do nó usa o formato:

[[*number, string*] \ [*number, string*] \ ... [*number, string*]].

Nota: Se sequências (usando aspas duplas) estiverem integradas na expressão, elas deverão ser precedidas pelo caractere de escape " \ ". O caractere " \ " também é o caractere de continuação de linha, que pode ser usado para alinhar os argumentos para clareza.

propriedades cplexoptnode

O nó de Otimização do CPLEX fornece a capacidade de usar otimização baseada em matemática complexa (CPLEX) via arquivo de modelo de Programação de Programação de Otimização (OPL). Essa funcionalidade estava disponível no produto IBM Analytical Decision Management , que não é mais suportado. Mas você também pode usar o nó CPLEX em SPSS Modeler sem precisar de IBM Analytical Decision Management.

Tabela 65. propriedades cplexoptnode

propriedades cplexoptnode	Tipo de dados	Descrição da propriedade
opl_model_text	*sequência*	O programa de script OPL (Optimization Programming Language) que o nó do CPLEX Optimization executará e, em seguida, gerará o resultado de otimização.

Tabela 65. propriedades cplexoptnode (continuação)

propriedades cplexoptnode	**Tipo de dados**	**Descrição da propriedade**
opl_tuple_set_name	*sequência*	O nome do conjunto de tuplas no modelo OPL que corresponde aos dados recebidos. Isso não é necessário e normalmente não é configurado por meio do script. Só deve ser usado para edição de mapeamentos de campo de uma origem de dados selecionada.
data_input_map	*Lista de propriedades estruturadas*	Os mapeamentos de campo de entrada para uma origem de dados. Isso não é necessário e normalmente não é configurado por meio do script. Só deve ser usado para edição de mapeamentos de campo de uma origem de dados selecionada.

Tabela 65. propriedades cplexoptnode (continuação)		
propriedades cplexoptnode	**Tipo de dados**	**Descrição da propriedade**
`md_data_input_map`	*Lista de propriedades estruturadas*	Os mapeamentos de campo entre cada tupla definida no OPL, com cada origem de dados de campo correspondente (dados recebidos). Os usuários podem editá-los cada um individualmente conforme a origem de dados. Com esse script, é possível configurar a propriedade diretamente para configurar todos os mapeamentos de uma só vez. Essa configuração não é mostrada na interface com o usuário Cada entidade na lista são dados estruturados: **Tag Origem de dados.** A tag da origem de dados, que pode ser localizada no menu suspenso da origem. Por exemplo, para `0_Products_Type`, a tag é `0`. **Índice Origem de dados.** A sequência física (índice) da origem de dados. Isso é determinado pela ordem de conexão. **Nó de Origem** O nó de origem (anotação) da origem de dados. Isso pode ser localizado na lista suspensa da origem de dados Por exemplo, para `0_Products_Type`, o nó de origem é `Products`. **Nó conectado.** O nó anterior (anotação) que conecta o nó de otimização atual do CPLEX. Isso pode ser localizado na lista suspensa da origem de dados Por exemplo, para `0_Products_Type`, o nó conectado é `Type`. **Nome do conjunto de tuplas.** O nome do conjunto de tuplas da origem de dados. Ele deve corresponder ao que está definido na OPL. **Nome do campo de tupla.** O nome do campo do conjunto de tuplas da origem de dados. Ele deve corresponder ao que está definido na definição do conjunto de tuplas da OPL. **Tipo de armazenamento.** O tipo de armazenamento campo. Os valores possíveis são `int`, `float` ou `string`.

Tabela 65. propriedades cplexoptnode (continuação)

propriedades cplexoptnode	Tipo de dados	Descrição da propriedade
		Nome do campo de dados. O nome de campo da origem de dados. **Exemplo:** `[[0,0,'Product','Type','Products','prod_id_tup','int','prod_id'], [0,0,'Product','Type','Products','prod_name_tup','string', 'prod_name'], [1,1,'Components','Type','Components', 'comp_id_tup','int','comp_id'], [1,1,'Components','Type', 'Components','comp_name_tup','string','comp_name']]`
opl_data_text	*sequência*	A definição de algumas variáveis ou dados usados para a OPL.
output_value_mode	*sequência*	Os valores possíveis são `raw` ou `dvar`. Se `dvar` for especificado, na guia Saída, o usuário deverá especificar o nome da variável de função do objeto na OPL para a saída. Se `raw` for especificado, a função objetiva será gerada diretamente, independentemente do nome.
decision_variable_name	*sequência*	O nome da variável de função objetiva definida na OPL. Isso é ativado apenas quando a propriedade `output_value_mode` é definida como `dvar`.
objective_function_value_fieldname	*sequência*	O nome do campo para o valor da função objetiva a ser usado na saída. O padrão é `_OBJECTIVE`.
output_tuple_set_names	*sequência*	O nome das tuplas predefinidas dos dados recebidos. Ele age como os índices da variável de decisão e deve ser gerado com as Saídas de variável. A Tupla de saída deve ser consistente com a definição de variável de decisão na OPL. Se houver vários índices, os nomes de tuplas deverão ser unidos por uma vírgula (`,`). Um exemplo para uma única tupla é `Products`, com a definição de OPL correspondente sendo `dvar float+ Production[Products];` Um exemplo para várias tuplas é `Products,Components`, com a definição de OPL correspondente sendo `dvar float+ Production[Products][Components];`

Tabela 65. propriedades cplexoptnode (continuação)		
propriedades cplexoptnode	**Tipo de dados**	**Descrição da propriedade**
decision_output_map	*Lista de propriedades estruturadas*	O mapeamento de campo entre variáveis definidas na OPL que será gerado e os campos de saída. Cada entidade na lista são dados estruturados: **Nome da variável.** O nome da variável na OPL a ser gerado. **Tipo de armazenamento.** Os valores possíveis são `int`, `float` ou `string`. **Nome do campo de saída.** O nome de campo esperado nos resultados (saída ou exportação). **Exemplo:** `[['Production','int','res'], ['Remark','string','res_1']['Cost', 'float','res_2']]`

Propriedades derive_stbnode

O nó Space-Time-Boxes deriva Space-Time-Boxes a partir de campos de latitude, longitude e de registro de data e hora. Também é possível identificar Space-Time-Boxes frequentes como hangouts.

Exemplo

```
node = modeler.script.stream().createAt("derive_stb", "My node", 96, 96)

# Individual Records mode
node.setPropertyValue("mode", "IndividualRecords")
node.setPropertyValue("latitude_field", "Latitude")
node.setPropertyValue("longitude_field", "Longitude")
node.setPropertyValue("timestamp_field", "OccurredAt")
node.setPropertyValue("densities", ["STB_GH7_1HOUR", "STB_GH7_30MINS"])
node.setPropertyValue("add_extension_as", "Prefix")
node.setPropertyValue("name_extension", "stb_")

# Hangouts mode
node.setPropertyValue("mode", "Hangouts")
node.setPropertyValue("hangout_density", "STB_GH7_30MINS")
node.setPropertyValue("id_field", "Event")
node.setPropertyValue("qualifying_duration", "30MINUTES")
node.setPropertyValue("min_events", 4)
node.setPropertyValue("qualifying_pct", 65)
```

Tabela 66. Propriedades do nó Space-Time-Boxes		
propriedades derive_stbnode	**Tipo de dados**	**Descrição da propriedade**
mode	`IndividualRecords` `Hangouts`	
latitude_field	*campo*	

Tabela 66. Propriedades do nó Space-Time-Boxes (continuação)		
propriedades derive_stbnode	**Tipo de dados**	**Descrição da propriedade**
longitude_field	*campo*	
timestamp_field	*campo*	
hangout_density	*density*	Uma única densidade. Veja densities para valores de densidade válidos.
densities	[*density*,*density*,..., *density*]	Cada densidade é uma sequência, por exemplo, STB_GH8_1DAY. **Nota:** Há limites para os quais as densidades são válidas. Para valores geohash, os valores de GH1 a GH15 podem ser utilizados. Para a parte temporal, os valores a seguir podem ser utilizados: EVER 1YEAR 1MONTH 1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2HOURS 1HOUR 30MINS 15MINS 10MINS 5MINS 2MINS 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SEC
id_field	*campo*	
qualifying_duration	1DAY 12HOURS 8HOURS 6HOURS 4HOURS 3HOURS 2Hours 1HOUR 30MIN 15MIN 10MIN 5MIN 2MIN 1MIN 30SECS 15SECS 10SECS 5SECS 2SECS 1SECS	Deve ser uma sequência.
min_events	*Número inteiro*	O valor de número inteiro válido mínimo é 2.
qualifying_pct	*Número inteiro*	Deve estar no intervalo de 1 a 100.

Tabela 66. Propriedades do nó Space-Time-Boxes (continuação)		
propriedades derive_stbnode	**Tipo de dados**	**Descrição da propriedade**
add_extension_as	Prefix Suffix	
name_extension	*sequência*	

Propriedades de distinctnode

O nó Distinto remove registros duplicados seja transmitindo o primeiro registro distinto para o fluxo de dados ou descartando o primeiro registro e transmitindo quaisquer duplicatas para o fluxo de dados.

Exemplo

```
node = stream.create("distinct", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("fields", ["Age" "Sex"])
node.setPropertyValue("keys_pre_sorted", True)
```

Tabela 67. Propriedades de distinctnode		
propriedades distinctnode	**Tipo de dados**	**Descrição da propriedade**
mode	Include Discard	É possível incluir o primeiro registro distinto no fluxo de dados ou descartar o primeiro registro distinto e transmitir quaisquer registros duplicados para o fluxo de dados.
grouping_fields	*lista*	Lista campos utilizados para determinar se os registros são idênticos. **Nota:** Esta propriedade é descontinuada a partir do IBM SPSS Modeler 16.
composite_value	Slot estruturado	Consulte o exemplo abaixo.
composite_values	Slot estruturado	Consulte o exemplo abaixo.
inc_record_count	*sinalização*	Cria um campo extra que especifica quantos registros de entrada foram agregados para formar cada registro agregado.
count_field	*sequência*	Especifica o nome do campo de contagem de registros.
sort_keys	Slot estruturado.	**Nota:** Esta propriedade é descontinuada a partir do IBM SPSS Modeler 16.
default_ascending	*sinalização*	
low_distinct_key_count	*sinalização*	Especifica que você tem apenas um pequeno número de registros e/ou um pequeno número de valores exclusivos de um ou mais campos-chave.

Tabela 67. Propriedades de distinctnode (continuação)		
propriedades distinctnode	**Tipo de dados**	**Descrição da propriedade**
keys_pre_sorted	*sinalização*	Especifica que todos os registros com os mesmos valores da chave são agrupados na entrada.
disable_sql_generation	*sinalização*	

Exemplo para propriedade composite_value

A propriedade composite_value tem a forma geral a seguir:

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION tem o formulário [FillType, Option1, Option2, ...].

Exemplos:

```
node.setKeyedPropertyValue("composite_value", "Age", ["First"])
node.setKeyedPropertyValue("composite_value", "Age", ["last"])
node.setKeyedPropertyValue("composite_value", "Age", ["Total"])
node.setKeyedPropertyValue("composite_value", "Age", ["Average"])
node.setKeyedPropertyValue("composite_value", "Age", ["Min"])
node.setKeyedPropertyValue("composite_value", "Age", ["Max"])
node.setKeyedPropertyValue("composite_value", "Date", ["Earliest"])
node.setKeyedPropertyValue("composite_value", "Date", ["Latest"])
node.setKeyedPropertyValue("composite_value", "Code", ["FirstAlpha"])
node.setKeyedPropertyValue("composite_value", "Code", ["LastAlpha"])
```

As opções customizadas requerem mais de um argumento que são incluídos como uma lista, por exemplo:

```
node.setKeyedPropertyValue("composite_value", "Name", ["MostFrequent", "FirstRecord"])
node.setKeyedPropertyValue("composite_value", "Date", ["LeastFrequent", "LastRecord"])
node.setKeyedPropertyValue("composite_value", "Pending", ["IncludesValue", "T", "F"])
node.setKeyedPropertyValue("composite_value", "Marital", ["FirstMatch", "Married", "Divorced",
"Separated"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Space"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "Comma"])
node.setKeyedPropertyValue("composite_value", "Code", ["Concatenate", "UnderScore"])
```

Exemplo para propriedade composite_values

A propriedade composite_values tem a forma geral a seguir:

```
node.setPropertyValue("composite_values", [
                      [FIELD1, [FILLOPTION1]],
                      [FIELD2, [FILLOPTION2]],
.
.
])
```

Exemplo:

```
node.setPropertyValue("composite_values", [
    ["Age", ["First"]],
    ["Name", ["MostFrequent", "First"]],
    ["Pending", ["IncludesValue", "T"]],
    ["Marital", ["FirstMatch", "Married", "Divorced", "Separated"]],
    ["Code", ["Concatenate", "Comma"]]
])
```

propriedades extensionprocessnode

Com o nó Transformação de extensão, é possível obter dados de um fluxo e aplicar transformações aos dados usando o script R ou o script Python para Spark.

Exemplo de Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_process", "extension_process")
node.setPropertyValue("syntax_type", "Python")

process_script = """
import spss.pyspark.runtime
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = StructType([StructField("Age", LongType(), nullable=True), \
                      StructField("Sex", StringType(), nullable=True), \
                      StructField("BP", StringType(), nullable=True), \
                      StructField("Na", DoubleType(), nullable=True), \
                      StructField("K", DoubleType(), nullable=True), \
                      StructField("Drug", StringType(), nullable=True)])
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()
    print df.dtypes[:]
    _newDF = df.select("Age","Sex","BP","Na","K","Drug")
    print _newDF.dtypes[:]
    cxt.setSparkOutputData(_newDF)
"""

node.setPropertyValue("python_syntax", process_script)
```

Exemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """day<-as.Date(modelerData$dob, format="%Y-%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabela 68. propriedades extensionprocessnode

propriedades extensionprocessnode	**Tipo de dados**	**Descrição da propriedade**
syntax_type	*R* *Python*	Especifique qual script é executado - R ou Python (R é o padrão).
r_syntax	*sequência*	A sintaxe de script do R a ser executada.
python_syntax	*sequência*	A sintaxe de script Python a ser executada.
use_batch_size	*sinalização*	Ative o uso do processamento em lote.

Tabela 68. propriedades extensionprocessnode (continuação)		
propriedades extensionprocessnode	**Tipo de dados**	**Descrição da propriedade**
batch_size	*Número inteiro*	Especifique o número de registros de dados para incluir em cada lote.
convert_flags	StringsAndDoubles LogicalValues	Opção para converter os campos de sinalização.
convert_missing	*sinalização*	Opção para converter valores ausentes para o valor NA do R.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	POSIXct POSIXlt	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.

Propriedades de mergenode

O nó Mesclagem seleciona diversos registros de entrada e cria um registro de saída único contendo alguns ou todos os campos de entrada. Ele é útil para mesclar dados de diferentes origens, como dados internos do cliente e dados demográficos comprados.

Exemplo

```
node = stream.create("merge", "My node")
# assume customerdata and salesdata are configured database import nodes
stream.link(customerdata, node)
stream.link(salesdata, node)
node.setPropertyValue("method", "Keys")
node.setPropertyValue("key_fields", ["id"])
node.setPropertyValue("common_keys", True)
node.setPropertyValue("join", "PartialOuter")
node.setKeyedPropertyValue("outer_join_tag", "2", True)
node.setKeyedPropertyValue("outer_join_tag", "4", True)
node.setPropertyValue("single_large_input", True)
node.setPropertyValue("single_large_input_tag", "2")
node.setPropertyValue("use_existing_sort_keys", True)
node.setPropertyValue("existing_sort_keys", [["id", "Ascending"]])
```

Tabela 69. Propriedades de mergenode

propriedades mergenode	**Tipo de dados**	**Descrição da propriedade**

`method`	`Order`
	
	`Keys`
	
	`Condition`
	
	`Rankedcondition`
		Especifique se os registros são mesclados na ordem em que eles são listados nos arquivos de dados, se um ou mais campos-chave serão utilizados para mesclar os registros com o mesmo valor nos campos-chave, se os registros serão mesclados se uma condição especificada for satisfeita ou se cada pareamento de linha nos conjuntos de dados primário e em todos os conjuntos secundários deve ser mesclado; utilizando a expressão de classificação para classificar diversas correspondências na ordem de baixo para cima.

`condition`	*sequência*	Se `method` for configurado como `Condition`, especifica a condição para inclusão ou descarte de registros.
`key_fields`	*lista*	
`common_keys`	*sinalização*	

`join`	`Inner`
	
	`FullOuter`
	
	`PartialOuter`
	
	`Anti`	

`outer_join_tag.n`	*sinalização*	Nesta propriedade, *n* é o nome da tag conforme exibido na caixa de diálogo Selecionar Conjunto de Dados. Observe que diversos nomes de tag podem ser especificados, já que qualquer número de conjuntos de dados pode contribuir com registros incompletos.
`single_large_input`	*sinalização*	Especifica se a otimização para ter uma entrada relativamente grande em comparação com as outras entradas será utilizada.
`single_large_input_tag`	*sequência*	Especifica o nome da tag conforme exibido na caixa de diálogo Selecionar Conjunto de Dados Grande. Observe que o uso desta propriedade difere ligeiramente da propriedade `outer_join_tag` (sinalização versus sequência) porque apenas um conjunto de dados de entrada pode ser especificado.
`use_existing_sort_keys`	*sinalização*	Especifica se as entradas já estão classificadas por um ou mais campos-chave.

Tabela 69. Propriedades de mergenode (continuação)		
propriedades mergenode	**Tipo de dados**	**Descrição da propriedade**
existing_sort_keys	[['string', 'Ascending'] \ ['string', 'Descending']]	Especifica os campos que já estiverem classificados e a direção na qual eles são classificados.
primary_dataset	*sequência*	Se method for Rankedcondition, selecione o conjunto de dados primários na mesclagem. Isso pode ser considerado como o lado esquerdo de uma mesclagem de junção externa.
rename_duplicate_fields	*Booleano*	Se method for Rankedcondition, e isso for configurado como Y, se o conjunto de dados mesclado resultante contiver vários campos com o mesmo nome de origens de dados diferentes, as respectivas tags das origens de dados serão incluídas no início dos cabeçalhos da coluna de campo.
merge_condition	*sequência*	
ranking_expression	*sequência*	
Num_matches	*Número inteiro*	O número de correspondências a serem retornadas, com base em merge_condition e ranking_expression. Mínimo 1, máximo 100.

Propriedades de rfmaggregatenode

O Nó de Recência, Frequência, Monetária (RFM) permite que você tire os dados transacionais históricos dos clientes, afaste quaisquer dados não utilizados e combine todos os seus dados de transações restantes em uma única linha que lista quando eles trataram pela última vez com você, quantas transações eles fizeram e o valor monetário total dessas transações.

Exemplo

```
node = stream.create("rfmaggregate", "My node")
node.setPropertyValue("relative_to", "Fixed")
node.setPropertyValue("reference_date", "2007-10-12")
node.setPropertyValue("id_field", "CardID")
node.setPropertyValue("date_field", "Date")
node.setPropertyValue("value_field", "Amount")
node.setPropertyValue("only_recent_transactions", True)
node.setPropertyValue("transaction_date_after", "2000-10-01")
```

Tabela 70. Propriedades de rfmaggregatenode		
propriedades rfmaggregatenode	**Tipo de dados**	**Descrição da propriedade**
relative_to	Fixed Today	Especifique a data a partir da qual a recência das transações será calculada.

Tabela 70. Propriedades de rfmaggregatenode (continuação)

propriedades rfmaggregatenode	**Tipo de dados**	**Descrição da propriedade**
reference_date	*Data*	Somente disponível se Fixed for escolhido em relative_to.
contiguous	*sinalização*	Se seus dados forem pré-classificados de forma que todos os registros com o mesmo ID apareçam juntos no fluxo de dados, selecionar esta opção acelera o processamento.
id_field	*campo*	Especifique o campo a ser utilizado para identificar o cliente e suas transações.
date_field	*campo*	Especifique o campo de data a ser utilizado para calcular a recência.
value_field	*campo*	Especifique o campo a ser utilizado para calcular o valor monetário.
extension	*sequência*	Especifique um prefixo ou sufixo para duplicar campos agregados.
add_as	Suffix Prefix	Especifique se o extension deve ser incluído como um sufixo ou um prefixo.
discard_low_value_records	*sinalização*	Ative o uso da configuração discard_records_below.
discard_records_below	*number*	Especifica um valor mínimo abaixo do qual quaisquer detalhes da transação não serão utilizados ao calcular os totais de RFM. As unidades de valor se relacionam com o campo value selecionado.
only_recent_transactions	*sinalização*	Ative o uso das configurações specify_transaction_date ou transaction_within_last.
specify_transaction_date	*sinalização*	
transaction_date_after	*Data*	Somente disponível se specify_transaction_date for selecionado. Especifique a data da transação após a qual os registros serão incluídos em sua análise.
transaction_within_last	*number*	Somente disponível se transaction_within_last for selecionado. Especifique o número e o tipo de períodos (dias, semanas, meses ou anos) desde Calcular Recência relativa até a data após a qual os registros serão incluídos em sua análise.

Tabela 70. Propriedades de rfmaggregatenode (continuação)		
propriedades rfmaggregatenode	**Tipo de dados**	**Descrição da propriedade**
transaction_scale	Days Weeks Months Years	Somente disponível se transaction_within_last for selecionado. Especifique o número e o tipo de períodos (dias, semanas, meses ou anos) desde Calcular Recência relativa até a data após a qual os registros serão incluídos em sua análise.
save_r2	*sinalização*	Exibe a data da segunda transação mais recente para cada cliente.
save_r3	*sinalização*	Somente disponível se save_r2 for selecionado. Exibe a data da terceira transação mais recente para cada cliente.

Propriedades de Rprocessnode

O nó Transformação R permite obter dados de um fluxo do IBM(r) SPSS(r) Modeler e modificá-los usando seu próprio script R customizado. Após os dados serem modificados, eles serão retornados para o fluxo.

Exemplo

```
node = stream.create("rprocess", "My node")
node.setPropertyValue("custom_name", "my_node")
node.setPropertyValue("syntax", """day<-as.Date(modelerData$dob, format="%Y-
%m-%d")
next_day<-day + 1
modelerData<-cbind(modelerData,next_day)
var1<-c(fieldName="Next
day",fieldLabel="",fieldStorage="date",fieldMeasure="",fieldFormat="",
fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
node.setPropertyValue("convert_datetime", "POSIXct")
```

Tabela 71. Propriedades de Rprocessnode		
propriedades Rprocessnode	**Tipo de dados**	**Descrição da propriedade**
syntax	*sequência*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*sinalização*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*sinalização*	

Tabela 71. Propriedades de Rprocessnode (continuação)		
propriedades Rprocessnode	**Tipo de dados**	**Descrição da propriedade**
use_batch_size	*sinalização*	Ativar uso do processamento em lote
batch_size	*Número inteiro*	Especifique o número de registros de dados a serem incluídos em cada lote

Propriedades de samplenode

O nó de Amostra seleciona um subconjunto de registros. É suportada uma variedade de tipos de amostra, incluindo amostras estratificadas, em cluster e não aleatórias (estruturadas). A amostragem pode ser útil para melhorar o desempenho e para selecionar grupos de registros ou transações relacionados para análise.

Exemplo

```
/* Create two Sample nodes to extract
   different samples from the same data */

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Simple")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("sample_type", "First")
node.setPropertyValue("first_n", 500)

node = stream.create("sample", "My node")
node.setPropertyValue("method", "Complex")
node.setPropertyValue("stratify_by", ["Sex", "Cholesterol"])
node.setPropertyValue("sample_units", "Proportions")
node.setPropertyValue("sample_size_proportions", "Custom")
node.setPropertyValue("sizes_proportions", [["M", "High", "Default"], ["M",
"Normal", "Default"],
 ["F", "High", 0.3], ["F", "Normal", 0.3]])
```

Tabela 72. Propriedades de samplenode		
propriedades samplenode	**Tipo de dados**	**Descrição da propriedade**
method	Simples Complexo	
mode	Include Discard	Inclui ou descarta registros que atendem a uma condição específica.
sample_type	First OneInN RandomPct	Especifica o método de amostragem.
first_n	*Número inteiro*	Registros até o ponto de corte especificado a serem incluídos ou descartados.
one_in_n	*number*	Inclui ou descarta a cada *n*º registro.

Tabela 72. Propriedades de samplenode (continuação)

propriedades samplenode	Tipo de dados	Descrição da propriedade
rand_pct	*number*	Especifique a porcentagem de registros a incluir ou descartar.
use_max_size	*sinalização*	Ative o uso da configuração maximum_size.
maximum_size	*Número inteiro*	Especifica a maior amostra a ser incluída ou descartada do fluxo de dados. Essa opção é redundante e, portanto, desativada quando First e Include são especificados.
set_random_seed	*sinalização*	Permite o uso da configuração de valor inicial aleatório.
random_seed	*Número inteiro*	Especifica o valor utilizado como um valor inicial aleatório.
complex_sample_type	Random Systematic	
sample_units	Proportions Counts	
sample_size_proportions	Fixed Custom Variable	
sample_size_counts	Fixed Custom Variable	
fixed_proportions	*number*	
fixed_counts	*Número inteiro*	
variable_proportions	*campo*	
variable_counts	*campo*	
use_min_stratum_size	*sinalização*	
minimum_stratum_size	*Número inteiro*	Essa opção só se aplica quando uma amostra Complexa é tomada com Sample units=Proportions.
use_max_stratum_size	*sinalização*	
maximum_stratum_size	*Número inteiro*	Essa opção só se aplica quando uma amostra Complexa é tomada com Sample units=Proportions.
clusters	*campo*	
stratify_by	*[field1 ... fieldN]*	

Tabela 72. Propriedades de samplenode (continuação)		
propriedades samplenode	**Tipo de dados**	**Descrição da propriedade**
specify_input_weight	*sinalização*	
input_weight	*campo*	
new_output_weight	*sequência*	
sizes_proportions	[[string *string value*] [string *string value*]...]	Se sample_units=proportions e sample_size_proportions=Custom, especifica um valor para cada combinação possível de valores de campos de estratificação.
default_proportion	*number*	
sizes_counts	[[string *string value*] [string *string value*]...]	Especifica um valor para cada combinação possível de valores de campo de estratificação. O uso é semelhante a sizes_proportions, mas especificando um número inteiro em vez de uma proporção.
default_count	*number*	

Propriedades de selectnode

O nó Selecionar seleciona ou descartará um subconjunto de registros do fluxo de dados com base em uma condição específica. Por exemplo, é possível selecionar os registros que pertencerem a uma região de vendas específica.

Exemplo

```
node = stream.create("select", "My node")
node.setPropertyValue("mode", "Include")
node.setPropertyValue("condition", "Age < 18")
```

Tabela 73. Propriedades de selectnode		
propriedades selectnode	**Tipo de dados**	**Descrição da propriedade**
mode	Include Discard	Especifica se os registros selecionados devem ser incluídos ou descartados.
condition	*sequência*	Condição para incluir ou descartar de registros.

Propriedades de sortnode

O nó Classificar classifica os registros em ordem crescente ou decrescente com base nos valores de um ou mais campos.

Exemplo

```
node = stream.create("sort", "My node")
node.setPropertyValue("keys", [["Age", "Ascending"], ["Sex", "Descending"]])
node.setPropertyValue("default_ascending", False)
node.setPropertyValue("use_existing_keys", True)
node.setPropertyValue("existing_keys", [["Age", "Ascending"]])
```

Tabela 74. Propriedades de sortnode

propriedades sortnode	**Tipo de dados**	**Descrição da propriedade**
keys	*lista*	Especifica os campos com relação aos quais você deseja classificar. Se nenhuma direção for especificada, o padrão será utilizado.
default_ascending	*sinalização*	Especifica a ordem de classificação padrão.
use_existing_keys	*sinalização*	Especifica se a classificação é otimizada utilizando a ordem de classificação anterior para campos que já estiverem classificados.
existing_keys		Especifica os campos que já estiverem classificados e a direção na qual eles são classificados. Usa o mesmo formato da propriedade keys.

propriedades de caixas de espaço

Space-Time-Boxes (STB) são uma extensão das localizações espaciais Geohashed. Mais especificamente, um STB é uma sequência alfanumérica que representa uma região de espaço e tempo de formato regular.

Tabela 75. propriedades de caixas de espaço

propriedades spacetimeboxes	**Tipo de dados**	**Descrição da propriedade**
mode	*IndividualRecords* *Hangouts*	
latitude_field	*campo*	
longitude_field	*campo*	
timestamp_field	*campo*	

Tabela 75. propriedades de caixas de espaço (continuação)		
propriedades spacetimeboxes	**Tipo de dados**	**Descrição da propriedade**

densities	*[densidade, densidade, densidade ...]*	Cada densidade é uma sequência. Por exemplo: STB_GH8_1DAY
		Observe que há limites para os quais densidades são válidas.
		Para o geohash, valores de GH1-GH15 podem ser usados.
		Para a parte temporal, os valores a seguir podem ser utilizados:
		`EVER`
		`1YEAR`
		`1MONTH`
		`1DAY`
		`12HOURS`
		`8HOURS`
		`6HOURS`
		`4HOURS`
		`3HOURS`
		`2HOURS`
		`1HOUR`
		`30MINS`
		`15MINS`
		`10MINS`
		`5MINS`
		`2 MINS`
		`1 MIN`
		`30SECS`
		`15SECS`
		`10SECS`
		`5 SECS`
		`2 SECS`
		`1SEC`

field_name_extension	*sequência*	
add_extension_as	*Prefixo* *Sufixo*	
hangout_density	*density*	Densidade única (ver acima)
id_field	*campo*	

qualifying_duration	`1DAY`
	`12HOURS`
	`8HOURS`
	`6HOURS`
	`4HOURS`
	`2HOURS`
	`1HOUR`
	`30MIN`
	`15MIN`
	`10MIN`
	`5MIN`
	`2MIN`
	`1MIN`
	`30SECS`
	`15SECS`
	`10SECS`
	`5SECS`
	`2SECS`
	`1SECS`
		Isso deve ser uma sequência.

min_events	*Número inteiro*	O valor mínimo é 2

Tabela 75. propriedades de caixas de espaço (continuação)		
propriedades spacetimeboxes	**Tipo de dados**	**Descrição da propriedade**
qualifying_pct	*Número inteiro*	Deve estar no intervalo de 1 a 100

propriedades streamingtimeseries

O nó Série temporal de fluxo constrói e pontua modelos de séries temporais em uma etapa.

Nota: Esse nó Série Temporal de Fluxo substitui o nó TS de Fluxo original que foi descontinuado na versão 18 do SPSS Modeler.

Tabela 76. propriedades streamingtimeseries		
Propriedades streamingtimeseries	**Valores**	**Descrição da propriedade**
targets	*campo*	O nó Séries Temporais de Fluxo prevê um ou mais destinos, opcionalmente usando um ou mais campos de entrada como preditores Os campos de frequência e de ponderação não são utilizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
candidate_inputs	[*field1 ... fieldN*]	Campos de entrada ou de preditores usados pelo modelo.
use_period	*sinalização*	
date_time_field	*campo*	

Tabela 76. propriedades streamingtimeseries (continuação)		
Propriedades `streamingtimeseries`	**Valores**	**Descrição da propriedade**

`input_interval`	`None`
	`Unknown`
	`Year`
	`Quarter`
	`Month`
	`Week`
	`Day`
	`Hour`
	`Hour_nonperiod`
	`Minute`
	`Minute_nonperiod`
	`Second`
	`Second_nonperiod`	

`period_field`	*campo*	
`period_start_value`	*Número inteiro*	
`num_days_per_week`	*Número inteiro*	

`start_day_of_week`	`Sunday`
	`Monday`
	`Tuesday`
	`Wednesday`
	`Thursday`
	`Friday`
	`Saturday`	

`num_hours_per_day`	*Número inteiro*	
`start_hour_of_day`	*Número inteiro*	

Tabela 76. propriedades streamingtimeseries (continuação)

Propriedades streamingtimeseries	**Valores**	**Descrição da propriedade**
timestamp_increments	*Número inteiro*	
cyclic_increments	*Número inteiro*	
cyclic_periods	*lista*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*sinalização*	
cross_hour	*sinalização*	
aggregate_and_distribute	*lista*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	

Tabela 76. propriedades streamingtimeseries (continuação)		
Propriedades `streamingtimeseries`	**Valores**	**Descrição da propriedade**

`group_default`	`Mean`
	`Sum`
	`Mode`
	`Min`
	`Max`	

`missing_imput`	`Linear_interp`
	`Series_mean`
	`K_mean`
	`K_median`
	`Linear_trend`	

`k_span_points`	*Número inteiro*	
`use_estimation_period`	*sinalização*	
`estimation_period`	`Observations` `Times`	
`date_estimation`	*lista*	Só disponível se você usar `date_time_field`
`period_estimation`	*lista*	Só disponível se você usar `use_period`
`observations_type`	`Latest` `Earliest`	
`observations_num`	*Número inteiro*	
`observations_exclude`	*Número inteiro*	

`method`	`ExpertModeler`
	`Exsmooth`
	`Arima`	

`expert_modeler_method`	`ExpertModeler`
	`Exsmooth`
	`Arima`	

Tabela 76. propriedades streamingtimeseries (continuação)

Propriedades streamingtimeseries	**Valores**	**Descrição da propriedade**
consider_seasonal	*sinalização*	
detect_outliers	*sinalização*	
expert_outlier_additive	*sinalização*	
expert_outlier_level_shift	*sinalização*	
expert_outlier_innovational	*sinalização*	
expert_outlier_level_shift	*sinalização*	
expert_outlier_transient	*sinalização*	
expert_outlier_seasonal_additive	*sinalização*	
expert_outlier_local_trend	*sinalização*	
expert_outlier_additive_patch	*sinalização*	
consider_newesmodels	*sinalização*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	

Tabela 76. propriedades streamingtimeseries (continuação)		
Propriedades `streamingtimeseries`	**Valores**	**Descrição da propriedade**

`futureValue_type_method`	`Compute`
	
	`specify`	

`exsmooth_transformation_type`	`None`
	
	`SquareRoot`
	
	`NaturalLog`	

`arima.p`	*Número inteiro*	
`arima.d`	*Número inteiro*	
`arima.q`	*Número inteiro*	
`arima.sp`	*Número inteiro*	
`arima.sd`	*Número inteiro*	
`arima.sq`	*Número inteiro*	

`arima_transformation_type`	`None`
	
	`SquareRoot`
	
	`NaturalLog`	

`arima_include_constant`	*sinalização*	
`tf_arima.p.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima.d.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima.q.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima.sp.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima.sd.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima.sq.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima.delay.` *fieldname*	*Número inteiro*	Para funções de transferência.

`tf_arima.transformation_type.` *fieldname*	`None`
	
	`SquareRoot`
	
	`NaturalLog`
		Para funções de transferência.

`arima_detect_outliers`	*sinalização*	

Tabela 76. propriedades streamingtimeseries (continuação)		
Propriedades streamingtimeseries	**Valores**	**Descrição da propriedade**
arima_outlier_additive	*sinalização*	
arima_outlier_level_shift	*sinalização*	
arima_outlier_innovational	*sinalização*	
arima_outlier_transient	*sinalização*	
arima_outlier_seasonal_additive	*sinalização*	
arima_outlier_local_trend	*sinalização*	
arima_outlier_additive_patch	*sinalização*	
conf_limit_pct	*real*	
events	*campos*	
forecastperiods	*Número inteiro*	
extend_records_into_future	*sinalização*	
conf_limits	*sinalização*	
noise_res	*sinalização*	

propriedades de streamingts (descontinuado)

Nota: Esse nó original do Streaming Time Series foi descontinuado na versão 18 do SPSS Modeler e substituído pelo novo nó do Streaming Time Series que foi projetado para aproveitar a energia do IBM SPSS Analytic Server e processar big data.

O nó TS de Fluxo constrói e escora os modelos de séries temporais em uma etapa, sem a necessidade de um nó Intervalos de Tempo.

Exemplo

```
node = stream.create("streamingts", "My node")
node.setPropertyValue("deployment_force_rebuild", True)
node.setPropertyValue("deployment_rebuild_mode", "Count")
node.setPropertyValue("deployment_rebuild_count", 3)
node.setPropertyValue("deployment_rebuild_pct", 11)
node.setPropertyValue("deployment_rebuild_field", "Year")
```

Tabela 77. Propriedades de streamingts		
propriedades streamingts	**Tipo de dados**	**Descrição da propriedade**
custom_fields	*sinalização*	Se custom_fields=false, as configurações de um nó Tipo de envio de dados são utilizadas. Se custom_fields=true, então targets e inputs deverão ser especificados.
targets	[*field1...fieldN*]	
inputs	[*field1...fieldN*]	

Tabela 77. Propriedades de streamingts (continuação)

propriedades streamingts	**Tipo de dados**	**Descrição da propriedade**

method	`ExpertModeler`
	`Exsmooth`
	`Arima`	

calculate_conf	*sinalização*	
conf_limit_pct	*real*	
use_time_intervals_node	*sinalização*	Se `use_time_intervals_node=true`, as configurações de um nó Intervalos de Tempo de envio de dados são utilizadas. Se `use_time_intervals_node=false`, `interval_offset_position`, então `interval_offset` e `interval_type` deverão ser especificados.
interval_offset_position	`LastObservation` `LastRecord`	`LastObservation` refere-se à **Última observação válida**. `LastRecord` refere-se a **Contagem de volta a partir do último registro**.
interval_offset	*number*	

interval_type	`Periods`
	`Years`
	`Quarters`
	`Months`
	`WeeksNonPeriodic`
	`DaysNonPeriodic`
	`HoursNonPeriodic`
	`MinutesNonPeriodic`
	`SecondsNonPeriodic`	

events	*campos*	

expert_modeler_method	`AllModels`
	`Exsmooth`
	`Arima`	

consider_seasonal	*sinalização*	
detect_outliers	*sinalização*	
expert_outlier_additive	*sinalização*	
expert_outlier_level_shift	*sinalização*	
expert_outlier_innovational	*sinalização*	
expert_outlier_transient	*sinalização*	
expert_outlier_seasonal_additive	*sinalização*	
expert_outlier_local_trend	*sinalização*	
expert_outlier_additive_patch	*sinalização*	

Tabela 77. Propriedades de streamingts (continuação)

propriedades streamingts	Tipo de dados	Descrição da propriedade

exsmooth_model_type	`Simple`
	`HoltsLinearTrend`
	`BrownsLinearTrend`
	`DampedTrend`
	`SimpleSeasonal`
	`WintersAdditive`
	`WintersMultiplicativ e`	

exsmooth_transformation_ type	`None`
	`SquareRoot`
	`NaturalLog`	

arima_p	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_d	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_q	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_sp	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_sd	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_sq	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais

arima_transformation_typ e	`None`
	`SquareRoot`
	`NaturalLog`
		Mesma propriedade do nó de modelagem de Séries Temporais

arima_include_constant	*sinalização*	Mesma propriedade do nó de modelagem de Séries Temporais
tf_arima_p.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_d.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_q.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_sp.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_sd.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.

Tabela 77. Propriedades de streamingts (continuação)		
propriedades streamingts	**Tipo de dados**	**Descrição da propriedade**
tf_arima_sq.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_delay.*fieldname*	*Número inteiro*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.

`tf_arima_transformation_type.`
fieldname
	`None`
	`SquareRoot`
	`NaturalLog`	

arima_detect_outlier_mode	`None` `Automatic`	
arima_outlier_additive	*sinalização*	
arima_outlier_level_shift	*sinalização*	
arima_outlier_innovational	*sinalização*	
arima_outlier_transient	*sinalização*	
arima_outlier_seasonal_additive	*sinalização*	
arima_outlier_local_trend	*sinalização*	
arima_outlier_additive_patch	*sinalização*	
deployment_force_rebuild	*sinalização*	
deployment_rebuild_mode	`Count` `Percent`	
deployment_rebuild_count	*number*	
deployment_rebuild_pct	*number*	
deployment_rebuild_field	*<field>*	

Capítulo 11. Propriedades do Nó de Operações de Campo

Propriedades anonymizenode

O nó Anonimizar transforma a maneira com que os nomes e valores de campo são representados no recebimento de dados, ocultando, assim, os dados originais. Isto poderá ser útil se desejar permitir que outros usuários construam modelos utilizando dados sensíveis, como nomes de cliente ou outros detalhes.

Exemplo

```
stream = modeler.script.stream()
varfilenode = stream.createAt("variablefile", "File", 96, 96)
varfilenode.setPropertyValue("full_filename", "$CLEO/DEMOS/DRUG1n")
node = stream.createAt("anonymize", "My node", 192, 96)
# Anonymize node requires the input fields while setting the values
stream.link(varfilenode, node)
node.setKeyedPropertyValue("enable_anonymize", "Age", True)
node.setKeyedPropertyValue("transformation", "Age", "Random")
node.setKeyedPropertyValue("set_random_seed", "Age", True)
node.setKeyedPropertyValue("random_seed", "Age", 123)
node.setKeyedPropertyValue("enable_anonymize", "Drug", True)
node.setKeyedPropertyValue("use_prefix", "Drug", True)
node.setKeyedPropertyValue("prefix", "Drug", "myprefix")
```

Tabela 78. Propriedades anonymizenode

propriedades anonymizenode	**Tipo de dados**	**Descrição da propriedade**
enable_anonymize	*sinalização*	Quando configurado como True, ativa a anonimização de valores de campo (equivalente a selecionar **Sim** para esse campo na coluna Anonimizar valores).
use_prefix	*sinalização*	Quando configurado como True, um prefixo customizado será usado se um tiver sido especificado. Aplica-se aos campos que serão anonimizados pelo método Hash e é equivalente a escolher o botão de opções **Customizado** na caixa de diálogo Substituir Valores para esse campo.
prefix	*sequência*	Equivalente a digitar um prefixo na caixa de texto na caixa de diálogo Substituir Valores. O prefixo padrão será o valor padrão se nada mais foi especificado.
transformation	Random Fixed	Determina se os parâmetros de transformação para um campo anonimizado pelo método Transformação serão aleatórios ou fixos.
set_random_seed	*sinalização*	Quando configurado como True, o valor inicial especificado será usado (se transformation também for configurado como Random).
random_seed	*Número inteiro*	Quando set_random_seed é configurado como True, esse é o valor inicial para o número aleatório.

Tabela 78. Propriedades anonymizenode (continuação)		
propriedades anonymizenode	**Tipo de dados**	**Descrição da propriedade**
`scale`	*number*	Quando `transformation` é configurado como `Fixed`, esse valor é usado para "escalar até." O valor máximo da escala normalmente é 10, mas poderá ser reduzido para evitar estouro.
`translate`	*number*	Quando `transformation` é configurado como `Fixed`, esse valor é usado para "converter". O valor máximo da conversão normalmente é 1000, mas poderá ser reduzido para evitar estouro.

propriedades autodataprepnode

O nó Automated Data Preparation (ADP) pode analisar seus dados e identificar correções, selecionar campos que são problemáticos ou que provavelmente não serão úteis, derivar novos atributos quando apropriado e aprimorar o desempenho por meio de técnicas de triagem e de amostragem inteligentes. É possível utilizar o nó de forma totalmente automatizada, permitindo que o nó escolha e aplique correções, ou é possível visualizar as mudanças antes que elas sejam feitas e aceitá-las, rejeitá-las ou modificá-las conforme desejado.

Exemplo

```
node = stream.create("autodataprep", "My node")
node.setPropertyValue("objective", "Balanced")
node.setPropertyValue("excluded_fields", "Filter")
node.setPropertyValue("prepare_dates_and_times", True)
node.setPropertyValue("compute_time_until_date", True)
node.setPropertyValue("reference_date", "Today")
node.setPropertyValue("units_for_date_durations", "Automatic")
```

Tabela 79. propriedades autodataprepnode		
propriedades autodataprepnode	**Tipo de dados**	**Descrição da propriedade**

`objective`	`Balanced`
	
	`Speed`
	
	`Accuracy`
	
	`Custom`	

`custom_fields`	*sinalização*	Se true, permite especificar campos de destino, de entrada e outros campos para o nó atual. Se false, as configurações atuais de um nó Tipo de envio de dados serão utilizadas.
`target`	*campo*	Especifica um campo de destino único.
`inputs`	[*field1 ... fieldN*]	Campos de entrada ou de preditores usados pelo modelo.

Tabela 79. propriedades autodataprepnode (continuação)		
propriedades autodataprepnode	**Tipo de dados**	**Descrição da propriedade**
use_frequency	*sinalização*	
frequency_field	*campo*	
use_weight	*sinalização*	
weight_field	*campo*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*sinalização*	Controla o acesso a todos os campos de data e hora
compute_time_until_date	*sinalização*	
reference_date	Today Fixed	
fixed_date	*Data*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*sinalização*	
reference_time	CurrentTime Fixed	
fixed_time	*time*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	*sinalização*	

Tabela 79. propriedades autodataprepnode (continuação)

propriedades autodataprepnode	**Tipo de dados**	**Descrição da propriedade**
extract_month_from_date	*sinalização*	
extract_day_from_date	*sinalização*	
extract_hour_from_time	*sinalização*	
extract_minute_from_time	*sinalização*	
extract_second_from_time	*sinalização*	
exclude_low_quality_inputs	*sinalização*	
exclude_too_many_missing	*sinalização*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*sinalização*	
maximum_number_categories	*number*	
exclude_if_large_category	*sinalização*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*sinalização*	
adjust_type_inputs	*sinalização*	
adjust_type_target	*sinalização*	
reorder_nominal_inputs	*sinalização*	
reorder_nominal_target	*sinalização*	
replace_outliers_inputs	*sinalização*	
replace_outliers_target	*sinalização*	
replace_missing_continuous_inputs	*sinalização*	
replace_missing_continuous_target	*sinalização*	
replace_missing_nominal_inputs	*sinalização*	
replace_missing_nominal_target	*sinalização*	
replace_missing_ordinal_inputs	*sinalização*	
replace_missing_ordinal_target	*sinalização*	
maximum_values_for_ordinal	*number*	

Tabela 79. propriedades autodataprepnode (continuação)

propriedades autodataprepnode	**Tipo de dados**	**Descrição da propriedade**
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*sinalização*	
rescaling_method	MinMax ZScore	
min_max_minimum	*number*	
min_max_maximum	*number*	
z_score_final_mean	*number*	
z_score_final_sd	*number*	
rescale_continuous_target	*sinalização*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*sinalização*	
maximize_association_with_target	*sinalização*	
p_value_for_merging	*number*	
merge_ordinal_features	*sinalização*	
merge_nominal_features	*sinalização*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*sinalização*	
p_value_for_binning	*number*	
perform_feature_selection	*sinalização*	
p_value_for_selection	*number*	
perform_feature_construction	*sinalização*	
transformed_target_name_extension	*sequência*	
transformed_inputs_name_extension	*sequência*	
constructed_features_root_name	*sequência*	

Tabela 79. propriedades autodataprepnode (continuação)		
propriedades autodataprepnode	**Tipo de dados**	**Descrição da propriedade**
years_duration_ name_extension	*sequência*	
months_duration_ name_extension	*sequência*	
days_duration_ name_extension	*sequência*	
hours_duration_ name_extension	*sequência*	
minutes_duration_ name_extension	*sequência*	
seconds_duration_ name_extension	*sequência*	
year_cyclical_name_extens ion	*sequência*	
month_cyclical_name_exten sion	*sequência*	
day_cyclical_name_extensi on	*sequência*	
hour_cyclical_name_extens ion	*sequência*	
minute_cyclical_name_exte nsion	*sequência*	
second_cyclical_name_exte nsion	*sequência*	

Propriedades de astimeintervalsnode

Use o nó Intervalos de tempo para especificar intervalos e derivar um novo campo de tempo para estimar ou prever. Uma variedade completa de intervalos de tempo é suportada, desde segundos até anos.

Tabela 80. Propriedades de astimeintervalsnode		
propriedades astimeintervalsnode	**Tipo de dados**	**Descrição da propriedade**
time_field	*campo*	Pode aceitar apenas um único campo contínuo. Esse campo é utilizado pelo nó como a chave de agregação para converter o intervalo. Se um campo de número inteiro for utilizado aqui, ele será considerado como um índice de tempo.
dimensions	*[field1 field2 ... fieldn]*	Estes campos são usados para criar séries temporais individuais com base nos valores do campo.

Tabela 80. Propriedades de astimeintervalsnode (continuação)		
propriedades astimeintervalsnode	**Tipo de dados**	**Descrição da propriedade**
fields_to_aggregate	*[field1 field2 ... fieldn]*	Esses campos são agregados como parte da mudança do período do campo de tempo. Todos os campos não incluídos nesse selecionador são filtrados dos dados que saem do nó.

Propriedades de binningnode

O nó Categorização cria automaticamente novos campos nominais (conjunto) com base nos valores de um ou mais campos existente contínuos (intervalo numérico). Por exemplo, é possível transformar um campo de receita contínuo em um novo campo categórico contendo grupos de receitas como desvios do menu. Depois de criar categorias para o novo campo, é possível gerar um nó Derivar com base nos pontos de corte.

Exemplo

```
node = stream.create("binning", "My node")
node.setPropertyValue("fields", ["Na", "K"])
node.setPropertyValue("method", "Rank")
node.setPropertyValue("fixed_width_name_extension", "_binned")
node.setPropertyValue("fixed_width_add_as", "Suffix")
node.setPropertyValue("fixed_bin_method", "Count")
node.setPropertyValue("fixed_bin_count", 10)
node.setPropertyValue("fixed_bin_width", 3.5)
node.setPropertyValue("tile10", True)
```

Tabela 81. Propriedades de binningnode		
propriedades binningnode	**Tipo de dados**	**Descrição da propriedade**
fields	*[field1 field2 ... fieldn]*	Transformação pendente de campos contínuos (intervalo numérico). É possível categorizar diversos campos simultaneamente.
method	FixedWidth EqualCount Rank SDev Optimal	Método utilizado para determinar os pontos de corte para novas categorias de campo (categorias).
rcalculate_bins	Always IfNecessary	Especifica se as categorias são recalculadas e se os dados são colocados na categoria relevante toda vez que o nó for executado ou se os dados são incluídos apenas nas categorias existentes e em quaisquer novas categorias que foram incluídas.

Tabela 81. Propriedades de binningnode (continuação)		
propriedades binningnode	**Tipo de dados**	**Descrição da propriedade**
fixed_width_name_extension	*sequência*	A extensão padrão é *_BIN*.
fixed_width_add_as	Suffix Prefix	Especifica se a extensão é incluída no término (sufixo) do nome do campo ou no início (prefixo). A extensão padrão é *income_BIN*.
fixed_bin_method	Width Count	
fixed_bin_count	*Número inteiro*	Especifica um número inteiro utilizado para determinar o número de bins (categorias) de largura fixa para o(s) novo(s) campo(s).
fixed_bin_width	*real*	Valor (número inteiro ou real) para calcular a largura da categoria.
equal_count_name_ extension	*sequência*	A extensão padrão é *_TILE*.
equal_count_add_as	Suffix Prefix	Especifica uma extensão, seja um prefixo ou sufixo, utilizada para o nome do campo gerado usando p-tiles padrão. A extensão padrão é *_TILE* e *N*, em que *N* é o número do ladrilho.
tile4	*sinalização*	Gera quatro categorias de quantil, cada uma contendo 25% dos casos.
tile5	*sinalização*	Gera cinco categorias de quintil.
tile10	*sinalização*	Gera 10 categorias de decil.
tile20	*sinalização*	Gera 20 categorias de vingtile.
tile100	*sinalização*	Gera 100 categorias de percentil.
use_custom_tile	*sinalização*	
custom_tile_name_extension	*sequência*	A extensão padrão é *_TILEN*.
custom_tile_add_as	Suffix Prefix	
custom_tile	*Número inteiro*	
equal_count_method	RecordCount ValueSum	O método RecordCount procura atribuir um número igual de registros a cada categoria, enquanto ValueSum atribui registros para que a soma dos valores em cada categoria seja igual.

Tabela 81. Propriedades de binningnode (continuação)

propriedades binningnode	Tipo de dados	Descrição da propriedade
tied_values_method	Next Current Random	Especifica em qual categoria os dados de valor empatado devem ser colocados.
rank_order	Ascending Descending	Esta propriedade inclui Ascending (o valor mais baixo é marcado 1) ou Descending (o valor mais alto é marcado 1).
rank_add_as	Suffix Prefix	Essa opção se aplica à classificação, à classificação fracional e à classificação de porcentagem.
rank	*sinalização*	
rank_name_extension	*sequência*	A extensão padrão é *_RANK*.
rank_fractional	*sinalização*	Classifica casos em que o valor do novo campo é igual à classificação dividido pela soma dos pesos dos casos não desconhecidos. Os postos fracionários entram no intervalo de 0 a 1.
rank_fractional_name_ extension	*sequência*	A extensão padrão é *_F_RANK*.
rank_pct	*sinalização*	Cada classificação é dividida pelo número de registros com valores válidos e multiplicada por 100. A porcentagem dos postos fracionários entra no intervalo de 1 a 100.
rank_pct_name_extension	*sequência*	A extensão padrão é *_P_RANK*.
sdev_name_extension	*sequência*	
sdev_add_as	Suffix Prefix	
sdev_count	One Two Three	
optimal_name_extension	*sequência*	A extensão padrão é *_OPTIMAL*.
optimal_add_as	Suffix Prefix	

Tabela 81. Propriedades de binningnode (continuação)		
propriedades binningnode	**Tipo de dados**	**Descrição da propriedade**
optimal_supervisor_field	*campo*	O campo escolhido como o campo de supervisão para o qual os campos selecionados para a categorização estão relacionados.
optimal_merge_bins	*sinalização*	Especifica que quaisquer categorias com contagens de caso pequenas serão incluídas em uma categoria vizinha maior.
optimal_small_bin_threshold	*Número inteiro*	
optimal_pre_bin	*sinalização*	Indica que a pré-categorização do conjunto de dados deve ocorrer.
optimal_max_bins	*Número inteiro*	Especifica um limite superior para evitar a criação de um grande número de categorias excessivamente.
optimal_lower_end_point	Inclusive Exclusive	
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

Propriedades de derivenode

O nó de Derivação modifica valores de dados ou cria novos campos a partir de um ou mais campos existentes. Ele cria campos de fórmula tipo, bandeira, nominal, estado, contagem e condicional.

Exemplo 1

```
# Create and configure a Flag Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("new_name", "DrugX_Flag")
node.setPropertyValue("result_type", "Flag")
node.setPropertyValue("flag_true", "1")
node.setPropertyValue("flag_false", "0")
node.setPropertyValue("flag_expr", "'Drug' == \"drugX\"")

# Create and configure a Conditional Derive field node
node = stream.create("derive", "My node")
node.setPropertyValue("result_type", "Conditional")
node.setPropertyValue("cond_if_cond", "@OFFSET(\"Age\", 1) = \"Age\"")
node.setPropertyValue("cond_then_expr", "(@OFFSET(\"Age\", 1) = \"Age\" ><
@INDEX")
node.setPropertyValue("cond_else_expr", "\"Age\"")
```

Exemplo 2

Este script presume que há duas colunas numéricas chamadas XPos e YPos que representam as coordenadas X e Y de um ponto (por exemplo, onde ocorreu um evento). O script cria um nó Derivar que calcula uma coluna geoespacial das coordenadas X e Y que representam esse ponto em um sistema de coordenadas específicas:

```
stream = modeler.script.stream()
# Other stream configuration code
node = stream.createAt("derive", "Location", 192, 96)
node.setPropertyValue("new_name", "Location")
node.setPropertyValue("formula_expr", "['XPos', 'YPos']")
node.setPropertyValue("formula_type", "Geospatial")
# Now we have set the general measurement type, define the
# specifics of the geospatial object
node.setPropertyValue("geo_type", "Point")
node.setPropertyValue("has_coordinate_system", True)
node.setPropertyValue("coordinate_system", "ETRS_1989_EPSG_Arctic_zone_5-47")
```

Tabela 82. Propriedades de derivenode

propriedades derivenode	**Tipo de dados**	**Descrição da propriedade**
new_name	*sequência*	Nome do novo campo.
mode	Single Multiple	Especifica campos únicos ou múltiplos.
fields	*lista*	Usado no modo Multiple apenas para selecionar diversos campos.
name_extension	*sequência*	Especifica a extensão para um ou mais novos nomes do campo.
add_as	Suffix Prefix	Inclui a extensão como um prefixo (no início) ou como um sufixo (no término) do nome do campo.
result_type	Formula Flag Set State Count Conditional	Os seis tipos de novos campos que podem ser criados.
formula_expr	*sequência*	A expressão para calcular um novo valor de campo em um nó Derivar.
flag_expr	*sequência*	
flag_true	*sequência*	
flag_false	*sequência*	

Tabela 82. Propriedades de derivenode (continuação)

propriedades derivenode	**Tipo de dados**	**Descrição da propriedade**
set_default	*sequência*	
set_value_cond	*sequência*	Estruturada para fornecer a condição associada a um determinado valor.
state_on_val	*sequência*	Especifica o valor para o novo campo quando a condição On for atendida.
state_off_val	*sequência*	Especifica o valor para o novo campo quando a condição Off for atendida.
state_on_expression	*sequência*	
state_off_expression	*sequência*	
state_initial	On Off	Atribui a cada registro do novo campo um valor inicial de On ou Off. Este valor pode alterar conforme cada condição for atendida.
count_initial_val	*sequência*	
count_inc_condition	*sequência*	
count_inc_expression	*sequência*	
count_reset_condition	*sequência*	
cond_if_cond	*sequência*	
cond_then_expr	*sequência*	
cond_else_expr	*sequência*	

Tabela 82. Propriedades de derivenode (continuação)

propriedades derivenode	**Tipo de dados**	**Descrição da propriedade**
formula_measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Essa propriedade pode ser utilizada para definir a medida associada ao campo derivado. À função setter pode ser passada uma sequência ou um dos valores MeasureType. O getter sempre retornará nos valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de coleta (listas com uma profundidade 0), essa propriedade define o tipo de medição associado aos valores subjacentes.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Para campos geoespaciais, essa propriedade define o tipo de objeto geoespacial representado por este campo. Isso deve ser consistente com a profundidade da lista dos valores
has_coordinate_system	*Booleano*	Para campos geoespaciais, essa propriedade define se esse campo tem um sistema de coordenadas

Tabela 82. Propriedades de derivenode (continuação)		
propriedades derivenode	**Tipo de dados**	**Descrição da propriedade**
coordinate_system	*sequência*	Para campos geoespaciais, essa propriedade define o sistema de coordenadas para este campo

Propriedades de ensemblenode

O Conjunto Ensemble combina dois ou mais nuggets de modelo para obter previsões mais precisas do que pode ser obtido a partir de qualquer modelo.

Exemplo

```
# Create and configure an Ensemble node
# Use this node with the models in demos\streams\pm_binaryclassifier.str
node = stream.create("ensemble", "My node")
node.setPropertyValue("ensemble_target_field", "response")
node.setPropertyValue("filter_individual_model_output", False)
node.setPropertyValue("flag_ensemble_method", "ConfidenceWeightedVoting")
node.setPropertyValue("flag_voting_tie_selection", "HighestConfidence")
```

Tabela 83. Propriedades de ensemblenode		
propriedades ensemblenode	**Tipo de dados**	**Descrição da propriedade**
ensemble_target_field	*campo*	Especifica o campo de destino para todos os modelos usados na combinação.
filter_individual_model_output	*sinalização*	Especifica se os resultados da escoragem de modelos individuais devem ser suprimidos.
flag_ensemble_method	Voting ConfidenceWeightedVoting RawPropensityWeightedVoting AdjustedPropensityWeightedVoting HighestConfidence AverageRawPropensity AverageAdjustedPropensity	Especifica o método utilizado para determinar o escore de combinação. Essa configuração se aplicará apenas se o destino selecionado for um campo de sinalização.

Tabela 83. Propriedades de ensemblenode (continuação)		
propriedades ensemblenode	**Tipo de dados**	**Descrição da propriedade**
set_ensemble_method	Voting ConfidenceWeightedVoting HighestConfidence	Especifica o método utilizado para determinar o escore de combinação. Essa configuração se aplicará apenas se o destino selecionado for um campo nominal.
flag_voting_tie_selection	Random HighestConfidence RawPropensity AdjustedPropensity	Se um método de votação for selecionado, especifica como os empates serão resolvidos. Essa configuração se aplicará apenas se o destino selecionado for um campo de sinalização.
set_voting_tie_selection	Random HighestConfidence	Se um método de votação for selecionado, especifica como os empates serão resolvidos. Essa configuração se aplicará apenas se o destino selecionado for um campo nominal.
calculate_standard_error	*sinalização*	Se o campo de destino for contínuo, um cálculo de erro padrão é executado por padrão para calcular a diferença entre os valores medidos ou estimados e os valores reais e para mostrar a proximidade com que essas estimativas corresponderam.

Propriedadades de fillernode

O nó de Preenchimento substitui valores do campo altera o armazenamento. É possível optar por substituir valores baseados em uma condição de CLEM, como `@BLANK(@FIELD)`. Como alternativa, é possível optar por substituir todos os valores em branco ou nulos por um valor específico. Um nó Filler é frequentemente usado juntamente com um nó Type para substituir valores ausentes.

Exemplo

```
node = stream.create("filler", "My node")
node.setPropertyValue("fields", ["Age"])
node.setPropertyValue("replace_mode", "Always")
node.setPropertyValue("condition", "(\"Age\" > 60) and (\"Sex\" = \"M\"")
node.setPropertyValue("replace_with", "\"old man\"")
```

Tabela 84. Propriedadades de fillernode		
propriedades fillernode	**Tipo de dados**	**Descrição da propriedade**
fields	*lista*	Campos do conjunto de dados cujos valores serão examinados e substituídos.

Tabela 84. Propriedadades de fillernode (continuação)		
propriedades `fillernode`	**Tipo de dados**	**Descrição da propriedade**

`replace_mode`	`Always`
	
	`Conditional`
	
	`Blank`
	
	`Null`
	
	`BlankAndNull`
		É possível substituir todos os valores, valores em branco ou valores nulos ou substituir com base em uma condição especificada.

`condition`	*sequência*	
`replace_with`	*sequência*	

Propriedades de filternode

O nó Filtro filtra (descarta) os campos, renomeia-os e mapeia-os de um nó de origem para outro.

Exemplo:

```
node = stream.create("filter", "My node")
node.setPropertyValue("default_include", True)
node.setKeyedPropertyValue("new_name", "Drug", "Chemical")
node.setKeyedPropertyValue("include", "Drug", False)
```

Utilizando a propriedade default_include. Observe que configurar o valor da propriedade `default_include` não inclui ou exclui automaticamente todos os campos, apenas determina o padrão para a seleção atual. Isso é funcionalmente equivalente a clicar no botão **Incluir campos por padrão** na caixa de diálogo do nó Filtro. Por exemplo, suponha que você execute o script a seguir:

```
node = modeler.script.stream().create("filter", "Filter")
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["Age", "Sex"]:
    node.setKeyedPropertyValue("include", f, True)
```

Isso fará com que o nó passe os campos Age e Sex e descarte todos os outros. Depois de executar o script anterior, agora suponha que você inclua as seguintes linhas no script para nomear mais dois campos:

```
node.setPropertyValue("default_include", False)
# Include these two fields in the list
for f in ["BP", "Na"]:
    node.setKeyedPropertyValue("include", f, True)
```

Isso incluirá mais dois campos no filtro para que um total de quatro campos sejam passados (Age, Sex, BP, Na). Em outras palavras, reconfigurar o valor de `default_include` para `False` não reconfigura automaticamente todos os campos.

Como alternativa, se agora você mudasse `default_include` para `True`, usando um script ou na caixa de diálogo do nó de Filtro, isso inverteria o comportamento de modo que os quatro campos listados acima

seriam descartados em vez de incluídos. Quando estiver em dúvida, experimentar os controles na caixa de diálogo do nó Filtro poderá ser útil para entender essa interação.

Tabela 85. Propriedades de filternode

propriedades filternode	Tipo de dados	Descrição da propriedade
default_include	*sinalização*	Propriedade definida como chave para especificar se o comportamento padrão é transmitir ou filtrar os campos: Observe que configurar essa propriedade não inclui ou exclui automaticamente todos os campos, apenas determina se os campos selecionados são incluídos ou excluídos por padrão. Consulte o exemplo abaixo para obter comentários adicionais.
include	*sinalização*	Propriedade definida como chave para inclusão e remoção de campo.
new_name	*sequência*	

Propriedades de historynode

O nó Histórico cria novos campos contendo dados de campos em registros anteriores. Os nós Históricos são mais frequentemente utilizados para dados sequenciais, como dados de séries temporais. Antes de usar um nó Histórico, você pode querer classificar os dados usando um nó Sort.

Exemplo

```
node = stream.create("history", "My node")
node.setPropertyValue("fields", ["Drug"])
node.setPropertyValue("offset", 1)
node.setPropertyValue("span", 3)
node.setPropertyValue("unavailable", "Discard")
node.setPropertyValue("fill_with", "undef")
```

Tabela 86. Propriedades de historynode

propriedades historynode	Tipo de dados	Descrição da propriedade
fields	*lista*	Campos para os quais você deseja um histórico.
offset	*number*	Especifica o registro mais recente (anterior ao registro atual) a partir do qual você deseja extrair os valores de campo de histórico.
span	*number*	Especifica o número de registros anteriores a partir do qual você deseja extrair os valores.

Tabela 86. Propriedades de historynode (continuação)		
propriedades historynode	**Tipo de dados**	**Descrição da propriedade**
unavailable	Discard Leave Fill	Para manipular registros que não possuem valores históricos, geralmente referenciando os vários primeiros registros (na parte superior do conjunto de dados) para os quais não há registros anteriores para uso como um histórico.
fill_with	String Number	Especifica um valor ou uma sequência a ser utilizada para registros nos quais nenhum valor histórico está disponível.

Propriedades de partitionnode

O nó Partição gera um campo de partição que divide os dados em subconjuntos separados para o treinamento, teste e estágios de validação de construção de modelo.

Exemplo

```
node = stream.create("partition", "My node")
node.setPropertyValue("create_validation", True)
node.setPropertyValue("training_size", 33)
node.setPropertyValue("testing_size", 33)
node.setPropertyValue("validation_size", 33)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 123)
node.setPropertyValue("value_mode", "System")
```

Tabela 87. Propriedades de partitionnode		
propriedades partitionnode	**Tipo de dados**	**Descrição da propriedade**
new_name	*sequência*	Nome do campo de partição gerado pelo nó.
create_validation	*sinalização*	Especifica se uma partição de validação deve ser criada.
training_size	*Número inteiro*	Porcentagem de registros (0 a 100) a ser alocada para a partição de treinamento.
testing_size	*Número inteiro*	Porcentagem de registros (0 a 100) a ser alocada para a partição de teste.
validation_size	*Número inteiro*	Porcentagem de registros (0 a 100) a ser alocada para a partição de validação. Ignorado se uma partição de validação não for criada.
training_label	*sequência*	Rótulo para a partição de treinamento.
testing_label	*sequência*	Rótulo para a partição de teste.
validation_label	*sequência*	Rótulo para a partição de validação. Ignorado se uma partição de validação não for criada.

Tabela 87. Propriedades de partitionnode (continuação)		
propriedades partitionnode	**Tipo de dados**	**Descrição da propriedade**
value_mode	System SystemAndLabel Label	Especifica os valores utilizados para representar cada partição nos dados. Por exemplo, a amostra de treinamento pode ser representada pelo número inteiro do sistema 1, o rótulo Training ou uma combinação dos dois, 1_Training.
set_random_seed	*Booleano*	Especifica se um valor inicial aleatório especificado pelo usuário deve ser utilizado.
random_seed	*Número inteiro*	Um valor inicial aleatório especificados pelo usuário. Para que esse valor seja usado, set_random_seed deve ser configurado como True.
enable_sql_generation	*Booleano*	Especifica se o SQL pushback deve ser usado para designar registros para partições.
unique_field		Especifica o campo de entrada utilizado para assegurar que os registros sejam designados a partições em um modo aleatório, porém repetitivo. Para que esse valor seja usado, enable_sql_generation deve ser configurado como True.

propriedades reclassifynode

O nó Reclassificar transforma um conjunto de valores categóricos em outro. A reclassificação é útil para reduzir as categorias ou para reagrupar dados para análise.

Exemplo

```
node = stream.create("reclassify", "My node")
node.setPropertyValue("mode", "Multiple")
node.setPropertyValue("replace_field", True)
node.setPropertyValue("field", "Drug")
node.setPropertyValue("new_name", "Chemical")
node.setPropertyValue("fields", ["Drug", "BP"])
node.setPropertyValue("name_extension", "reclassified")
node.setPropertyValue("add_as", "Prefix")
node.setKeyedPropertyValue("reclassify", "drugA", True)
node.setPropertyValue("use_default", True)
node.setPropertyValue("default", "BrandX")
node.setPropertyValue("pick_list", ["BrandX", "Placebo", "Generic"])
```

Tabela 88. propriedades reclassifynode		
propriedades reclassifynode	**Tipo de dados**	**Descrição da propriedade**
mode	Single Multiple	Single reclassifica as categorias para um campo. Multiple ativa opções que possibilitam a transformação de mais de um campo de cada vez.
replace_field	*sinalização*	
field	*sequência*	Usado apenas no modo Single.
new_name	*sequência*	Usado apenas no modo Single.
fields	*[field1 field2 ... fieldn]*	Usado apenas no modo Multiple.
name_extension	*sequência*	Usado apenas no modo Multiple.
add_as	Suffix Prefix	Usado apenas no modo Multiple.
reclassify	*sequência*	Propriedade estruturada para valores de campo.
use_default	*sinalização*	Utilize o valor padrão.
default	*sequência*	Especifica um valor padrão.
pick_list	*[string string ... string]*	Permite que um usuário importe uma lista de novos valores conhecidos para preencher a lista suspensa na tabela.

Propriedades de reordernode

O nó Reordenar Campo define a ordem natural utilizada para exibir campos de recebimento de dados. Esta ordem afeta a exibição de campos em uma variedade de locais, como tabelas, listas e o Seletor de campo. Esta operação é útil ao trabalhar com conjuntos de dados amplos para tornar os campos de interesse mais visíveis.

Exemplo

```
node = stream.create("reorder", "My node")
node.setPropertyValue("mode", "Custom")
node.setPropertyValue("sort_by", "Storage")
node.setPropertyValue("ascending", False)
node.setPropertyValue("start_fields", ["Age", "Cholesterol"])
node.setPropertyValue("end_fields", ["Drug"])
```

Tabela 89. Propriedades de reordernode		
propriedades reordernode	**Tipo de dados**	**Descrição da propriedade**
mode	Custom Auto	É possível classificar valores automaticamente ou especificar uma ordem customizada.

Tabela 89. Propriedades de reordernode (continuação)		
propriedades reordernode	**Tipo de dados**	**Descrição da propriedade**
sort_by	Name Type Storage	
ascending	*sinalização*	
start_fields	*[field1 field2 ... fieldn]*	Novos campos são inseridos após esses campos.
end_fields	*[field1 field2 ... fieldn]*	Novos campos são inseridos antes desses campos.

Propriedades de reprojectnode

No SPSS Modeler, itens como as funções espaciais do Construtor de Expressões, o nó Spatio-Temporal Prediction (STP), e o nó Visualização de Mapa usam o sistema de coordenadas projetado. Utilize o nó Reprojetar para alterar o sistema de coordenadas de quaisquer dados que importar que utilizam um sistema de coordenadas geográficas.

Tabela 90. Propriedades de reprojectnode		
propriedades reprojectnode	**Tipo de dados**	**Descrição da propriedade**
reproject_fields	*[field1 field2 ... fieldn]*	Lista todos os campos que devem ser reprojetados.
reproject_type	Streamdefault Specify	Escolha como deseja reprojetar os campos.
coordinate_system	*sequência*	O nome do sistema de coordenadas a ser aplicado aos campos. Exemplo: set reprojectnode.coordinate_system = “WGS_1984_World_Mercator”

Propriedades de restructurenode

O nó Reestruturar converte um campo nominal ou sinalizador em um grupo de campos que podem ser preenchidos com os valores do outro campo. Por exemplo, dado um campo denominado *tipo de pagamento*, com valores de *credit*, *cashe débit*, três novos campos seriam criados (*crédito*, *caixa*, *débito*), cada um deles poderá conter o valor do pagamento real feito.

Exemplo

```
node = stream.create("restructure", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("include_field_name", True)
```

```
node.setPropertyValue("value_mode", "OtherFields")
node.setPropertyValue("value_fields", ["Age", "BP"])
```

Tabela 91. Propriedades de restructurenode

propriedades `restructurenode`	**Tipo de dados**	**Descrição da propriedade**
`fields_from`	[*category category category*] `all`	
`include_field_name`	*sinalização*	Indica se o nome do campo deve ser usado no nome do campo reestruturado.

`value_mode`	`OtherFields`
	
	`Flags`
		Indica o modo para especificar os valores para os campos reestruturados. Com `OtherFields`, deve-se especificar quais campos usar (veja abaixo). Com `Flags`, os valores são sinalizações numéricas.

`value_fields`	*lista*	Necessário se `value_mode` for `OtherFields`. Especifica quais campos deseja utilizar como campos de valor.

Propriedades de rfmanalysisnode

O nó de Análise de Recência, Frequência, Monetária (RFM) possibilita determinar quantitativamente quais os clientes provavelmente serão os melhores examinando como recentemente eles compraram duramente de você (recência), com que frequência compraram (frequência), e o quanto gastaram sobre todas as transações (monetária).

Exemplo

```
node = stream.create("rfmanalysis", "My node")
node.setPropertyValue("recency", "Recency")
node.setPropertyValue("frequency", "Frequency")
node.setPropertyValue("monetary", "Monetary")
node.setPropertyValue("tied_values_method", "Next")
node.setPropertyValue("recalculate_bins", "IfNecessary")
node.setPropertyValue("recency_thresholds", [1, 500, 800, 1500, 2000, 2500])
```

Tabela 92. Propriedades de rfmanalysisnode

propriedades `rfmanalysisnode`	**Tipo de dados**	**Descrição da propriedade**
`recency`	*campo*	Especifica o campo de recência. Isso pode ser uma data, um registro de data e hora ou número simples.
`frequency`	*campo*	Especifica o campo de frequência.
`monetary`	*campo*	Especifica o campo monetário.
`recency_bins`	*Número inteiro*	Especifica o número de categorias de recência a serem geradas.
`recency_weight`	*number*	Especifica o peso a ser aplicado aos dados de recência. O padrão é 100.

Tabela 92. Propriedades de rfmanalysisnode (continuação)		
propriedades rfmanalysisnode	**Tipo de dados**	**Descrição da propriedade**
frequency_bins	*Número inteiro*	Especifica o número de categorias de frequência a serem geradas.
frequency_weight	*number*	Especifica o peso a ser aplicado aos dados de frequência. O padrão é 10.
monetary_bins	*Número inteiro*	Especifica o número de categorias monetárias a serem geradas.
monetary_weight	*number*	Especifica o peso a ser aplicado aos dados monetários. O padrão é 1.
tied_values_method	Next Current	Especifica em qual categoria os dados de valor vinculados devem ser colocados.
recalculate_bins	Always IfNecessary	
add_outliers	*sinalização*	Disponível apenas se recalculate_bins for configurado como IfNecessary. Se configurado, os registros que estiverem em uma categoria inferior serão incluídos na categoria inferior e os registros acima da categoria mais alta serão incluídos na categoria mais alta.
binned_field	Recency Frequency Monetary	
recency_thresholds	*valor valor*	Disponível apenas se recalculate_bins for configurado como Always. Especifique os limites superior e inferior para as categorias de recência. O limite superior de uma categoria é utilizado como o limite inferior da próxima categoria, por exemplo, [10 30 60] define duas categorias, a primeira categoria com limites superior e inferior de 10 e 30 e a segunda categoria com limites de 30 e 60.
frequency_thresholds	*valor valor*	Disponível apenas se recalculate_bins for configurado como Always.
monetary_thresholds	*valor valor*	Disponível apenas se recalculate_bins for configurado como Always.

Propriedades de settoflagnode

O nó Configurar para Sinalizador deriva diversos campos de sinalização com base nos valores categóricos definidos para um ou mais campos nominais.

Exemplo

```
node = stream.create("settoflag", "My node")
node.setKeyedPropertyValue("fields_from", "Drug", ["drugA", "drugX"])
node.setPropertyValue("true_value", "1")
node.setPropertyValue("false_value", "0")
node.setPropertyValue("use_extension", True)
node.setPropertyValue("extension", "Drug_Flag")
node.setPropertyValue("add_as", "Suffix")
node.setPropertyValue("aggregate", True)
node.setPropertyValue("keys", ["Cholesterol"])
```

Tabela 93. Propriedades de settoflagnode

propriedades settoflagnode	**Tipo de dados**	**Descrição da propriedade**
fields_from	[*category category category*] all	
true_value	*sequência*	Especifica o valor true utilizado pelo nó ao configurar um sinalizador. O padrão é T.
false_value	*sequência*	Especifica o valor false utilizado pelo nó ao configurar um sinalizador. O padrão é F.
use_extension	*sinalização*	Utiliza uma extensão como um sufixo ou prefixo para o novo campo de sinalização.
extension	*sequência*	
add_as	Suffix Prefix	Especifica se a extensão é incluída como um prefixo ou sufixo.
aggregate	*sinalização*	Agrupa registros com base em campos-chave. Todos os campos de sinalização em um grupo serão ativados se algum registro for configurado como true.
keys	*lista*	Campos-chave.

Propriedades de statisticstransformnode

O nó Transformação de Estatísticas executa uma seleção de comandos de sintaxe do IBM SPSS Statistics com relação às origens de dados no IBM SPSS Modeler. Este nó requer uma cópia licenciada de IBM SPSS Statistics.

As propriedades desse nó são descritas em “Propriedades de statisticstransformnode” na página 431.

propriedades timeintervalsnode (descontinuado)

Nota: Esse nó foi descontinuado na versão 18 do SPSS Modeler e substituído pelo novo nó Séries Temporais

O nó Intervalos de Tempo especifica os intervalos e cria rótulos (se necessário) para modelar dados de séries temporais. Se os valores não forem uniformemente espaçados, o nó poderá preencher ou agregar valores conforme necessário para gerar um intervalo uniforme entre os registros.

Exemplo

```
node = stream.create("timeintervals", "My node")
node.setPropertyValue("interval_type", "SecondsPerDay")
node.setPropertyValue("days_per_week", 4)
node.setPropertyValue("week_begins_on", "Tuesday")
node.setPropertyValue("hours_per_day", 10)
node.setPropertyValue("day_begins_hour", 7)
node.setPropertyValue("day_begins_minute", 5)
node.setPropertyValue("day_begins_second", 17)
node.setPropertyValue("mode", "Label")
node.setPropertyValue("year_start", 2005)
node.setPropertyValue("month_start", "January")
node.setPropertyValue("day_start", 4)
node.setKeyedPropertyValue("pad", "AGE", "MeanOfRecentPoints")
node.setPropertyValue("agg_mode", "Specify")
node.setPropertyValue("agg_set_default", "Last")
```

Tabela 94. Propriedades de timeintervalsnode		
propriedades `timeintervalsnode`	**Tipo de dados**	**Descrição da propriedade**

`interval_type`	`None`
	`Periods`
	`CyclicPeriods`
	`Years`
	`Quarters`
	`Months`
	`DaysPerWeek`
	`DaysNonPeriodic`
	`HoursPerDay`
	`HoursNonPeriodic`
	`MinutesPerDay`
	`MinutesNonPeriodic`
	`SecondsPerDay`
	`SecondsNonPeriodic`	

`mode`	`Label` `Create`	Especifica se você deseja rotular os registros de modo consecutivo ou construir a série com base em um campo de data, de registro de data e hora ou de tempo especificado.
`field`	*campo*	Ao construir a série a partir dos dados, especifica o campo que indica a data e hora de cada registro.
`period_start`	*Número inteiro*	Especifica o intervalo inicial para períodos ou períodos cíclicos
`cycle_start`	*Número inteiro*	Ciclo inicial para períodos cíclicos.
`year_start`	*Número inteiro*	Para tipos de intervalo onde aplicável, o ano em que cai o primeiro intervalo.
`quarter_start`	*Número inteiro*	Para tipos de intervalo onde aplicável, o trimestre em que cai o primeiro intervalo.

Tabela 94. Propriedades de timeintervalsnode (continuação)		
propriedades `timeintervalsnode`	**Tipo de dados**	**Descrição da propriedade**
month_start	January February March April May June July August September October November December	
day_start	*Número inteiro*	
hour_start	*Número inteiro*	
minute_start	*Número inteiro*	
second_start	*Número inteiro*	
periods_per_cycle	*Número inteiro*	Para períodos cíclicos, número dentro de cada ciclo.
fiscal_year_begins	January February March April May June July August September October November December	Para intervalos trimestrais, especifica o mês em que o ano fiscal começa.
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday Sunday	Para intervalos periódicos (dias por semana, horas por dia, minutos por dia e segundos por dia), especifica o dia em que a semana começa.

Tabela 94. Propriedades de timeintervalsnode (continuação)		
propriedades `timeintervalsnode`	**Tipo de dados**	**Descrição da propriedade**
`day_begins_hour`	*Número inteiro*	Para intervalos periódicos (horas por dia, minutos por dia, segundos por dia), especifica a hora em que o dia começa. Pode ser utilizado em combinação com o `day_begins_minute` e `day_begins_second` para especificar um horário exato, como *8:05:01*. Consulte o exemplo de uso abaixo.
`day_begins_minute`	*Número inteiro*	Para intervalos periódicos (horas por dia, minutos por dia e segundos por dia), especifica o minuto em que o dia começa (por exemplo, o *5* em *8:05*).
`day_begins_second`	*Número inteiro*	Para intervalos periódicos (horas por dia, minutos por dia e segundos por dia), especifica o segundo em que o dia começa (por exemplo, o *17* em *8:05:17*).
`days_per_week`	*Número inteiro*	Para intervalos periódicos (dias por semana, horas por dia, minutos por dia e segundos por dia), especifica o número de dias por semana.
`hours_per_day`	*Número inteiro*	Para intervalos periódicos (horas por dia, minutos por dia e segundos por dia), especifica o número de horas no dia.

`interval_increment`	`1`
	`2`
	`3`
	`4`
	`5`
	`6`
	`10`
	`15`
	`20`
	`30`
		Para minutos por dia e segundos por dia, especifica o número de minutos ou de segundos para incrementar para cada registro.

`field_name_extension`	*sequência*	

Tabela 94. Propriedades de timeintervalsnode (continuação)

propriedades timeintervalsnode	**Tipo de dados**	**Descrição da propriedade**
field_name_extension_as_prefix	*sinalização*	
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	

Tabela 94. Propriedades de timeintervalsnode (continuação)		
propriedades timeintervalsnode	**Tipo de dados**	**Descrição da propriedade**
aggregate	Mean Sum Mode Min Max First Last TrueIfAnyTrue	Especifica o método de agregação para um campo.
pad	Blank MeanOfRecentPoints True False	Especifica o método de preenchimento para um campo.
agg_mode	All Specify	Especifica se deseja agregar ou preencher todos os campos com funções padrão conforme necessário ou especificar os campos e funções a serem utilizados.
agg_range_default	Mean Sum Mode Min Max	Especifica a função padrão a ser utilizada ao agregar os campos contínuos.
agg_set_default	Mode First Last	Especifica a função padrão a ser utilizada ao agregar os campos nominais.

Tabela 94. Propriedades de timeintervalsnode (continuação)		
propriedades `timeintervalsnode`	**Tipo de dados**	**Descrição da propriedade**

`agg_flag_default`	`TrueIfAnyTrue`
	
	`Mode`
	
	`First`
	
	`Last`	

`pad_range_default`	`Blank`
	
	`MeanOfRecentPoints`
		Especifica a função padrão a ser utilizada ao preencher campos contínuos.

`pad_set_default`	`Blank`
	
	`MostRecentValue`	

`pad_flag_default`	`Blank`
	
	`True`
	
	`False`	

`max_records_to_create`	*Número inteiro*	Especifica o número máximo de registros a serem criados ao preencher a série.
`estimation_from_beginning`	*sinalização*	
`estimation_to_end`	*sinalização*	
`estimation_start_offset`	*Número inteiro*	
`estimation_num_holdouts`	*Número inteiro*	
`create_future_records`	*sinalização*	
`num_future_records`	*Número inteiro*	
`create_future_field`	*sinalização*	
`future_field_name`	*sequência*	

Propriedades de transposenode

O nó Transpose troca os dados em linhas e colunas para que os registros se tornem campos e campos se tornem registros.

Exemplo

```
node = stream.create("transpose", "My node")
node.setPropertyValue("transposed_names", "Read")
node.setPropertyValue("read_from_field", "TimeLabel")
```

```
node.setPropertyValue("max_num_fields", "1000")
node.setPropertyValue("id_field_name", "ID")
```

Tabela 95. Propriedades de transposenode

propriedades transposenode	**Tipo de dados**	**Descrição da propriedade**
transpose_method	*Enumeração*	Especifica o método transpor: Normal (normal), CASE para VAR (casetovar) ou VAR para CASE (vartocase).
transposed_names	Prefix Read	Propriedade para o método de transpor Normal. Novos nomes de campos podem ser gerados automaticamente com base em um prefixo especificado ou podem ser lidos a partir de um campo existente nos dados.
prefix	*sequência*	Propriedade para o método de transpor Normal.
num_new_fields	*Número inteiro*	Propriedade para o método de transpor Normal. Ao utilizar um prefixo, especifica o número máximo de novos campos a serem criados.
read_from_field	*campo*	Propriedade para o método de transpor Normal. Campo a partir do qual os nomes são lidos. Este deve ser um campo instanciado ou ocorrerá um erro quando o nó for executado.
max_num_fields	*Número inteiro*	Propriedade para o método de transpor Normal. Ao ler nomes de um campo, especifica um limite superior para evitar a criação de um grande de campos excessivamente.
transpose_type	Numeric String Custom	Propriedade para o método de transpor Normal. Por padrão, apenas os campos contínuos (intervalo numérico) são transpostos, no entanto, é possível escolher um subconjunto customizado de campos numéricos ou transpor todos os campos de sequência.
transpose_fields	*lista*	Propriedade para o método de transpor Normal. Especifica os campos para transpor quando a opção Custom é usada.
id_field_name	*campo*	Propriedade para o método de transpor Normal.
transpose_casetovar_id fields	*campo*	Propriedade para o método transpor do CASE para VAR (casetovar). Aceita vários campos para serem usados como campos de índice. field1 ... fieldN

Tabela 95. Propriedades de transposenode (continuação)		
propriedades transposenode	**Tipo de dados**	**Descrição da propriedade**
transpose_casetovar_columnfields	*campo*	Propriedade para o método transpor do CASE para VAR (casetovar). Aceita vários campos para serem usados como campos de coluna. field1 ... fieldN
transpose_casetovar_valuefields	*campo*	Propriedade para o método transpor do CASE para VAR (casetovar). Aceita vários campos para serem usados como campos de valor. field1 ... fieldN
transpose_vartocase_idfields	*campo*	Propriedade para o método de transpor de VAR para CASE (vartocase). Aceita vários campos para serem usados como campos de variáveis de ID. field1 ... fieldN
transpose_vartocase_valfields	*campo*	Propriedade para o método de transpor de VAR para CASE (vartocase). Aceita vários campos para serem usados como campos de variáveis de valor. field1 ... fieldN

Propriedades typenode

O nó Tipo especifica metadados e propriedades do campo. Por exemplo, é possível especificar um nível de medição (contínuo, nominal, ordinal ou de sinalização) para cada campo, configurar opções para manipular valores omissos e nulos do sistema, configurar a função de um campo para propósitos de modelagem, especificar rótulos de campo e valor e especificar valores para um campo.

Exemplo

```
node = stream.createAt("type", "My node", 50, 50)
node.setKeyedPropertyValue("check", "Cholesterol", "Coerce")
node.setKeyedPropertyValue("direction", "Drug", "Input")
node.setKeyedPropertyValue("type", "K", "Range")
node.setKeyedPropertyValue("values", "Drug", ["drugA", "drugB", "drugC",
"drugD", "drugX",
 "drugY", "drugZ"])
node.setKeyedPropertyValue("null_missing", "BP", False)
node.setKeyedPropertyValue("whitespace_missing", "BP", False)
node.setKeyedPropertyValue("description", "BP", "Blood Pressure")
node.setKeyedPropertyValue("value_labels", "BP", [["HIGH", "High Blood
Pressure"],
 ["NORMAL", "normal blood pressure"]])
```

Observe que, em alguns casos, poderá ser necessário instanciar totalmente o nó Tipo para outros nós para funcionar corretamente, como a propriedade `fields from` do nó Configurar para Sinalizador. É possível simplesmente conectar um nó Tabela e executá-lo para instanciar os campos:

```
tablenode = stream.createAt("table", "Table node", 150, 50)
stream.link(node, tablenode)
tablenode.run(None)
stream.delete(tablenode)
```

Tabela 96. Propriedades typenode

propriedades typenode	**Tipo de dados**	**Descrição da propriedade**

`direction`	`Input`
	
	`Target`
	
	`Both`
	
	`None`
	
	`Partition`
	
	`Split`
	
	`Frequency`
	
	`RecordID`
		Propriedade definida como chave para funções de campo.
		
		Nota: Os valores de `In` e `Out` estão agora descontinuados. O suporte para eles poderá ser retirado em uma liberação futura.

`type`	`Range`
	
	`Flag`
	
	`Set`
	
	`Typeless`
	
	`Discrete`
	
	`OrderedSet`
	
	`Default`
		Nível de medição do campo (anteriormente chamado de "tipo" de campo). Configurando `type` para `Default` limpará qualquer configuração de parâmetro `values` e se `value_mode` tiver o valor `Specify`, ele será reconfigurado para `Read`. Se `value_mode` estiver configurado como `Pass` ou `Read`, a configuração de `type` não afetará `value_mode`.
		
		Nota: Os tipos de dados utilizados internamente diferem dos tipos visíveis no nó de tipo. A correspondência é a seguinte: Intervalo-> Conjunto Contínuo-> Nominal OrderedSet -> Ordinal Discreto-> Categórico

Tabela 96. Propriedades typenode (continuação)

propriedades typenode	Tipo de dados	Descrição da propriedade
storage	Unknown String Integer Real Time Date Timestamp	Propriedade definida como chave somente leitura para tipo de armazenamento de campo.
check	None Nullify Coerce Discard Warn Abort	Propriedade definida como chave para verificação de tipo e de intervalo de campo
values	[*value value*]	Para campos contínuos, o primeiro valor é o mínimo e o último valor é o máximo. Para os campos nominais, especifique todos os valores. Para campos de sinalização, o primeiro valor representa *false* e o último valor representa *true*. A configuração desta propriedade configura automaticamente a propriedade value_mode para Specify.
value_mode	Read Pass Read+ Current Specify	Determina como os valores são configurados. Note que não é possível configurar essa propriedade como Specify diretamente; para usar valores específicos, configure a propriedade values.

Tabela 96. Propriedades typenode (continuação)

propriedades typenode	Tipo de dados	Descrição da propriedade
extend_values	*sinalização*	Aplica-se quando value_mode é configurado como Read. Configure como T para incluir valores recém-lidos em quaisquer valores existentes para o campo. Configure como F para descartar os valores existentes em favor dos valores recém-lidos.
enable_missing	*sinalização*	Quando configurado como T, ativa o rastreamento de valores ausentes para o campo.
missing_values	[*value value ...*]	Especifica valores de dados que denotam dados ausentes.
range_missing	*sinalização*	Especifica se um intervalo de valores omissos (em branco) é definido para um campo.
missing_lower	*sequência*	Quando range_missing é true, especifica o limite inferior do limite de valor ausente.
missing_upper	*sequência*	Quando range_missing é true, especifica o limite superior do intervalo de valor ausente.
null_missing	*sinalização*	Quando configurado como T, *nulos* (valores indefinidos que são exibidos como $null$ no software) são considerados valores ausentes.
whitespace_missing	*sinalização*	Quando configurados como T, valores contendo apenas espaço em branco (espaços, guias e novas linhas) são considerados valores ausentes.
description	*sequência*	Especifica a descrição de um campo.
value_labels	*[[Value LabelString] [Value LabelString] ...]*	Utilizado para especificar rótulos para pares de valores.
display_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exibido (aplica-se apenas aos campos com armazenamento REAL). Um valor de –1 usará o padrão do fluxo.
export_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exportado (aplica-se apenas aos campos com armazenamento REAL). Um valor de –1 usará o padrão do fluxo.

Tabela 96. Propriedades typenode (continuação)

propriedades typenode	Tipo de dados	Descrição da propriedade
decimal_separator	DEFAULT PERIOD COMMA	Configura o separador decimal para o campo (aplica-se apenas aos campos com armazenamento REAL).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Configura o formato de data para o campo (aplica-se apenas aos campos com o armazenamento DATE ou TIMESTAMP).
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Configura o formato de hora para o campo (aplica-se apenas aos campos com armazenamento TIME ou TIMESTAMP).
number_format	DEFAULT STANDARD SCIENTIFIC CURRENCY	Configura o formato de exibição de número para o campo.

Tabela 96. Propriedades typenode (continuação)

propriedades typenode	Tipo de dados	Descrição da propriedade
standard_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exibido em formato padrão. Um valor de –1 usará o padrão do fluxo. Observe que o slot display_places existente também altera isso, mas agora foi descontinuado.
scientific_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exibido no formato científico. Um valor de –1 usará o padrão do fluxo.
currency_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exibido no formato de moeda. Um valor de –1 usará o padrão do fluxo.
grouping_symbol	DEFAULT NONE LOCALE PERIOD COMMA SPACE	Configura o símbolo de agrupamento para o campo.
column_width	*Número inteiro*	Configura a largura da coluna para o campo. Um valor de –1 configurará a largura da coluna para Auto.
justify	AUTO CENTER LEFT RIGHT	Configura a justificação da coluna para o campo.

Tabela 96. Propriedades typenode (continuação)		
propriedades typenode	**Tipo de dados**	**Descrição da propriedade**
measure_type	Range / MeasureType.RANGE Discrete / MeasureType.DISCRETE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS Collection / MeasureType.COLLECTION Geospatial / MeasureType.GEOSPATIAL	Esta propriedade definida como chave é semelhante a type na medida em que pode ser usada para definir a medição associada ao campo. O que é diferente é que no script Python, a função de configurador também pode ser passada um dos valores MeasureType enquanto a função que obtém sempre retornará nos valores MeasureType.
collection_measure	Range / MeasureType.RANGE Flag / MeasureType.FLAG Set / MeasureType.SET OrderedSet / MeasureType.ORDERED_SET Typeless / MeasureType.TYPELESS	Para campos de coleção (listas com uma profundidade 0), essa propriedade definida como chave define o tipo de medição associado aos valores subjacentes.
geo_type	Point MultiPoint LineString MultiLineString Polygon MultiPolygon	Para campos geoespaciais, esta propriedade definida como chave define o tipo de objeto geoespacial representado por este campo. Isso deverá estar consistente com a profundidade da lista dos valores.
has_coordinate_ system	*Booleano*	Para campos geoespaciais, essa propriedade define se esse campo tem um sistema de coordenadas
coordinate_ system	*sequência*	Para campos geoespaciais, esta propriedade definida como chave define o sistema de coordenadas para este campo.

Tabela 96. Propriedades typenode (continuação)

propriedades typenode	Tipo de dados	Descrição da propriedade
custom_storage_ type	Unknown / MeasureType.UNKNOWN String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP List / MeasureType.LIST	Esta propriedade definida como chave é semelhante a custom_storage na medida que pode ser usada para definir o armazenamento de substituição para o campo. O que é diferente é que no script Python, a função de configurador também pode ser passada um dos valores StorageType enquanto a função que obtém sempre retornará nos valores StorageType.
custom_list_ storage_type	String / MeasureType.STRING Integer / MeasureType.INTEGER Real / MeasureType.REAL Time / MeasureType.TIME Date / MeasureType.DATE Timestamp / MeasureType.TIMESTAMP	Para campos de lista, esta propriedade definida como chave especifica o tipo de armazenamento dos valores subjacentes.
custom_list_depth	*Número inteiro*	Para campos de lista, esta propriedade definida como chave especifica a profundidade do campo
max_list_length	*Número inteiro*	Disponível apenas para dados com um nível de medição de *Geoespacial* ou *Coleção*. Configure o comprimento máximo da lista ao especificar o número de elementos que a lista pode conter.
max_string_ length	*Número inteiro*	Apenas disponível para dados *sem tipo* e usado quando você está gerando SQL para criar uma tabela. Digite o valor da maior sequência em seus dados; isso gera uma coluna na tabela que é grande o suficiente para conter a sequência.

Capítulo 12. Propriedades do Nó de Gráfico

Propriedades Comuns do Nó Gráfico

Esta seção descreve as propriedades disponíveis para os nós de gráfico, incluindo propriedades comuns e as propriedades que são específicas para cada tipo de nó.

Tabela 97. Propriedades comuns do nó de gráfico

Propriedades comuns do nó de gráfico	**Tipo de dados**	**Descrição da propriedade**
`title`	*sequência*	Especifica o título. Exemplo: "Este é um título".
`caption`	*sequência*	Especifica a legenda. Exemplo: "Esta é uma legenda".

`output_mode`	`Screen`
	
	`File`
		Especifica se a saída do nó de gráfico é exibida ou gravada em um arquivo.

`output_format`	`BMP`
	
	`JPEG`
	
	`PNG`
	
	`HTML`
	
	`output` (*.cou*)
		Especifica o tipo de saída. O tipo exato de saída permitido para cada nó varia.

`full_filename`	*sequência*	Especifica o caminho de destino e o nome de arquivo para a saída gerada a partir do nó de gráfico.
`use_graph_size`	*sinalização*	Controla se o gráfico será dimensionado explicitamente, utilizando as propriedades de largura e altura abaixo. Isso afeta apenas os gráficos que forem gerados para a tela. Não disponível para o nó Distribuição.
`graph_width`	*number*	Quando `use_graph_size` for `True`, configura a largura do gráfico em pixels.
`graph_height`	*number*	Quando `use_graph_size` for `True`, configura a altura do gráfico em pixels.

Desativando campos opcionais

Campos opcionais, como um campo de sobreposição para gráficos, podem ser desativados ao configurar o valor da propriedade para `" "` (sequência vazia), conforme mostrado no exemplo a seguir:

```
plotnode.setPropertyValue("color_field", "")
```

Especificando cores

As cores de títulos, legendas, planos de fundo e rótulos podem ser especificadas utilizando as sequências hexadecimais começando com o símbolo hash (#). Por exemplo, para configurar o plano de fundo do gráfico para azul-celeste, a seguinte instrução é utilizada:

```
mygraphnode.setPropertyValue("graph_background", "#87CEEB")
```

Aqui, os dois primeiros dígitos, 87, especificam o conteúdo em vermelho; os dois dígitos do meio, CE, especificam o conteúdo em verde e os dois últimos dígitos, EB, especificam o conteúdo em azul. Cada dígito pode ter um valor no intervalo de 0 a 9 ou A a F. Juntos, esses valores podem especificar uma cor vermelho-verde-azul, ou RGB.

Nota: Ao especificar cores em RGB, é possível usar o Seletor de Campo na interface com o usuário para determinar o código de cor correto. Basta passar o mouse sobre a cor para ativar uma ToolTip com as informações desejadas.

Propriedades de collectionnode

O nó de Coleção mostra a distribuição de valores para um campo numérico com relação aos valores de outro campo. (Ele cria gráficos semelhantes a histogramas). Ele é útil para ilustrar uma variável ou campo cujos valores se alteram ao longo do tempo. Usando gráficos 3D, também é possível incluir um eixo simbólico exibindo distribuições por categoria.

Exemplo

```
node = stream.create("collection", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("collect_field", "Drug")
node.setPropertyValue("over_field", "Age")
node.setPropertyValue("by_field", "BP")
node.setPropertyValue("operation", "Sum")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1)
node.setPropertyValue("range_max", 100)
node.setPropertyValue("bins", "ByNumber")
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 5)
```

Tabela 98. Propriedades collectionnode

propriedades collectionnode	**Tipo de dados**	**Descrição da propriedade**
over_field	*campo*	
over_label_auto	*sinalização*	
over_label	*sequência*	
collect_field	*campo*	
collect_label_auto	*sinalização*	
collect_label	*sequência*	

Tabela 98. Propriedades collectionnode (continuação)		
propriedades collectionnode	**Tipo de dados**	**Descrição da propriedade**
three_D	*sinalização*	
by_field	*campo*	
by_label_auto	*sinalização*	
by_label	*sequência*	
operation	Sum Mean Min Max SDev	
color_field	*sequência*	
panel_field	*sequência*	
animation_field	*sequência*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
use_grid	*sinalização*	
graph_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.

Propriedades de distributionnode

O nó Distribuição mostra a ocorrência de valores simbólicos (categóricos), como tipo ou gênero da hipoteca. O nó Distribuição pode ser usado geralmente para mostrar desbalanceamentos nos dados, que poderão então ser corrigidos utilizando um nó Balanceamento antes de criar um modelo.

Exemplo

```
node = stream.create("distribution", "My node")
# "Plot" tab
node.setPropertyValue("plot", "Flags")
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("normalize", True)
node.setPropertyValue("sort_mode", "ByOccurence")
node.setPropertyValue("use_proportional_scale", True)
```

Tabela 99. Propriedades de distributionnode

propriedades distributionnode	**Tipo de dados**	**Descrição da propriedade**
plot	SelectedFields Flags	
x_field	*campo*	
color_field	*campo*	Campo de sobreposição.
normalize	*sinalização*	
sort_mode	ByOccurence Alphabetic	
use_proportional_scale	*sinalização*	

Propriedades de evaluationnode

O nó Avaliação ajuda a avaliar e comparar modelos preditivos. O gráfico de avaliação mostra quão bem os modelos preveem resultados específicos. Ele classifica os registros com base no valor previsto e na confiança da predição. Ele divide os registros em grupos de tamanhos iguais (**quantis**) e, em seguida, representa o valor do critério de negócios para cada quantil do mais alto para o mais baixo. Diversos modelos são mostrados como linhas separadas no gráfico.

Exemplo

```
node = stream.create("evaluation", "My node")
# "Plot" tab
node.setPropertyValue("chart_type", "Gains")
node.setPropertyValue("cumulative", False)
node.setPropertyValue("field_detection_method", "Name")
node.setPropertyValue("inc_baseline", True)
node.setPropertyValue("n_tile", "Deciles")
node.setPropertyValue("style", "Point")
node.setPropertyValue("point_type", "Dot")
node.setPropertyValue("use_fixed_cost", True)
node.setPropertyValue("cost_value", 5.0)
node.setPropertyValue("cost_field", "Na")
node.setPropertyValue("use_fixed_revenue", True)
node.setPropertyValue("revenue_value", 30.0)
node.setPropertyValue("revenue_field", "Age")
node.setPropertyValue("use_fixed_weight", True)
node.setPropertyValue("weight_value", 2.0)
node.setPropertyValue("weight_field", "K")
```

Tabela 100. Propriedades de evaluationnode		
propriedades evaluationnode	**Tipo de dados**	**Descrição da propriedade**
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	*sinalização*	
field_detection_method	Metadata Name	
use_fixed_cost	*sinalização*	
cost_value	*number*	
cost_field	*sequência*	
use_fixed_revenue	*sinalização*	
revenue_value	*number*	
revenue_field	*sequência*	
use_fixed_weight	*sinalização*	
weight_value	*number*	
weight_field	*campo*	
n_tile	Quartiles Quintles Deciles Vingtiles Percentiles 1000-tiles	
cumulative	*sinalização*	
style	Line Point	

Tabela 100. Propriedades de evaluationnode (continuação)		
propriedades evaluationnode	**Tipo de dados**	**Descrição da propriedade**

point_type	`Rectangle`
	`Dot`
	`Triangle`
	`Hexagon`
	`Plus`
	`Pentagon`
	`Star`
	`BowTie`
	`HorizontalDash`
	`VerticalDash`
	`IronCross`
	`Factory`
	`House`
	`Cathedral`
	`OnionDome`
	`ConcaveTriangle`
	`OblateGlobe`
	`CatEye`
	`FourSidedPillow`
	`RoundRectangle`
	`Fan`	

export_data	*sinalização*	
data_filename	*sequência*	
delimiter	*sequência*	
new_line	*sinalização*	
inc_field_names	*sinalização*	
inc_best_line	*sinalização*	
inc_business_rule	*sinalização*	
business_rule_condition	*sequência*	
plot_score_fields	*sinalização*	
score_fields	*[field1 ... fieldN]*	
target_field	*campo*	
use_hit_condition	*sinalização*	
hit_condition	*sequência*	
use_score_expression	*sinalização*	
score_expression	*sequência*	
caption_auto	*sinalização*	

Propriedades de graphboardnode

O nó Elemento do Gráfico oferece muitos tipos diferentes de gráficos em um único nó. Utilizando esse nó, é possível escolher os campos de dados que desejar explorar e, em seguida, selecionar um gráfico a partir dos disponíveis para os dados selecionados. O nó filtra automaticamente todos os tipos de gráficos que não funcionariam com as opções de campo.

Nota: Se você configurar uma propriedade que não for válida para o tipo de gráfico (por exemplo, especificar `y_field` para um histograma), essa propriedade será ignorada.

Nota: Na IU, na guia Detalhado de muitos tipos de gráficos diferentes, há um campo **Summary** que não é atualmente suportado pelo script.

Exemplo

```
node = stream.create("graphboard", "My node")
node.setPropertyValue("graph_type", "Line")
node.setPropertyValue("x_field", "K")
node.setPropertyValue("y_field", "Na")
```

Tabela 101. Propriedades de graphboardnode

propriedades graphboard	Tipo de dados	Descrição da propriedade
graph_type	2DDotplot 3DArea 3DBar 3DDensity 3DHistogram 3DPie 3DScatterplot Area ArrowMap Bar BarCounts BarCountsMap BarMap BinnedScatter Boxplot Bubble ChoroplethMeans ChoroplethMedians ChoroplethSums ChoroplethValues	Identifica o tipo de gráfico.

Tabela 101. Propriedades de graphboardnode (continuação)

propriedades graphboard	Tipo de dados	Descrição da propriedade
	ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap	

Tabela 101. Propriedades de graphboardnode (continuação)		
propriedades graphboard	**Tipo de dados**	**Descrição da propriedade**
	PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	
x_field	*campo*	Especifica um rótulo customizado para o eixo *x*. Disponível apenas para rótulos.
y_field	*campo*	Especifica um rótulo customizado para o eixo *y*. Disponível apenas para rótulos.
z_field	*campo*	Usado em alguns gráficos 3D.
color_field	*campo*	Usado nos mapas de utilização.
size_field	*campo*	Utilizado em gráficos de bolha.
categories_field	*campo*	
values_field	*campo*	
rows_field	*campo*	
columns_field	*campo*	
fields	*campo*	
start_longitude_field	*campo*	Usado com setas em um mapa de referência
end_longitude_field	*campo*	
start_latitude_field	*campo*	
end_latitude_field	*campo*	
data_key_field	*campo*	Usado em vários mapas.
panelrow_field	*sequência*	
panelcol_field	*sequência*	
animation_field	*sequência*	

Tabela 101. Propriedades de graphboardnode (continuação)		
propriedades graphboard	**Tipo de dados**	**Descrição da propriedade**
longitude_field	*campo*	Usado com coordenadas em mapas.
latitude_field	*campo*	
map_color_field	*campo*	

Propriedades de histogramnode

O nó Histograma mostra a ocorrência de valores para campos numéricos. Ele é normalmente utilizado para explorar os dados antes de manipulações e construções de modelo. Semelhante ao nó Distribuição, o nó Histograma revela frequentemente desequilíbrios nos dados.

Exemplo

```
node = stream.create("histogram", "My node")
# "Plot" tab
node.setPropertyValue("field", "Drug")
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("panel_field", "Sex")
node.setPropertyValue("animation_field", "")
# "Options" tab
node.setPropertyValue("range_mode", "Automatic")
node.setPropertyValue("range_min", 1.0)
node.setPropertyValue("range_max", 100.0)
node.setPropertyValue("num_bins", 10)
node.setPropertyValue("bin_width", 10)
node.setPropertyValue("normalize", True)
node.setPropertyValue("separate_bands", False)
```

Tabela 102. propriedades histogramnode		
propriedades histogramnode	**Tipo de dados**	**Descrição da propriedade**
field	*campo*	
color_field	*campo*	
panel_field	*campo*	
animation_field	*campo*	
range_mode	Automatic UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
normalize	*sinalização*	

Tabela 102. propriedades histogramnode (continuação)		
propriedades histogramnode	**Tipo de dados**	**Descrição da propriedade**
separate_bands	*sinalização*	
x_label_auto	*sinalização*	
x_label	*sequência*	
y_label_auto	*sinalização*	
y_label	*sequência*	
use_grid	*sinalização*	
graph_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
normal_curve	*sinalização*	Indica se a curva de distribuição normal deve ser mostrada na saída.

propriedades de mapvisualization

O nó Visualização de Mapa pode aceitar diversas conexões de entrada e exibir dados geoespaciais em um mapa como uma série de camadas. Cada camada é um campo geoespacial único, por exemplo, a camada de base pode ser um mapa de um país e, acima dele, você pode ter uma camada para estradas, uma camada para rios e uma camada para as cidades.

Tabela 103. propriedades de mapvisualization		
propriedades mapvisualization	**Tipo de dados**	**Descrição da propriedade**
tag	*sequência*	Configura o nome da tag para a entrada A tag padrão é um número baseado na ordem em que as entradas foram conectadas ao nó (a primeira tag de conexão é 1, a segunda tag de conexão é 2, etc.
layer_field	*campo*	Seleciona qual campo geográfico do conjunto de dados é exibido como uma camada no mapa. A seleção padrão é baseada na seguinte ordem de classificação: • Primeiro Ponto • LineString • Polígono • Multiponto • MultiLinestring • Último- MultiPolygon Se houver dois campos com o mesmo tipo de medição, o primeiro campo alfabético (por nome) será selecionado por padrão.

Tabela 103. propriedades de mapvisualization (continuação)		
propriedades `mapvisualization`	**Tipo de dados**	**Descrição da propriedade**
`color_type`	*Booleano*	Especifica se uma cor padrão é aplicada a todos os recursos do campo geográfico ou a um campo de sobreposição que colore os recursos com base em valores de outro campo no conjunto de dados Os valores possíveis são `standard` ou `overlay`. O padrão é `standard`.
`color`	*sequência*	Se `standard` for selecionado para `color_type`, a lista suspensa conterá a mesma paleta de cores que a ordem de cores da categoria do gráfico na guia Exibição das opções de usuário O padrão é a cor da categoria do gráfico 1
`color_field`	*campo*	Se `overlay` for selecionado para `color_type`, o menu suspenso conterá todos os campos do mesmo conjunto de dados que o geo-campo selecionado como a camada
`symbol_type`	*Booleano*	Especifica se um símbolo padrão é aplicado a todos os registros do campo geográfico, ou um símbolo de sobreposição que altera o ícone de símbolo para os pontos com base em valores de outro campo no conjunto de dados Os valores possíveis são `standard` ou `overlay`. O padrão é `standard`.
`symbol`	*sequência*	Se `standard` for selecionado para `symbol_type`, o menu suspenso conterá uma seleção de símbolos que podem ser usados para exibir pontos no mapa
`symbol_field`	*campo*	Se `overlay` for selecionado para `symbol_type`, o menu suspenso conterá todos os campos nominal, ordinal ou categórico do mesmo conjunto de dados que o campo geográfico selecionado como a camada.
`size_type`	*Booleano*	Especifica se um tamanho padrão é aplicado a todos os registros do campo geográfico ou a um tamanho de sobreposição que altera o tamanho do ícone de símbolo ou a espessura da linha com base em valores de outro campo no conjunto de dados Os valores possíveis são `standard` ou `overlay`. O padrão é `standard`.

Tabela 103. propriedades de mapvisualization (continuação)		
propriedades `mapvisualization`	**Tipo de dados**	**Descrição da propriedade**
`size`	*sequência*	Se `standard` for selecionado para `size_type`, para `point` ou `multipoint`, o menu suspenso conterá uma seleção de tamanhos para o símbolo selecionado Para `linestring` ou `multilinestring`, o menu suspenso contém uma seleção de espessuras de linha
`size_field`	*campo*	Se `overlay` for selecionado para `size_type`, o menu suspenso conterá todos os campos do mesmo conjunto de dados que o campo geográfico selecionado como a camada
`transp_type`	*Booleano*	Especifica se uma transparência padrão é aplicada a todos registros do campo geográfico ou uma transparência de sobreposição que altera o nível de transparência para o símbolo, linha ou polígono com base em valores de outro campo no conjunto de dados. Os valores possíveis são `standard` ou `overlay`. O padrão é `standard`.
`transp`	*Número inteiro*	Se `standard` for selecionado para `transp_type`, a lista suspensa conterá uma seleção de níveis de transparência iniciando em `0%` (opaco) e aumentando para `100%` (transparente) em incrementos de `10%`. Configura a transparência de pontos, linhas ou polígonos no mapa Se `overlay` for selecionado para `size_type`, o menu suspenso conterá todos os campos do mesmo conjunto de dados que o campo geográfico selecionado como a camada Para `points`, `multipoints`, `linestringse` `multilinestrings`, `polygons` e `multipolygons` (que são a camada inferior), o padrão é `0%`. Para `polygons` e `multipolygons` que não são a camada inferior, o padrão é `50%` (para evitar obscurecer camadas abaixo desses polígonos).
`transp_field`	*campo*	Se `overlay` for selecionado para `transp_type`, o menu suspenso conterá todos os campos do mesmo conjunto de dados que o campo geográfico selecionado como a camada

Tabela 103. propriedades de mapvisualization (continuação)		
propriedades mapvisualization	**Tipo de dados**	**Descrição da propriedade**
data_label_field	*campo*	Especifica o campo a ser usado como rótulos de dados no mapa Por exemplo, se a camada à qual essa configuração é aplicada for uma camada de polígono, o rótulo de dados poderá ser o campo name -contendo o nome de cada polígono. Portanto, selecionar o campo name aqui resultaria em esses nomes sendo exibidos no mapa
use_hex_binning	*Booleano*	Ativa a categorização hexadecimal e ativa todas as listas suspensas de agregação Essa configuração é desativada por padrão
color_aggregation e transp_aggregation	*sequência*	Se você selecionar um campo de sobreposição para uma camada de pontos usando a categorização hexadecimal, todos os valores para esse campo deverão ser agregados para todos os pontos dentro do hexágono. Portanto, você deve especificar uma função de agregação para quaisquer campos de sobreposição que deseja aplicar ao mapa. As funções de agregação disponíveis são: Contínuo (armazenamento Real ou Integer): • Soma • Média • Mín. • Máx. • Mediana • 1º Quartil • 3º Quartil Contínuo (armazenamento de Data, Hora ou Registro de Data e Hora): • Média • Mín. • Máx. Nominal / Categórica: • Modo • Mín. • Máx. Sinalização: • Verdadeiro se algum for verdadeiro • Falso se algum falso

Tabela 103. propriedades de mapvisualization (continuação)		
propriedades mapvisualization	**Tipo de dados**	**Descrição da propriedade**
custom_storage	*sequência*	Configura o tipo de armazenamento geral do campo O padrão é List. Se List for especificado, os seguintes controles custom_value_storage e list_depth serão desativados.
custom_value_storage	*sequência*	Configura os tipos de armazenamento dos elementos na lista em vez de para o campo como um todo O padrão é Real.
list_depth	*Número inteiro*	Configura a profundidade do campo de lista A profundidade necessária depende do tipo de geofield, seguindo estes critérios: • Ponto-0 • LineString - 1 • Polígono-2 • Multiponto-1 • Sequência de MultiLine-2 • Multipolígono-3 Você deve saber o tipo de campo geoespacial que está convertendo de volta para uma lista e a profundidade necessária para esse tipo de campo. Se configurado incorretamente, o campo não poderá ser usado.. O valor padrão é 0, mínimo é 0e máximo é 10.

Propriedades de multiplotnode

O nó Multigráficos cria uma representação que exibe diversos campos *Y* em um único campo *X*. Os campos *Y* são representados como linhas coloridas, em que cada linha é equivalente a um nó Gráfico com Estilo configurado para **Linha** e Modo X configurado para **Classificar**. Os multigráficos são úteis quando desejar explorar a flutuação de diversas variáveis ao longo do tempo.

Exemplo

```
node = stream.create("multiplot", "My node")
# "Plot" tab
node.setPropertyValue("x_field", "Age")
node.setPropertyValue("y_fields", ["Drug", "BP"])
node.setPropertyValue("panel_field", "Sex")
# "Overlay" section
node.setPropertyValue("animation_field", "")
node.setPropertyValue("tooltip", "test")
node.setPropertyValue("normalize", True)
node.setPropertyValue("use_overlay_expr", False)
node.setPropertyValue("overlay_expression", "test")
```

```
node.setPropertyValue("records_limit", 500)
node.setPropertyValue("if_over_limit", "PlotSample")
```

Tabela 104. Propriedades de multiplotnode

propriedades multiplotnode	Tipo de dados	Descrição da propriedade
x_field	*campo*	
y_fields	*lista*	
panel_field	*campo*	
animation_field	*campo*	
normalize	*sinalização*	
use_overlay_expr	*sinalização*	
overlay_expression	*sequência*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*sinalização*	
x_label	*sequência*	
y_label_auto	*sinalização*	
y_label	*sequência*	
use_grid	*sinalização*	
graph_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.

Propriedades de plotnode

O nó Gráfico mostra o relacionamento entre os campos numéricos. É possível criar um gráfico utilizando pontos (gráfico de dispersão) ou linhas.

Exemplo

```
node = stream.create("plot", "My node")
# "Plot" tab
node.setPropertyValue("three_D", True)
node.setPropertyValue("x_field", "BP")
node.setPropertyValue("y_field", "Cholesterol")
node.setPropertyValue("z_field", "Drug")
# "Overlay" section
node.setPropertyValue("color_field", "Drug")
node.setPropertyValue("size_field", "Age")
node.setPropertyValue("shape_field", "")
node.setPropertyValue("panel_field", "Sex")
```

```
node.setPropertyValue("animation_field", "BP")
node.setPropertyValue("transp_field", "")
node.setPropertyValue("style", "Point")
# "Output" tab
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "JPEG")
node.setPropertyValue("full_filename", "C:/temp/graph_output/
plot_output.jpeg")
```

Tabela 105. Propriedades de plotnode

propriedades plotnode	**Tipo de dados**	**Descrição da propriedade**
x_field	*campo*	Especifica um rótulo customizado para o eixo *x*. Disponível apenas para rótulos.
y_field	*campo*	Especifica um rótulo customizado para o eixo *y*. Disponível apenas para rótulos.
three_D	*sinalização*	Especifica um rótulo customizado para o eixo *y*. Disponível apenas para os rótulos nos gráficos 3D.
z_field	*campo*	
color_field	*campo*	Campo de sobreposição.
size_field	*campo*	
shape_field	*campo*	
panel_field	*campo*	Especifica um campo nominal ou de sinalização para uso ao criar um gráfico separado para cada categoria. Os gráficos são agrupados em painéis em uma janela de saída.
animation_field	*campo*	Especifica um campo nominal ou de sinalização para ilustrar categorias de valores de dados ao criar uma série de gráficos exibidos em sequência utilizando animação.
transp_field	*campo*	Especifica um campo para ilustrar as categorias de valores de dados usando um nível diferente de transparência para cada categoria. Não disponível para gráficos de linha.
overlay_type	None Smoother Function	Especifica se uma função de sobreposição ou um suavizador LOESS é exibido.
overlay_expression	*sequência*	Especifica a expressão usada quando overlay_type é configurado como Function.
style	Point Line	

Tabela 105. Propriedades de plotnode (continuação)

propriedades plotnode	Tipo de dados	Descrição da propriedade
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
x_mode	Sort Overlay AsRead	
x_range_mode	Automatic UserDefined	
x_range_min	*number*	
x_range_max	*number*	
y_range_mode	Automatic UserDefined	
y_range_min	*number*	
y_range_max	*number*	
z_range_mode	Automatic UserDefined	
z_range_min	*number*	
z_range_max	*number*	
jitter	*sinalização*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	

Tabela 105. Propriedades de plotnode (continuação)		
propriedades plotnode	**Tipo de dados**	**Descrição da propriedade**
x_label_auto	*sinalização*	
x_label	*sequência*	
y_label_auto	*sinalização*	
y_label	*sequência*	
z_label_auto	*sinalização*	
z_label	*sequência*	
use_grid	*sinalização*	
graph_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
use_overlay_expr	*sinalização*	Descontinuado em favor de overlay_type.

Propriedades de timeplotnode

O nó Gráfico de Tempo exibe um ou mais conjuntos de dados de séries temporais. Geralmente, você primeiro usaria um nó Intervalos de Tempo para criar um campo *TimeLabel* , que seria usado para rotular o eixo *x*

Exemplo

```
node = stream.create("timeplot", "My node")
node.setPropertyValue("y_fields", ["sales", "men", "women"])
node.setPropertyValue("panel", True)
node.setPropertyValue("normalize", True)
node.setPropertyValue("line", True)
node.setPropertyValue("smoother", True)
node.setPropertyValue("use_records_limit", True)
node.setPropertyValue("records_limit", 2000)
# Appearance settings
node.setPropertyValue("symbol_size", 2.0)
```

Tabela 106. Propriedades de timeplotnode		
propriedades timeplotnode	**Tipo de dados**	**Descrição da propriedade**
plot_series	Series Models	
use_custom_x_field	*sinalização*	
x_field	*campo*	
y_fields	*lista*	
panel	*sinalização*	
normalize	*sinalização*	
line	*sinalização*	

Tabela 106. Propriedades de timeplotnode (continuação)		
propriedades timeplotnode	**Tipo de dados**	**Descrição da propriedade**
points	*sinalização*	
point_type	Rectangle Dot Triangle Hexagon Plus Pentagon Star BowTie HorizontalDash VerticalDash IronCross Factory House Cathedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Fan	
smoother	*sinalização*	É possível incluir suavizadores no gráfico apenas se você configurar panel para True.
use_records_limit	*sinalização*	
records_limit	*Número inteiro*	
symbol_size	*number*	Especifica um tamanho do símbolo.
panel_layout	Horizontal Vertical	

eplotnode Propriedades

O nó E-Plot (Beta) mostra a relação entre campos numéricos. Ele é semelhante ao nó do Plot, mas suas opções diferem e sua saída usa uma nova interface de grafite específica para este nó. Use o nódulo de nível beta para brincar com novos recursos de grafismo.

Tabela 107. propriedades do eplotnode		
propriedades eplotnode	**Tipo de dados**	**Descrição da propriedade**
x_field	*sequência*	Especifique o campo para exibir no eixo X horizontal.
y_field	*sequência*	Especifique o campo para exibir no eixo Y vertical.
color_field	*sequência*	Especifique o campo a ser usado para a sobreposição do mapa de cores na saída, se desejado
size_field	*sequência*	Especifique o campo a ser usado para a sobreposição do mapa de tamanho na saída, se desejado

Tabela 107. propriedades do eplotnode (continuação)		
propriedades eplotnode	**Tipo de dados**	**Descrição da propriedade**
shape_field	*sequência*	Especifique o campo a ser usado para a sobreposição do mapa de forma na saída, se desejado
interested_fields	*sequência*	Especifique os campos que deseja incluir na saída.
records_limit	*Número inteiro*	Especifique um número para o número máximo de registros para plotar na saída.. 2000 é o padrão.
if_over_limit	*Booleano*	Especifique se usar a opção Sample ou a opção Use all data se o records_limit for ultrapassado. Sample é o padrão e ele amostra aleatoriamente os dados até atingir o records_limit. Se você especificar Use all data para ignorar o records_limit e plotar todos os pontos de dados, observe que isso pode diminuir drasticamente o desempenho.

Propriedades tsnenode

t-Distributed Stochastic Neighbor Embedding (t-SNE) é uma ferramenta para visualização de dados de alta dimensionais. Ele converte afinidades de pontos de dados em probabilidades. Este nó t-SNE em SPSS Modeler é implementado em Python e requer a biblioteca scikit-learn© Python .

Tabela 108. propriedades tsnenode		
propriedades tsnenode	**Tipo de dados**	**Descrição da propriedade**
mode_type	*sequência*	Especifique simple ou modo expert .
n_components	*sequência*	Dimensão do espaço integrado (2D ou 3D). Especifique 2 ou 3. O padrão é 2.
method	*sequência*	Especifique barnes_hut ou exact. O padrão é barnes_hut.
init	*sequência*	Inicialização da integração... Especifique random ou pca. O padrão é random.
target_field Renomeado para target a partir da versão 18.2.1.1	*sequência*	Nome do campo de destino.. Será um colormap no gráfico de saída. O gráfico usará uma cor se nenhum campo de destino for especificado..
perplexity	*Valor flutuante*	A perplexidade está relacionada ao número de vizinhos mais próximos usados em outros algoritmos de aprendizagem. Conjuntos de dados maiores geralmente requerem uma maior perplexidade. Considere selecionar um valor entre 5 e 50. O padrão é 30.

Tabela 108. propriedades tsnenode (continuação)		
propriedades tsnenode	**Tipo de dados**	**Descrição da propriedade**
early_exaggeration	*Valor flutuante*	Controla o quão apertado os clusters naturais no espaço original estão no espaço integrado e quanto espaço haverá entre eles. O padrão é 12.0.
learning_rate	*Valor flutuante*	O padrão é 200.
n_iter	*Número inteiro*	O número máximo de iterações para a otimização Configure como pelo menos 250 O patrão é 1000.
angle	*Valor flutuante*	O tamanho angular do nó distante, conforme medido a partir de um ponto Especifique um valor no intervalo de 0 a 1. O padrão é 0.5.
enable_random_seed	*Booleano*	Configure como true para ativar o parâmetro random_seed O padrão é false.
random_seed	*Número inteiro*	O valor inicial de número aleatório a ser usado O padrão é None.
n_iter_without_progress	*Número inteiro*	Número máximo de iterações sem progresso O padrão é 300.
min_grad_norm	*sequência*	Se a norma gradiente estiver abaixo desse limite, a otimização será interrompida. O padrão é 1.0E-7. Os valores possíveis são: • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	*Booleano*	Configure como true para executar o t-SNE com várias perplexidades diferentes O patrão é false.
output_Rename	*Booleano*	Especifique true se você desejar fornecer um nome customizado ou false para nomear a saída automaticamente O padrão é false.
output_to	*sequência*	Especifique Screen ou Output. O padrão é Screen.
full_filename	*sequência*	Especifique o nome do arquivo de saída.
output_file_type	*sequência*	Formato do arquivo de saída.. Especifique HTML ou Output object. O padrão é HTML.

Propriedades de webnode

O nó Web ilustra a intensidade do relacionamento entre os valores de dois ou mais campos simbólicos (categóricos). O gráfico utiliza linhas de várias larguras para indicar a intensidade da conexão. Você pode usar um nó da Web, por exemplo, para explorar a relação entre a compra de um conjunto de itens em um site de e-commerce.

Exemplo

```
node = stream.create("web", "My node")
# "Plot" tab
node.setPropertyValue("use_directed_web", True)
node.setPropertyValue("to_field", "Drug")
node.setPropertyValue("fields", ["BP", "Cholesterol", "Sex", "Drug"])
node.setPropertyValue("from_fields", ["BP", "Cholesterol", "Sex"])
node.setPropertyValue("true_flags_only", False)
node.setPropertyValue("line_values", "Absolute")
node.setPropertyValue("strong_links_heavier", True)
# "Options" tab
node.setPropertyValue("max_num_links", 300)
node.setPropertyValue("links_above", 10)
node.setPropertyValue("num_links", "ShowAll")
node.setPropertyValue("discard_links_min", True)
node.setPropertyValue("links_min_records", 5)
node.setPropertyValue("discard_links_max", True)
node.setPropertyValue("weak_below", 10)
node.setPropertyValue("strong_above", 19)
node.setPropertyValue("link_size_continuous", True)
node.setPropertyValue("web_display", "Circular")
```

Tabela 109. Propriedades de webnode

propriedades webnode	Tipo de dados	Descrição da propriedade
use_directed_web	*sinalização*	
fields	*lista*	
to_field	*campo*	
from_fields	*lista*	
true_flags_only	*sinalização*	
line_values	Absolute OverallPct PctLarger PctSmaller	
strong_links_heavier	*sinalização*	
num_links	ShowMaximum ShowLinksAbove ShowAll	

Tabela 109. Propriedades de webnode (continuação)

propriedades webnode	Tipo de dados	Descrição da propriedade
max_num_links	*number*	
links_above	*number*	
discard_links_min	*sinalização*	
links_min_records	*number*	
discard_links_max	*sinalização*	
links_max_records	*number*	
weak_below	*number*	
strong_above	*number*	
link_size_continuous	*sinalização*	
web_display	Circular Network Directed Grid	
graph_background	*Cor*	As cores do gráfico padrão são descritas no início desta seção.
symbol_size	*number*	Especifica um tamanho do símbolo.

Capítulo 13. Propriedades do Nó de Modelagem

Propriedades comuns do nó de modelagem

As propriedades a seguir são comuns a alguns ou todos os nós de modelagem. Todas as exceções serão observadas na documentação de nós de modelagem individuais conforme apropriado.

Tabela 110. Propriedades comuns do nó de modelagem

Propriedade	Valores	Descrição da propriedade
`custom_fields`	*sinalização*	Se true, permite especificar campos de destino, de entrada e outros campos para o nó atual. Se false, as configurações atuais de um nó Tipo de envio de dados serão utilizadas.

`target`
 	
ou
 	
`targets`
	campo
	
	ou
	
	[*field1 ... fieldN*]
		Especifica um único campo de destino ou diversos campos de destino dependendo do tipo de modelo.

`inputs`	[*field1 ... fieldN*]	Campos de entrada ou de preditores usados pelo modelo.
`partition`	*campo*	
`use_partitioned_data`	*sinalização*	Se um campo de partição for definido, essa opção assegurará que apenas os dados da partição de treinamento sejam utilizados para construir o modelo.
`use_split_data`	*sinalização*	
`splits`	*[field1 ... fieldN]*	Especifica o campo ou campos a serem utilizados para modelagem de divisão. Efetivo apenas se `use_split_data` for configurado como `True`.
`use_frequency`	*sinalização*	Os campos de peso e de frequência são usados por modelos específicos, conforme observado para cada tipo de modelo.
`frequency_field`	*campo*	
`use_weight`	*sinalização*	
`weight_field`	*campo*	
`use_model_name`	*sinalização*	
`model_name`	*sequência*	Nome customizado para o novo modelo.

Tabela 110. Propriedades comuns do nó de modelagem (continuação)		
Propriedade	**Valores**	**Descrição da propriedade**
mode	Simple Expert	

Propriedades anomalydetectionnode

O nó de Detecção de Anomalias identifica casos incomuns, ou valores discrepantes, que não estão em conformidade com os padrões de dados "normais". Com este nó, é possível identificar outliers mesmo que eles não se encaixem em nenhum padrão anteriormente conhecido e mesmo que você não esteja exatamente certo do que você está procurando.

Exemplo

```
node = stream.create("anomalydetection", "My node")
node.setPropertyValue("anomaly_method", "PerRecords")
node.setPropertyValue("percent_records", 95)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("peer_group_num_auto", True)
node.setPropertyValue("min_num_peer_groups", 3)
node.setPropertyValue("max_num_peer_groups", 10)
```

Tabela 111. Propriedades anomalydetectionnode		
Propriedades anomalydetectionnode	**Valores**	**Descrição da propriedade**
inputs	*[field1 ... fieldN]*	Os modelos de Detecção de Anomalias selecionam registros com base nos campos de entrada especificados. Eles não usam um campo de destino. Os campos de peso e de frequência também não são utilizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
mode	Expert Simple	
anomaly_method	IndexLevel PerRecords NumRecords	Especifica o método utilizado para determinar o valor de corte para sinalizar registros como anômalos.
index_level	*number*	Especifica o valor mínimo de corte para sinalizar anomalias.
percent_records	*number*	Configura o limite para sinalizar registros com base na porcentagem de registros nos dados de treinamento.

Tabela 111. Propriedades anomalydetectionnode (continuação)		
Propriedades anomalydetectionnode	**Valores**	**Descrição da propriedade**
num_records	*number*	Configura o limite para sinalizar registros com base no número de registros nos dados de treinamento.
num_fields	*Número inteiro*	O número de campos para relatar cada registro anômalo.
impute_missing_values	*sinalização*	
adjustment_coeff	*number*	Valor utilizado para balancear o peso relativo fornecido para campos contínuos e categóricos no cálculo da distância.
peer_group_num_auto	*sinalização*	Calcula automaticamente o número de grupos de peers.
min_num_peer_groups	*Número inteiro*	Especifica o número mínimo de grupos de pares usados quando peer_group_num_auto é configurado como True.
max_num_per_groups	*Número inteiro*	Especifica o número máximo de grupos de peers.
num_peer_groups	*Número inteiro*	Especifica o número de grupos de pares usados quando peer_group_num_auto é configurado como False.
noise_level	*number*	Determina como os valores discrepantes são tratados durante o armazenamento em cluster. Especifique um valor entre 0 e 0,5.
noise_ratio	*number*	Especifica a parte da memória alocada para o componente que deve ser utilizada para o armazenamento em buffer de ruído. Especifique um valor entre 0 e 0,5.

Propriedades de apriorinode

O nó A Priori extrai um conjunto de regras dos dados, removendo as regras com o conteúdo de informações mais alto. A Priori oferece cinco métodos diferentes de seleção de regras e usa um esquema de indexação sofisticado para processar conjuntos de dados grandes com eficiência. Para grandes problemas, o A Priori geralmente é mais rápido para treinar; ele não tem um limite arbitrário no número de regras que podem ser retidas e pode manipular regras com até 32 condições prévias. O A Priori requer que os campos de entrada e saída sejam todos categóricos, mas entrega melhor desempenho por ser otimizado para esse tipo de dado.

Exemplo

```
node = stream.create("apriori", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("partition", "Test")
# For non-transactional
node.setPropertyValue("use_transactional_data", False)
node.setPropertyValue("consequents", ["Age"])
node.setPropertyValue("antecedents", ["BP", "Cholesterol", "Drug"])
# For transactional
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("content_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Apriori_bp_choles_drug")
node.setPropertyValue("min_supp", 7.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_antecedents", 7)
node.setPropertyValue("true_flags", False)
node.setPropertyValue("optimize", "Memory")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("evaluation", "ConfidenceRatio")
node.setPropertyValue("lower_bound", 7)
```

Tabela 112. Propriedades de apriorinode

Propriedades apriorinode	**Valores**	**Descrição da propriedade**
consequents	*campo*	Os modelos a priori utilizam Subsequentes e Antecedentes ao invés dos campos de destino e de entrada padrão. Os campos de peso e frequência não são usados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
antecedents	[*field1 ... fieldN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	
true_flags	*sinalização*	
optimize	Speed Memory	
use_transactional_data	*sinalização*	Quando o valor é true, a pontuação para cada ID de transação é independente de outros IDs de transação. Quando os dados a serem pontuados são muito grandes para obter um desempenho aceitável, é recomendável separar os dados.
contiguous	*sinalização*	

Tabela 112. Propriedades de apriorinode (continuação)		
Propriedades apriorinode	**Valores**	**Descrição da propriedade**
id_field	*sequência*	
content_field	*sequência*	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*number*	
optimize	Speed Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.

Propriedades de associationrulesnode

O nó de Regras de Associação é semelhante ao nó A Priori; no entanto, ao contrário do A Priori, o nó de Regras de Associação pode processar dados de lista. Além disso, o nó de Regras de Associação pode ser usado com IBM SPSS Analytic Server para processar Big Data e aproveitar a vantagem do processamento paralelo mais rápido.

Tabela 113. Propriedades de associationrulesnode		
propriedades associationrulesnode	**Tipo de dados**	**Descrição da propriedade**
predictions	*campo*	Os campos nessa lista podem aparecer apenas como um preditor de uma regra
conditions	*[field1...fieldN]*	Os campos nesta lista podem aparecer apenas como uma condição de uma regra
max_rule_conditions	*Número inteiro*	O número máximo de condições que podem ser incluídas em uma única regra. Mínimo 1, máximo 9.
max_rule_predictions	*Número inteiro*	O número máximo de predições que podem ser incluídas em uma única regra. Mínimo 1, máximo 5.
max_num_rules	*Número inteiro*	O número máximo de regras que podem ser consideradas parte da construção de regra. Mínimo 1, máximo 10.000.

Tabela 113. Propriedades de associationrulesnode (continuação)		
propriedades associationrulesnode	**Tipo de dados**	**Descrição da propriedade**
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Deployability	O critério de regra que determina o valor pelo qual as "N" principais regras no modelo são escolhidas.
true_flags	*Booleano*	Configurar como *Y* determina que somente os valores reais para os campos de sinalização são considerados durante a construção de regras.
rule_criterion	*Booleano*	Configurar como *Y* determina se os valores de critérios de regra são utilizados para exclusão de regras durante a construção de modelo.
min_confidence	*number*	0,1 a 100 - o valor de porcentagem para o nível de confiança mínimo necessário para uma regra produzida pelo modelo. Se o modelo produzir uma regra com um nível de confiança menor que o valor especificado aqui, a regra será descartada.
min_rule_support	*number*	0,1 a 100 - o valor de porcentagem para o suporte de regra mínimo necessário para uma regra produzida pelo modelo. Se o modelo produzir uma regra com um nível de suporte de regra menor que o valor especificado, a regra será descartada.
min_condition_support	*number*	0,1 a 100 – o valor de porcentagem para o suporte de condição mínimo necessário para uma regra produzida pelo modelo. Se o modelo produzir uma regra com um nível de suporte de condição menor que o valor especificado, a regra será descartada.
min_lift	*Número inteiro*	1 a 10 – representa a elevação mínima necessária para uma regra produzida pelo modelo. Se o modelo produzir uma regra com um nível de elevação menor que o valor especificado, a regra será descartada.

Tabela 113. Propriedades de associationrulesnode (continuação)

propriedades associationrulesnode	**Tipo de dados**	**Descrição da propriedade**
exclude_rules	*Booleano*	Utilizado para selecionar uma lista de campos relacionados a partir da qual você não deseja que o modelo crie regras. Exemplo: configure :gsarsnode.exclude_rules = [[[field1,field2, field3]], [[field4, field5]]-em que cada lista de campos separados por [] é uma linha na tabela.
num_bins	*Número inteiro*	Configura o número de categorias automáticas para os quais os campos contínuos são categorizados. Mínimo 2, máximo 10.
max_list_length	*Número inteiro*	Aplica-se a todos os campos da lista para os quais o comprimento máximo não é conhecido. Os elementos na lista até o número especificado aqui são incluídos na construção de modelo e quaisquer elementos adicionais são descartados. Mínimo 1, máximo 100.
output_confidence	*Booleano*	
output_rule_support	*Booleano*	
output_lift	*Booleano*	
output_condition_support	*Booleano*	
output_deployability	*Booleano*	
rules_to_display	upto all	O número máximo de regras para exibir nas tabelas de saída.
display_upto	*Número inteiro*	Se upto for configurado em rules_to_display, configure o número de regras a serem exibidas nas tabelas de saída. Mínimo 1.
field_transformations	*Booleano*	
records_summary	*Booleano*	
rule_statistics	*Booleano*	
most_frequent_values	*Booleano*	
most_frequent_fields	*Booleano*	
word_cloud	*Booleano*	

Tabela 113. Propriedades de associationrulesnode (continuação)		
propriedades associationrulesnode	**Tipo de dados**	**Descrição da propriedade**
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Deployability	
word_cloud_display	*Número inteiro*	Minimum 1, maximum 20
max_predictions	*Número inteiro*	O número máximo de regras que podem ser aplicadas a cada entrada na escoragem.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Seleciona a medida usada para determinar a força das regras.
allow_repeats	*Booleano*	Determina se regras com a mesma predição são incluídas na escoragem.
check_input	NoPredictions Predictions NoCheck	

Propriedades de autoclassifiernode

O nó Previsor Categórico Automático cria e compara inúmeros modelos diferentes para resultados binários (sim ou não, perda de clientes ou não, e assim por diante), permitindo escolher a melhor abordagem para uma determinada análise. Vários algoritmos de modelagem são suportados, possibilitando a seleção dos métodos que você deseja usar, as opções específicas para cada e os critérios para comparar os resultados. O nó gera um conjunto de modelos com base nas opções especificadas e classifica os melhores candidatos de acordo com os critérios que você especificar.

Exemplo

```
node = stream.create("autoclassifier", "My node")
node.setPropertyValue("ranking_measure", "Accuracy")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_accuracy_limit", True)
node.setPropertyValue("accuracy_limit", 0.9)
```

```
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("use_costs", True)
node.setPropertyValue("svm", False)
```

Tabela 114. Propriedades de autoclassifiernode

Propriedades autoclassifiernode	**Valores**	**Descrição da propriedade**
target	*campo*	Para destinos de sinalizador, o nó Classificador Automático requer um único campo de destino e um ou mais campos de entrada. Os campos de peso e de frequência também podem ser especificados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
ranking_measure	Accuracy Area_under_curve Profit Lift Num_variables	
ranking_dataset	Training Test	
number_of_models	*Número inteiro*	Número de modelos a serem incluídos no nugget do modelo. Especifique um número inteiro entre 1 e 100.
calculate_variable_importance	*sinalização*	
enable_accuracy_limit	*sinalização*	
accuracy_limit	*Número inteiro*	Número inteiro entre 0 e 100.
enable_ area_under_curve _limit	*sinalização*	
area_under_curve_limit	*number*	Número real entre 0,0 e 1,0.
enable_profit_limit	*sinalização*	
profit_limit	*number*	Integer maior que 0.
enable_lift_limit	*sinalização*	
lift_limit	*number*	Número real maior que 1,0.
enable_number_of_variables_lim it	*sinalização*	
number_of_variables_limit	*number*	Integer maior que 0.
use_fixed_cost	*sinalização*	

Tabela 114. Propriedades de autoclassifiernode (continuação)		
Propriedades autoclassifiernode	**Valores**	**Descrição da propriedade**
fixed_cost	*number*	Número real maior que 0,0.
variable_cost	*campo*	
use_fixed_revenue	*sinalização*	
fixed_revenue	*number*	Número real maior que 0,0.
variable_revenue	*campo*	
use_fixed_weight	*sinalização*	
fixed_weight	*number*	Número real maior que 0,0.
variable_weight	*campo*	
lift_percentile	*number*	Número inteiro entre 0 e 100.
enable_model_build_time_limit	*sinalização*	
model_build_time_limit	*number*	Número inteiro configurado para o número de minutos para limitar o tempo gasto para construir cada modelo individual.
enable_stop_after_time_limit	*sinalização*	
stop_after_time_limit	*number*	Número real configurado para o número de horas para limitar o tempo decorrido geral para uma execução de classificador automático.
enable_stop_after_valid_model_produced	*sinalização*	
use_costs	*sinalização*	
<algorithm>	*sinalização*	Ativa ou desativa o uso de um algoritmo específico.
<algorithm>.<property>	*sequência*	Configura um valor da propriedade para um algoritmo específico. Consulte o tópico "Configurando Propriedades de Algoritmo" na página 228 para obter informações adicionais.

Configurando Propriedades de Algoritmo

Para os nós Classificador Automático, Numeração Automática e Cluster Automático, as propriedades para algoritmos específicos utilizados pelo nó podem ser configuradas utilizando o formato geral:

```
autonode.setKeyedPropertyValue(<algorithm>, <property>, <value>)
```

Por exemplo:

```
node.setKeyedPropertyValue("neuralnetwork", "method", "MultilayerPerceptron")
```

Os nomes de algoritmo para o nó de Previsor categórico automático são `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` e `knn`.

Os nomes de algoritmo para o nó de Previsor contínuo automático são `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` e knn.

Os nomes de algoritmos para o nó de Clusterização automática são `twostep`, `k-means` e kohonen.

Os nomes da propriedade são padrão, conforme documentado para cada nó de algoritmo.

As propriedades de algoritmo que contiverem pontos ou outra escoragem devem ser agrupadas entre aspas simples, por exemplo:

```
node.setKeyedPropertyValue("logreg", "tolerance", "1.0E-5")
```

Diversos valores também podem ser designados para a propriedade, por exemplo:

```
node.setKeyedPropertyValue("decisionlist", "search_direction", ["Up",
"Down"])
```

Para ativar ou desativar o uso de um algoritmo específico:

```
node.setPropertyValue("chaid", True)
```

Nota: Nos casos em que determinadas opções de algoritmo não estiverem disponíveis no nó Classificador Automático, ou quando apenas um único valor puder ser especificado ao invés de um intervalo de valores, os mesmos limites se aplicam ao script como quando acessar o nó de maneira padrão.

Propriedades de autoclusternode

O nó Cluster Automático estima e compara modelos de armazenamento em cluster, que identificam grupos de registros com características semelhantes. O nó trabalha da mesma maneira que outros nós de modelagem automatizados, permitindo experimentar várias combinações de opções em uma única passagem de modelagem. Os modelos podem ser comparados usando medidas básicas com as quais tentar filtrar e classificar a utilidade dos modelos de cluster, além de fornecer uma medida com base na importância de campos particulares.

Exemplo

```
node = stream.create("autocluster", "My node")
node.setPropertyValue("ranking_measure", "Silhouette")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_silhouette_limit", True)
node.setPropertyValue("silhouette_limit", 5)
```

Tabela 115. Propriedades de autoclusternode

Propriedades autoclusternode	**Valores**	**Descrição da propriedade**
`evaluation`	*campo*	**Nota:** Somente nó Cluster Automático Identifica o campo para o qual um valor de importância será calculado. Como alternativa, pode ser utilizado para identificar quão bem o cluster diferencia o valor deste campo e, portanto, quão bem o modelo irá prever este campo.

Tabela 115. Propriedades de autoclusternode (continuação)

Propriedades autoclusternode	**Valores**	**Descrição da propriedade**
ranking_measure	Silhouette Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importance	
ranking_dataset	Training Test	
summary_limit	*Número inteiro*	Número de modelos para lista no relatório. Especifique um número inteiro entre 1 e 100.
enable_silhouette_limit	*sinalização*	
silhouette_limit	*Número inteiro*	Número inteiro entre 0 e 100.
enable_number_less_limit	*sinalização*	
number_less_limit	*number*	Número real entre 0,0 e 1,0.
enable_number_greater_limit	*sinalização*	
number_greater_limit	*number*	Integer maior que 0.
enable_smallest_cluster_limit	*sinalização*	
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*Número inteiro*	Integer maior que 0.
enable_largest_cluster_limit	*sinalização*	
largest_cluster_units	Percentage Counts	
largest_cluster_limit_percentage	*number*	

Tabela 115. Propriedades de autoclusternode (continuação)		
Propriedades autoclusternode	**Valores**	**Descrição da propriedade**
largest_cluster_limit_count	*Número inteiro*	
enable_smallest_largest_limit	*sinalização*	
smallest_largest_limit	*number*	
enable_importance_limit	*sinalização*	
importance_limit_condition	Greater_than Less_than	
importance_limit_greater_than	*number*	Número inteiro entre 0 e 100.
importance_limit_less_than	*number*	Número inteiro entre 0 e 100.
<algorithm>	*sinalização*	Ativa ou desativa o uso de um algoritmo específico.
<algorithm>.<property>	*sequência*	Configura um valor da propriedade para um algoritmo específico. Consulte o tópico "Configurando Propriedades de Algoritmo" na página 228 para obter informações adicionais.

Propriedades de autonumericnode

O nó Previsor Contínuo Automático estima e compara modelos para resultados de intervalos numéricos contínuos usando vários métodos diferentes. O nó trabalha da mesma maneira que o nó Previsor Categórico Automático, permitindo escolher os algoritmos para usar e experimentar com várias combinações de opções em uma única passagem de modelagem. Os algoritmos suportados incluem redes neurais, Árvore C e R, algoritmo Detector de Interação Automático Chi-quadrado, regressão linear, regressão linear generalizada e Support Vector Machines (SVM). Os modelos podem ser comparados com base em correlação, erro relativo ou número de variáveis utilizadas.

Exemplo

```
node = stream.create("autonumeric", "My node")
node.setPropertyValue("ranking_measure", "Correlation")
node.setPropertyValue("ranking_dataset", "Training")
node.setPropertyValue("enable_correlation_limit", True)
node.setPropertyValue("correlation_limit", 0.8)
node.setPropertyValue("calculate_variable_importance", True)
node.setPropertyValue("neuralnetwork", True)
node.setPropertyValue("chaid", False)
```

Tabela 116. Propriedades de autonumericnode		
Propriedades autonumericnode	**Valores**	**Descrição da propriedade**
custom_fields	*sinalização*	Se True, as configurações de campo customizado serão usadas ao invés das configurações do nó de tipo.
target	*campo*	O nó Numeração Automática requer um único campo de destino e um ou mais campos de entrada. Os campos de peso e de frequência também podem ser especificados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
inputs	*[field1 ... field2]*	
partition	*campo*	
use_frequency	*sinalização*	
frequency_field	*campo*	
use_weight	*sinalização*	
weight_field	*campo*	
use_partitioned_data	*sinalização*	Se um campo de partição for definido, apenas os dados de treinamento serão usados para construção de modelo.
ranking_measure	Correlation NumberOfFields	
ranking_dataset	Test Training	
number_of_models	*Número inteiro*	Número de modelos a serem incluídos no nugget do modelo. Especifique um número inteiro entre 1 e 100.
calculate_variable_importance	*sinalização*	
enable_correlation_limit	*sinalização*	
correlation_limit	*Número inteiro*	
enable_number_of_fields_limit	*sinalização*	
number_of_fields_limit	*Número inteiro*	
enable_relative_error_limit	*sinalização*	
relative_error_limit	*Número inteiro*	
enable_model_build_time_limit	*sinalização*	

Tabela 116. Propriedades de autonumericnode (continuação)		
Propriedades autonumericnode	**Valores**	**Descrição da propriedade**
model_build_time_limit	*Número inteiro*	
enable_stop_after_time_limit	*sinalização*	
stop_after_time_limit	*Número inteiro*	
stop_if_valid_model	*sinalização*	
<algorithm>	*sinalização*	Ativa ou desativa o uso de um algoritmo específico.
<algorithm>.<property>	*sequência*	Configura um valor da propriedade para um algoritmo específico. Consulte o tópico "Configurando Propriedades de Algoritmo" na página 228 para obter informações adicionais.

Propriedades de bayesnetnode

O nó de Rede Bayesiana permite construir um modelo de probabilidade combinando evidências observadas e registradas com conhecimento do mundo real para estabelecer a probabilidade de ocorrências. O nó focaliza as redes Tree Augmented Naïve Bayes (TAN) e Markov Blanket que são usadas principalmente para classificação.

Exemplo

```
node = stream.create("bayesnet", "My node")
node.setPropertyValue("continue_training_existing_model", True)
node.setPropertyValue("structure_type", "MarkovBlanket")
node.setPropertyValue("use_feature_selection", True)
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("independence", "Pearson")
```

Tabela 117. Propriedades de bayesnetnode		
Propriedades bayesnetnode	**Valores**	**Descrição da propriedade**
inputs	*[field1 ... fieldN]*	Os modelos de rede bayesiana utilizam um único campo de destino e um ou mais campos de entrada. Os campos contínuos são automaticamente categorizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
continue_training_existing_model	*sinalização*	

Tabela 117. Propriedades de bayesnetnode (continuação)		
Propriedades bayesnetnode	**Valores**	**Descrição da propriedade**
structure_type	TAN MarkovBlanket	Selecione a estrutura a ser utilizada ao construir a rede bayesiana.
use_feature_selection	*sinalização*	
parameter_learning_method	Likelihood Bayes	Especifica o método utilizado para estimar as tabelas de probabilidade condicional entre os nós nos quais os valores dos pais são conhecidos.
mode	Expert Simple	
missing_values	*sinalização*	
all_probabilities	*sinalização*	
independence	Likelihood Pearson	Especifica o método usado para determinar se as observações emparelhadas nas duas variáveis são independentes entre si.
significance_level	*number*	Especifica o valor de corte para determinar a independência.
maximal_conditioning_set	*number*	Configura o número máximo de variáveis de condicionamento a serem utilizadas para teste de independência.
inputs_always_selected	*[field1 ... fieldN]*	Especifica quais campos do conjunto de dados devem sempre ser utilizados durante a construção da rede bayesiana. **Nota:** O campo de destino é sempre selecionado.
maximum_number_inputs	*number*	Especifica o número máximo de campos de entrada a serem utilizados na construção da rede bayesiana.
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades de buildr

O nó Construção R permite inserir script R customizado para executar construção e escoragem de modelo implementado no IBM SPSS Modeler.

Exemplo

```
node = stream.create("buildr", "My node")
node.setPropertyValue("score_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-
c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabela 118. Propriedades de buildr

Propriedades buildr	Valores	Descrição da propriedade
build_syntax	*sequência*	Sintaxe do script R para construção de modelo.
score_syntax	*sequência*	Sintaxe do script R para escoragem de modelo.
convert_flags	StringsAndDoubles LogicalValues	Opção para converter os campos de sinalização.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	POSIXct POSIXlt	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.
convert_missing	*sinalização*	Opção para converter valores ausentes em valor NA do R.
output_html	*sinalização*	Opção para exibir gráficos em uma guia no nugget do modelo R.
output_text	*sinalização*	Opção para gravar a saída de texto do console R em uma guia no nugget do modelo R.

Propriedades de c50node

O nó C5.0 constrói uma árvore de decisão ou um conjunto de regras. O modelo funciona dividindo a amostra com base no campo que fornece o ganho máximo de informações em cada nível. O campo de destino deve ser categórico. São permitidas várias divisões em mais de dois subgrupos.

Exemplo

```
node = stream.create("c50", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "C5_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("output_type", "DecisionTree")
node.setPropertyValue("use_xval", True)
node.setPropertyValue("xval_num_folds", 3)
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("favor", "Generality")
node.setPropertyValue("min_child_records", 3)
# "Costs" tab
node.setPropertyValue("use_costs", True)
node.setPropertyValue("costs", [["drugA", "drugX", 2]])
```

Tabela 119. Propriedades de c50node

Propriedades c50node	Valores	Descrição da propriedade
target	*campo*	Os modelos C50 utilizam um único campo de destino e um ou mais campos de entrada. Um campo de ponderação também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
output_type	DecisionTree RuleSet	
group_symbolics	*sinalização*	
use_boost	*sinalização*	
boost_num_trials	*number*	
use_xval	*sinalização*	
xval_num_folds	*number*	
mode	Simple Expert	
favor	Accuracy Generality	Favorece a precisão ou a generalidade.
expected_noise	*number*	
min_child_records	*number*	
pruning_severity	*number*	
use_costs	*sinalização*	
costs	*estruturado*	Esta é uma propriedade estruturada.
use_winnowing	*sinalização*	
use_global_pruning	*sinalização*	Ativado (True) por padrão.

Tabela 119. Propriedades de c50node (continuação)		
Propriedades c50node	**Valores**	**Descrição da propriedade**
`calculate_variable_importance`	*sinalização*	
`calculate_raw_propensities`	*sinalização*	
`calculate_adjusted_propensities`	*sinalização*	

`adjusted_propensity_partition`	`Test`
	
	`Validation`	

Propriedades carmanode

O modelo CARMA extrai um conjunto de regras dos dados sem requerer a especificação de campos de entrada ou saída. Em contraste com Apriori o nó CARMA oferece configurações de construção para suporte de regra (suporte para tanto antecedentes e consequentes) em vez de apenas suporte antecedente. Isso significa que as regras geradas podem ser usadas para uma variedade maior de aplicativos—por exemplo, para localizar uma lista de produtos ou serviços (antecedentes) cujo consequente é o item que você deseja promover nesta temporada de férias.

Exemplo

```
node = stream.create("carma", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("use_transactional_data", True)
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Drug"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "age_bp_drug")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 10.0)
node.setPropertyValue("min_conf", 30.0)
node.setPropertyValue("max_size", 5)
# Expert Options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 300)
node.setPropertyValue("vary_support", True)
node.setPropertyValue("estimated_transactions", 30)
node.setPropertyValue("rules_without_antecedents", True)
```

Tabela 120. Propriedades carmanode		
Propriedades carmanode	**Valores**	**Descrição da propriedade**
inputs	*[field1 ... fieldn]*	Os modelos de CARMA utilizam uma lista de campos de entrada, mas não de destino. Os campos de peso e frequência não são usados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
id_field	*campo*	Campo utilizado como o campo de ID para construção de modelo.
contiguous	*sinalização*	Utilizado para especificar se os IDs no campo ID são contíguos.
use_transactional_data	*sinalização*	
content_field	*campo*	
min_supp	*number(percent)*	Relaciona ao suporte da regra ao invés de ao suporte da antecedent. O padrão é 20%.
min_conf	*number(percent)*	O padrão é 20%.
max_size	*number*	O padrão é 10.
mode	Simple Expert	O padrão é Simple.
exclude_multiple	*sinalização*	Exclui regras com diversos subsequentes. O padrão é False.
use_pruning	*sinalização*	O padrão é False.
pruning_value	*number*	O padrão é 500.
vary_support	*sinalização*	
estimated_transactions	*Número inteiro*	
rules_without_antecedents	*sinalização*	

Propriedades de cartnode

O nó Classificação e Regressão (C e R) gera uma árvore de decisão que permite prever ou classificar futuras observações. O método usa particionamento recursivo para dividir os registros de treinamento em segmentos, minimizando a impureza em cada etapa, em que um nó na árvore será considerado "puro" se 100% dos casos no nó caírem em uma categoria específica do campo de destino. Campos de destino e entrada podem ser intervalos numéricos ou categóricos (nominal, ordinal ou sinalizadores); todos os splits são binários (apenas dois subgrupos).

Exemplo

```
node = stream.createAt("cart", "My node", 200, 100)
# "Fields" tab
node.setPropertyValue("custom_fields", True)
```

```
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "BP", "Cholesterol"])
# "Build Options" tab, "Objective" panel
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", """Grow Node Index 0 Children 1 2
Grow Node Index 2 Children 3 4""")
# "Build Options" tab, "Basics" panel
node.setPropertyValue("prune_tree", False)
node.setPropertyValue("use_std_err_rule", True)
node.setPropertyValue("std_err_multiplier", 3.0)
node.setPropertyValue("max_surrogates", 7)
# "Build Options" tab, "Stopping Rules" panel
node.setPropertyValue("use_percentage", True)
node.setPropertyValue("min_parent_records_pc", 5)
node.setPropertyValue("min_child_records_pc", 3)
# "Build Options" tab, "Advanced" panel
node.setPropertyValue("min_impurity", 0.0003)
node.setPropertyValue("impurity_measure", "Twoing")
# "Model Options" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Cart_Drug")
```

Tabela 121. Propriedades de cartnode

Propriedades cartnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos de árvore C&R requerem um único campo de destino e um ou mais campos de entrada. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
continue_training_existing_model	*sinalização*	
objective	Standard Boosting Bagging psm	O psm é usado para conjuntos de dados muito grandes e requer uma conexão do servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*sinalização*	

Tabela 121. Propriedades de cartnode (continuação)

Propriedades cartnode	Valores	Descrição da propriedade
tree_directives	*sequência*	Especificar diretivas para o crescimento da árvore. As diretivas podem ser agrupadas entre aspas triplas para evitar escape de novas linhas ou de aspas. Observe que as diretivas podem ser altamente sensíveis a pequenas mudanças nos dados ou nas opções de modelagem e podem não generalizar a outros conjuntos de dados.
use_max_depth	Default Custom	
max_depth	*Número inteiro*	Profundidade máxima da árvore, de 0 a 1000. Usado apenas se use_max_depth = Custom.
prune_tree	*sinalização*	Poda a árvore para evitar super ajuste.
use_std_err	*sinalização*	Utiliza a diferença máxima em risco (nos Erros Padrão).
std_err_multiplier	*number*	Diferença máxima.
max_surrogates	*number*	Máximo de substitutos.
use_percentage	*sinalização*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*sinalização*	
costs	*estruturado*	Propriedade estruturada.
priors	Data Equal Custom	
custom_priors	*estruturado*	Propriedade estruturada.
adjust_priors	*sinalização*	
trails	*number*	Número de modelos de componente para boosting ou bagging.

Tabela 121. Propriedades de cartnode (continuação)		
Propriedades cartnode	**Valores**	**Descrição da propriedade**
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regra de combinação padrão para variáveis resposta categórica.
range_ensemble_method	Mean Median	Regra de combinação padrão para variáveis resposta contínua.
large_boost	*sinalização*	Aplica boosting em conjuntos de dados muito grandes.
min_impurity	*number*	
impurity_measure	Gini Twoing Ordered	
train_pct	*number*	Conjunto de prevenção ao super ajuste
set_random_seed	*sinalização*	Replica a opção de resultados.
seed	*number*	
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

propriedades chaidnode

O nó CHAID gera árvores de decisão usando estatísticas qui-quadrado para identificar as divisões ideais. Ao contrário dos nós Árvore C e R e QUEST, o CHAID pode gerar árvores não binárias, o que significa que algumas divisões têm mais de duas ramificações. Os campos de destino e de entrada podem ser um intervalo numérico (contínuo) ou categóricos. Exaustivo CHAID é uma modificação de CHAID que faz um trabalho mais minucioso de examinar todas as divisões possíveis mas demora mais tempo para computar.

Exemplo

```
filenode = stream.createAt("variablefile", "My node", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("chaid", "My node", 200, 100)
```

```
stream.link(filenode, node)

node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "CHAID")
node.setPropertyValue("method", "Chaid")
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("tree_directives", "Test")
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("merge_alpha", 0.04)
node.setPropertyValue("chi_square", "Pearson")
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("epsilon", 0.003)
node.setPropertyValue("max_iterations", 75)
node.setPropertyValue("split_merged_categories", True)
node.setPropertyValue("bonferroni_adjustment", True)
```

Tabela 122. propriedades chaidnode

Propriedades chaidnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos CHAID requerem um único campo de destino e um ou mais campos de entrada. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
continue_training_existing_model	*sinalização*	
objective	Standard Boosting Bagging psm	O psm é usado para conjuntos de dados muito grandes e requer uma conexão do servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*sinalização*	
tree_directives	*sequência*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	

Tabela 122. propriedades chaidnode (continuação)		
Propriedades chaidnode	**Valores**	**Descrição da propriedade**
max_depth	*Número inteiro*	Profundidade máxima da árvore, de 0 a 1000. Usado apenas se use_max_depth = Custom.
use_percentage	*sinalização*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*sinalização*	
costs	*estruturado*	Propriedade estruturada.
trails	*number*	Número de modelos de componente para boosting ou bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regra de combinação padrão para variáveis resposta categórica.
range_ensemble_method	Mean Median	Regra de combinação padrão para variáveis resposta contínua.
large_boost	*sinalização*	Aplica boosting em conjuntos de dados muito grandes.
split_alpha	*number*	Nível de significância para divisão.
merge_alpha	*number*	Nível de significância para mesclagem.
bonferroni_adjustment	*sinalização*	Ajusta valores de significância usando o método de Bonferroni.
split_merged_categories	*sinalização*	Permite redivisão de categorias mescladas.
chi_square	Pearson LR	Método utilizado para calcular a estatística chi-quadrada: Razão de Verossimilhança ou Pearson
epsilon	*number*	Mudança mínima nas frequências de célula esperadas.
max_iterations	*number*	Iterações máximas para convergência.
set_random_seed	*Número inteiro*	
seed	*number*	
calculate_variable_importance	*sinalização*	

Tabela 122. propriedades chaidnode (continuação)		
Propriedades chaidnode	**Valores**	**Descrição da propriedade**
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	
maximum_number_of_models	*Número inteiro*	

propriedades coxregnode

O nó de Regressão de Cox permite construir um modelo de sobrevivência para dados de sobrevivência na presença de registros censurados. O modelo produz uma função de sobrevivência que prevê a probabilidade de que o evento de interesse tenha ocorrido em um determinado momento (*t*) para valores dados das variáveis de entrada.

Exemplo

```
node = stream.create("coxreg", "My node")
node.setPropertyValue("survival_time", "tenure")
node.setPropertyValue("method", "BackwardsStepwise")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("removal_criterion", "Conditional")
node.setPropertyValue("survival", True)
```

Tabela 123. propriedades coxregnode		
Propriedades coxregnode	**Valores**	**Descrição da propriedade**
survival_time	*campo*	Os modelos de regressão de Cox requerem um único campo contendo os tempos de sobrevivência.
target	*campo*	Os modelos de regressão de Cox requerem um único campo de destino e um ou mais campos de entrada. Consulte o tópico “Propriedades comuns do nó de modelagem” na página 219 para obter informações adicionais.
method	Enter Stepwise BackwardsStepwise	
groups	*campo*	

Tabela 123. propriedades coxregnode (continuação)

Propriedades coxregnode	Valores	Descrição da propriedade
model_type	MainEffects Custom	
custom_terms	["*BP*Sex*" "*BP*Age*"]	
mode	Expert Simple	
max_iterations	*number*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	

Tabela 123. propriedades coxregnode (continuação)		
Propriedades coxregnode	**Valores**	**Descrição da propriedade**
removal_criterion	LR Wald Conditional	
probability_entry	*number*	
probability_removal	*number*	
output_display	EachStep LastStep	
ci_enable	*sinalização*	
ci_value	90 95 99	
correlation	*sinalização*	
display_baseline	*sinalização*	
survival	*sinalização*	
hazard	*sinalização*	
log_minus_log	*sinalização*	
one_minus_survival	*sinalização*	
separate_line	*campo*	
value	*number* ou *string*	Se nenhum valor for especificado para um campo, a opção padrão "Mean" será utilizada para esse campo.

Propriedades decisionlistnode

O nó de Lista de Decisão identifica subgrupos, ou segmentos, que mostram uma probabilidade maior ou menor de um determinado resultado binário relativo à população geral. Por exemplo, você pode procurar clientes que provavelmente não irão migrar para o concorrente ou têm maior probabilidade de responder favoravelmente a uma campanha. É possível incorporar seu conhecimento de negócios no modelo, incluindo seus próprios segmentos customizados e visualizando modelos alternativos lado a lado para comparar os resultados. Os modelos de Lista de Decisão consistem em uma lista de regras na qual cada regra possui uma condição e um resultado. As regras são aplicadas na ordem, e a primeira regra que corresponder determina o resultado.

Exemplo

```
node = stream.create("decisionlist", "My node")
node.setPropertyValue("search_direction", "Down")
node.setPropertyValue("target_value", 1)
node.setPropertyValue("max_rules", 4)
node.setPropertyValue("min_group_size_pct", 15)
```

Tabela 124. Propriedades decisionlistnode

Propriedades decisionlistnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos de Lista de Decisão utilizam um único campo de destino e um ou mais campos de entrada. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
model_output_type	Model InteractiveBuilder	
search_direction	Up Down	Relaciona para localizar segmentos, em que UP é o equivalente de Alta Probabilidade e Down é equivalente a Baixa Probabilidade.
target_value	*sequência*	Se não for especificado, será assumido o valor true para sinalizadores.
max_rules	*Número inteiro*	O número máximo de segmentos, exceto o restante.
min_group_size	*Número inteiro*	Tamanho mínimo do segmento.
min_group_size_pct	*number*	Tamanho mínimo do segmento como uma porcentagem.
confidence_level	*number*	O limite mínimo que um campo de entrada deve melhorar a probabilidade de resposta (fornecer a elevação), para que valha a pena inclui-lo em uma definição do segmento.
max_segments_per_rule	*Número inteiro*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*Número inteiro*	Procura largura para listas.

Tabela 124. Propriedades decisionlistnode (continuação)		
Propriedades decisionlistnode	**Valores**	**Descrição da propriedade**
max_rules_per_cycle	*Número inteiro*	Procura largura para regras de segmento.
segment_growth	*number*	
include_missing	*sinalização*	
final_results_only	*sinalização*	
reuse_fields	*sinalização*	Permite que atributos (campos de entrada que aparecem nas regras) sejam reutilizados.
max_alternatives	*Número inteiro*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades discriminantnode

A análise discriminante faz pressupostos mais rigorosos do que a regressão logística mas pode ser uma alternativa valiosa ou complementar a uma análise de regressão logística quando essas suposições forem atendidas.

Exemplo

```
node = stream.create("discriminant", "My node")
node.setPropertyValue("target", "custcat")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "Stepwise")
```

Tabela 125. Propriedades discriminantnode		
Propriedades discriminantnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos discriminantes requerem um único campo de destino e um ou mais campos de entrada. Os campos de peso e frequência não são usados. Consulte o tópico “Propriedades comuns do nó de modelagem” na página 219 para obter informações adicionais.
method	Enter Stepwise	

Tabela 125. Propriedades discriminantnode (continuação)

Propriedades discriminantnode	**Valores**	**Descrição da propriedade**
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
means	*sinalização*	Opções de estatísticas na caixa de diálogo Saída Avançada.
univariate_anovas	*sinalização*	
box_m	*sinalização*	
within_group_covariance	*sinalização*	
within_groups_correlation	*sinalização*	
separate_groups_covariance	*sinalização*	
total_covariance	*sinalização*	
fishers	*sinalização*	
unstandardized	*sinalização*	
casewise_results	*sinalização*	Opções de classificação na caixa de diálogo Saída Avançada.
limit_to_first	*number*	O valor padrão é 10.
summary_table	*sinalização*	
leave_one_classification	*sinalização*	
combined_groups	*sinalização*	
separate_groups_covariance	*sinalização*	Opção de matrizes de **Covariância de grupos separados**.
territorial_map	*sinalização*	
combined_groups	*sinalização*	Opção de gráfico **Grupos combinados**.
separate_groups	*sinalização*	Opção de gráfico **Grupos separados**.
summary_of_steps	*sinalização*	
F_pairwise	*sinalização*	

Tabela 125. Propriedades discriminantnode (continuação)		
Propriedades discriminantnode	**Valores**	**Descrição da propriedade**
stepwise_method	WilksLambda UnexplainedVariance MahalanobisDistance SmallestF RaosV	
V_to_enter	*number*	
criteria	UseValue UseProbability	
F_value_entry	*number*	O valor padrão é 3,84.
F_value_removal	*number*	O valor padrão é 2,71.
probability_entry	*number*	O valor padrão é 0,05.
probability_removal	*number*	O valor padrão é 0,10.
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

propriedades extensionmodelnode

Com o nó Modelo de Extensão, é possível executar R ou Python para scripts spark para construir e pontuar resultados.

Exemplo de Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_build", "extension_build")
node.setPropertyValue("syntax_type", "Python")

build_script = """
import json
import spss.pyspark.runtime
```

```
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTree

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]

target = "Drug"
predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

def metaMap(row,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(set([row[col]]))
        else:
            meta.append((row[col],row[col]))
        col += 1
    return meta

def metaReduce(meta1,meta2,schema):
    col = 0
    meta = []
    for (cname, ctype) in schema:
        if ctype == 'string':
            meta.append(meta1[col].union(meta2[col]))
        else:
            meta.append((min(meta1[col][0],meta2[col][0]),max(meta1[col][1],meta2[col][1])))
        col += 1
    return meta

metadata = df.rdd.map(lambda row: metaMap(row,schema)).reduce(lambda x,y:metaReduce(x,y,schema))

def setToList(v):
    if isinstance(v,set):
        return list(v)
    return v

metadata = map(lambda x: setToList(x), metadata)
print metadata

lookup = {}
for i in range(0,len(schema)):
    lookup[schema[i][0]] = i

def row2LabeledPoint(dm,lookup,target,predictors,row):
    target_index = lookup[target]
    tval = dm[target_index].index(row[target_index])
    pvals = []
    for predictor in predictors:
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            pval = dm[predictor_index].index(row[predictor_index])
        else:
            pval = row[predictor_index]
        pvals.append(pval)
    return LabeledPoint(tval,DenseVector(pvals))

# count number of target classes
predictorClassCount = len(metadata[lookup[target]])

# define function to extract categorical predictor information from datamodel
def getCategoricalFeatureInfo(dm,lookup,predictors):
    info = {}
    for i in range(0,len(predictors)):
        predictor = predictors[i]
        predictor_index = lookup[predictor]
        if isinstance(dm[predictor_index],list):
            info[i] = len(dm[predictor_index])
    return info

# convert dataframe to an RDD containing LabeledPoint
lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))

treeModel = DecisionTree.trainClassifier(
    lps,
    numClasses=predictorClassCount,
    categoricalFeaturesInfo=getCategoricalFeatureInfo(metadata, lookup, predictors),
    impurity='gini',
    maxDepth=5,
```

```
    maxBins=100)

_outputPath = cxt.createTemporaryFolder()
treeModel.save(cxt.getSparkContext(), _outputPath)
cxt.setModelContentFromPath("TreeModel", _outputPath)
cxt.setModelContentFromString("model.dm",json.dumps(metadata), mimeType="application/json")\
   .setModelContentFromString("model.structure",treeModel.toDebugString())

"""

node.setPropertyValue("python_build_syntax", build_script)
```

Exemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_build_syntax", """modelerModel <-
lm(modelerData$Na~modelerData$K,modelerData)
modelerDataModel
modelerModel
""")
```

Tabela 126. propriedades extensionmodelnode

Propriedades extensionmodelnode	**Valores**	**Descrição da propriedade**
syntax_type	*R* *Python*	Especifique qual script é executado - R ou Python (R é o padrão).
r_build_syntax	*sequência*	A sintaxe de script R para construção de modelo.
r_score_syntax	*sequência*	A sintaxe de script R para pontuação de modelo.
python_build_syntax	*sequência*	A sintaxe de script Python para construção de modelo.
python_score_syntax	*sequência*	A sintaxe de script Python para pontuação de modelo.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opção para converter os campos de sinalização.
convert_missing	*sinalização*	Opção para converter valores ausentes em valor NA do R.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.
output_html	*sinalização*	Opção para exibir gráficos em uma guia no nugget do modelo R.
output_text	*sinalização*	Opção para gravar a saída de texto do console R em uma guia no nugget do modelo R.

Propriedades factornode

O nó PCA/Fator fornece técnicas poderosas de redução de dados para reduzir a complexidade de seus dados. A análise de componentes principais (PCA) localiza combinações lineares dos campos de entrada que realizam as melhores tarefas de captura de variância no conjunto de campos inteiro, no qual os componentes são ortogonais (perpendiculares) uns aos outros. A análise fatorial tenta identificar os fatores subjacentes que explicam o padrão de correlações dentro de um conjunto observado de campos. Para ambas as abordagens, o objetivo é localizar um pequeno número de campos derivados que sumarizam efetivamente as informações no conjunto de campos original.

Exemplo

```
node = stream.create("factor", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["BP", "Na", "K"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Factor_Age")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("method", "GLS")
# Expert options
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", True)
node.setPropertyValue("matrix", "Covariance")
node.setPropertyValue("max_iterations", 30)
node.setPropertyValue("extract_factors", "ByFactors")
node.setPropertyValue("min_eigenvalue", 3.0)
node.setPropertyValue("max_factor", 7)
node.setPropertyValue("sort_values", True)
node.setPropertyValue("hide_values", True)
node.setPropertyValue("hide_below", 0.7)
# "Rotation" section
node.setPropertyValue("rotation", "DirectOblimin")
node.setPropertyValue("delta", 0.3)
node.setPropertyValue("kappa", 7.0)
```

Tabela 127. Propriedades factornode

Propriedades `factornode`	Valores	Descrição da propriedade
`inputs`	[*field1 ... fieldN*]	Os modelos de PCA/Fator utilizam uma lista de campos de entrada, mas não de destino. Os campos de peso e frequência não são usados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.

Tabela 127. Propriedades factornode (continuação)		
Propriedades factornode	**Valores**	**Descrição da propriedade**
method	PC ULS GLS ML PAF Alpha Image	
mode	Simple Expert	
max_iterations	*number*	
complete_records	*sinalização*	
matrix	Correlation Covariance	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
rotation	None Varimax DirectOblimin Equamax Quartimax Promax	
delta	*number*	Se você selecionar DirectOblimin como seu tipo de dados de rotação, será possível especificar um valor para delta. Se você não especificar um valor, o valor padrão para delta será utilizado.

Tabela 127. Propriedades factornode (continuação)		
Propriedades factornode	**Valores**	**Descrição da propriedade**
kappa	*number*	Se você selecionar Promax como seu tipo de dados de rotação, será possível especificar um valor para kappa. Se você não especificar um valor, o valor padrão para kappa será utilizado.
sort_values	*sinalização*	
hide_values	*sinalização*	
hide_below	*number*	

propriedades de featureselectionnode

O nó Seleção de Variável verifica campos de entrada para remoção com base em um conjunto de critérios (como a porcentagem de valores omissos); em seguida, classifica a importância de entradas restantes com relação a um destino especificado. Por exemplo, dado um conjunto de dados com centenas de entradas potenciais, quais têm maior probabilidade de serem úteis na modelagem de resultados do paciente?

Exemplo

```
node = stream.create("featureselection", "My node")
node.setPropertyValue("screen_single_category", True)
node.setPropertyValue("max_single_category", 95)
node.setPropertyValue("screen_missing_values", True)
node.setPropertyValue("max_missing_values", 80)
node.setPropertyValue("criteria", "Likelihood")
node.setPropertyValue("unimportant_below", 0.8)
node.setPropertyValue("important_above", 0.9)
node.setPropertyValue("important_label", "Check Me Out!")
node.setPropertyValue("selection_mode", "TopN")
node.setPropertyValue("top_n", 15)
```

Para obter um exemplo mais detalhado que cria e aplica um modelo de Seleção de Variável, consulte em

Tabela 128. propriedades de featureselectionnode		
Propriedades featureselectionnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos de Seleção de Recurso classificam preditores com relação ao destino especificado. Os campos de peso e frequência não são usados. Consulte o tópico “Propriedades comuns do nó de modelagem” na página 219 para obter informações adicionais.
screen_single_category	*sinalização*	Se True, os campos de telas que possuem muitos registros cairão na mesma categoria em relação ao número total de registros.

Tabela 128. propriedades de featureselectionnode (continuação)

Propriedades `featureselectionnode`	**Valores**	**Descrição da propriedade**
`max_single_category`	*number*	Especifica o limite usado quando `screen_single_category` é `True`.
`screen_missing_values`	*sinalização*	Se `True`, os campos de telas com muitos valores ausentes, expressos como uma porcentagem do número total de registros.
`max_missing_values`	*number*	
`screen_num_categories`	*sinalização*	Se `True`, faz triagem de campos com muitas categorias em relação ao número total de registros.
`max_num_categories`	*number*	
`screen_std_dev`	*sinalização*	Se `True`, faz triagem de campos com um desvio padrão menor ou igual ao mínimo especificado.
`min_std_dev`	*number*	
`screen_coeff_of_var`	*sinalização*	Se `True`, faz triagem de campos com coeficiente de variância menor ou igual ao mínimo especificado.
`min_coeff_of_var`	*number*	

`criteria`	`Pearson`
	
	`Likelihood`
	
	`CramersV`
	
	`Lambda`
		Ao classificar preditores categóricos com relação a uma variável resposta categórica, especifica a medida na qual o valor de importância é baseado.

`unimportant_below`	*number*	Especifica o limite de valores *p* utilizados para classificar variáveis como importantes, marginais ou insignificantes. Aceita valores de 0,0 a 1,0.
`important_above`	*number*	Aceita valores de 0,0 a 1,0.
`unimportant_label`	*sequência*	Especifica o rótulo para a classificação não importante.
`marginal_label`	*sequência*	
`important_label`	*sequência*	

`selection_mode`	`ImportanceLevel`
	
	`ImportanceValue`
	
	`TopN`	

Tabela 128. propriedades de featureselectionnode (continuação)		
Propriedades `featureselectionnode`	**Valores**	**Descrição da propriedade**
`select_important`	*sinalização*	Quando `selection_mode` é configurado como `ImportanceLevel`, especifica se selecionará campos importantes.
`select_marginal`	*sinalização*	Quando `selection_mode` é configurado como `ImportanceLevel`, especifica se selecionará campos marginais.
`select_unimportant`	*sinalização*	Quando `selection_mode` é configurado como `ImportanceLevel`, especifica se selecionará campos não importantes.
`importance_value`	*number*	Quando `selection_mode` é configurado como `ImportanceValue`, especifica o valor de corte a ser usado. Aceita valores de 0 a 100.
`top_n`	*Número inteiro*	Quando `selection_mode` é configurado como TopN, especifica o valor de corte a ser usado. Aceita valores de 0 a 1000.

Propriedades de genlinnode

O Modelo Linear Generalizado expande o modelo linear geral para que a variável dependente esteja linearmente relacionada aos fatores e às covariáveis por meio de uma função de ligação especificada. Além do mais, o modelo permite que a variável dependente tenha uma distribuição não normal. Ele cobre a funcionalidade de um grande número de modelos estatísticos, incluindo regressão linear, regressão logística, modelos de loglinear para dados de contagem e modelos de sobrevivência censurados de intervalo.

Exemplo

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tabela 129. Propriedades de genlinnode		
Propriedades genlinnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos lineares generalizados requerem um único campo de destino que deve ser um campo nominal ou de sinalização e um ou mais campos de entrada. Um campo de ponderação também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
use_weight	*sinalização*	
weight_field	*campo*	O tipo de campo é apenas contínuo.
target_represents_trials	*sinalização*	
trials_type	Variable FixedValue	
trials_field	*campo*	O tipo de campo é contínuo, de sinalização ou ordinal.
trials_number	*number*	O valor padrão é 10.
model_type	MainEffects MainAndAllTwoWayEffects	
offset_type	Variable FixedValue	
offset_field	*campo*	O tipo de campo é apenas contínuo.
offset_value	*number*	Deve ser um número real.
base_category	Last First	
include_intercept	*sinalização*	
mode	Simple Expert	

Tabela 129. Propriedades de genlinnode (continuação)

Propriedades genlinnode	Valores	Descrição da propriedade
distribution	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Gaussiana inversa. NEGBIN: Binomial negativa.
negbin_para_type	Specify Estimate	
negbin_parameter	*number*	O valor padrão é 1. Deve conter um número real não negativo.
tweedie_parameter	*number*	

Tabela 129. Propriedades de genlinnode (continuação)

Propriedades genlinnode	Valores	Descrição da propriedade
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: Log complementar-log. LOGC: complemento de log. NEGBIN: Binomial negativa. NLOGLOG: log negativo-log. CUMCAUCHIT: Cauchit cumulativo. CUMCLOGLOG: log complementar cumulativo-log. CUMLOGIT: logit cumulativo. CUMNLOGLOG: log negativo cumulativo-log. CUMPROBIT: probito cumulativo.
power	*number*	O valor deve ser um número real diferente de zero.
method	Hybrid Fisher NewtonRaphson	
max_fisher_iterations	*number*	O valor padrão é 1; somente números inteiros positivos são permitidos.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	

Tabela 129. Propriedades de genlinnode (continuação)

Propriedades genlinnode	Valores	Descrição da propriedade
scale_value	*number*	O valor padrão é 1; deve ser maior que 0.
covariance_matrix	ModelEstimator RobustEstimator	
max_iterations	*number*	O valor padrão é 100; somente números inteiros não negativos.
max_step_halving	*number*	O valor padrão é 5; somente números inteiros positivos.
check_separation	*sinalização*	
start_iteration	*number*	O valor padrão é 20; somente números inteiros positivos são permitidos.
estimates_change	*sinalização*	
estimates_change_min	*number*	O valor padrão é 1E-006; somente números positivos são permitidos.
estimates_change_type	Absolute Relative	
loglikelihood_change	*sinalização*	
loglikelihood_change_min	*number*	Apenas números positivos permitidos.
loglikelihood_change_type	Absolute Relative	
hessian_convergence	*sinalização*	
hessian_convergence_min	*number*	Apenas números positivos permitidos.
hessian_convergence_type	Absolute Relative	
case_summary	*sinalização*	
contrast_matrices	*sinalização*	
descriptive_statistics	*sinalização*	
estimable_functions	*sinalização*	
model_info	*sinalização*	
iteration_history	*sinalização*	
goodness_of_fit	*sinalização*	
print_interval	*number*	O valor padrão é 1; deve ser um número inteiro positivo.
model_summary	*sinalização*	

Tabela 129. Propriedades de genlinnode (continuação)

Propriedades genlinnode	Valores	Descrição da propriedade
lagrange_multiplier	*sinalização*	
parameter_estimates	*sinalização*	
include_exponential	*sinalização*	
covariance_estimates	*sinalização*	
correlation_estimates	*sinalização*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Profile	
tolerancelevel	*number*	O valor padrão é 0,0001.
confidence_interval	*number*	O valor padrão é 95.
loglikelihood_function	Full Kernel	
singularity_tolerance	1E-007 1E-008 1E-009 1E-010 1E-011 1E-012	
value_order	Ascending Descending DataOrder	
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	

Tabela 129. Propriedades de genlinnode (continuação)		
Propriedades genlinnode	**Valores**	**Descrição da propriedade**
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades de glmmnode

Um modelo linear generalizado misto (GLMM) estende o modelo linear para que a resposta possa ter uma distribuição não normal, esteja linearmente relacionada aos fatores e às covariáveis via uma função de ligação especificada e para que as observações possam ser correlacionadas. Os modelos lineares generalizados mistos abrangem uma ampla variedade de modelos, desde regressão linear simples até modelos multiníveis complexos para dados longitudinais não normais.

Tabela 130. Propriedades de glmmnode		
Propriedades glmmnode	**Valores**	**Descrição da propriedade**
residual_subject_spec	*estruturado*	A combinação de valores dos campos categóricos especificados que definem exclusivamente assuntos no conjunto de dados
repeated_measures	*estruturado*	Campos utilizados para identificar observações repetidas.
residual_group_spec	[*field1 ... fieldN*]	Campos que definem conjuntos independentes de parâmetros de covariância de efeitos repetidos.
residual_covariance_type	Diagonal AR1 ARMA11 COMPOUND_SYMMETRY IDENTITY TOEPLITZ UNSTRUCTURED VARIANCE_COMPONENTS	Especifica a estrutura de covariâncias para residuais.
custom_target	*sinalização*	Indica se deve usar o destino definido em nó de envio de dados (false) ou o destino customizado especificado por target_field (true).

Tabela 130. Propriedades de glmmnode (continuação)

Propriedades glmmnode	**Valores**	**Descrição da propriedade**
target_field	*campo*	Campo para usar como destino se custom_target for true.
use_trials	*sinalização*	Indica se um campo ou um valor adicional que especifica o número de avaliações deve ser utilizado quando a resposta de destino for um número de eventos que ocorrem em um conjunto de avaliações. O padrão é false.
use_field_or_value	Field Value	Indica se um campo (padrão) ou um valor é utilizado para especificar o número de avaliações.
trials_field	*campo*	Campo a ser utilizado para especificar o número de avaliações.
trials_value	*Número inteiro*	Valor a ser utilizado para especificar o número de avaliações. Se especificado, o valor mínimo é 1.
use_custom_target_reference	*sinalização*	Indica se a categoria de referência customizada deve ser utilizada para uma variável resposta categórica. O padrão é false.
target_reference_value	*sequência*	Categoria de referência para usar se use_custom_target_reference for true.
dist_link_combination	Nominal Logit GammaLog BinomialLogit PoissonLog BinomialProbit NegbinLog BinomialLogC Custom	Modelos comuns para a distribuição de valores para o destino. Escolha Custom para especificar uma distribuição da lista fornecida por target_distribution.

Tabela 130. Propriedades de glmmnode (continuação)

Propriedades glmmnode	Valores	Descrição da propriedade
target_distribution	Normal Binomial Multinomial Gamma Inverse NegativeBinomial Poisson	Distribuição de valores para destino quando dist_link_combination é Custom.
link_function_type	Identity LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Função de ligação para relacionar valores de destino aos preditores. Se target_distribution for Binomial será possível usar qualquer uma das funções de ligação listadas. Se target_distribution for Multinomial, será possível usar CLOGLOG, CAUCHIT, LOGIT, NLOGLOG ou PROBIT. Se target_distribution for , qualquer coisa diferente de Binomial ou Multinomial, será possível usar IDENTITY, LOG ou POWER.
link_function_param	*number*	Valor do parâmetro da função de ligação a ser utilizado. Somente aplicável se normal_link_function ou link_function_type for POWER.
use_predefined_inputs	*sinalização*	Indica se os campos de efeito fixo devem ser aqueles definidos como envio de dados como campos de entrada (true) ou aqueles de fixed_effects_list (false). O padrão é false.
fixed_effects_list	*estruturado*	Se use_predefined_inputs for false, especifica os campos de entrada para usar como campos de efeito fixo.
use_intercept	*sinalização*	Se true (padrão), inclui o intercepto no modelo.

Tabela 130. Propriedades de glmmnode (continuação)

Propriedades glmmnode	Valores	Descrição da propriedade
random_effects_list	*estruturado*	Lista de campos para especificar como efeitos aleatórios.
regression_weight_field	*campo*	Campo a ser utilizado como campo de ponderação da análise.
use_offset	None offset_value offset_field	Indica como o deslocamento for especificado. Valor None significa que nenhum deslocamento é usado.
offset_value	*number*	Valor a ser usado para deslocamento se use_offset for configurado como offset_value.
offset_field	*campo*	Campo para usar para valor de deslocamento se use_offset for configurado como offset_field.
target_category_order	Ascending Descending Data	Ordenação de classificação para variáveis resposta categórica. Valor Data especifica usando a ordem de classificação localizada nos dados. O patrão é Ascending.
inputs_category_order	Ascending Descending Data	Ordenação de classificação para preditores categóricos. Valor Data especifica usando a ordem de classificação localizada nos dados. O patrão é Ascending.
max_iterations	*Número inteiro*	Número máximo de iterações que o algoritmo executará. Um número inteiro não negativo; o padrão é 100.
confidence_level	*Número inteiro*	O nível de confiança utilizado para calcular estimativas de intervalo dos coeficientes do modelo. Um número inteiro não negativo; o máximo é 100 e o padrão é 95.
degrees_of_freedom_method	Fixed Varied	Especifica como os graus de liberdade são calculados para teste de significância.
test_fixed_effects_coefficients	Model Robust	Método para calcular a matriz de covariância de estimativa de parâmetro.
use_p_converge	*sinalização*	Opção para a convergência de parâmetro.
p_converge	*number*	Em branco, ou qualquer valor positivo.

Tabela 130. Propriedades de glmmnode (continuação)

Propriedades glmmnode	Valores	Descrição da propriedade
p_converge_type	Absolute Relative	
use_l_converge	*sinalização*	Opção para convergência de log da verossimilhança.
l_converge	*number*	Em branco, ou qualquer valor positivo.
l_converge_type	Absolute Relative	
use_h_converge	*sinalização*	Opção para convergência da Hessiana.
h_converge	*number*	Em branco, ou qualquer valor positivo.
h_converge_type	Absolute Relative	
max_fisher_steps	*Número inteiro*	
singularity_tolerance	*number*	
use_model_name	*sinalização*	Indica se deve especificar um nome customizado para o modelo (true) ou para usar o nome gerado pelo sistema (false). O padrão é false.
model_name	*sequência*	Se use_model_name for true, especifica o nome do modelo a ser usado.
confidence	onProbability onIncrease	Base para calcular o valor de confiança de escoragem: a probabilidade prevista mais alta ou a diferença entre as probabilidades mais altas e a segunda probabilidade mais alta prevista.
score_category_probabilities	*sinalização*	Se true, produz probabilidades previstas para destino categóricos. O patrão é false.
max_categories	*Número inteiro*	Se score_category_probabilities for true, especifica o número máximo de categorias a salvar.
score_propensity	*sinalização*	Se true, produz pontuações de propensão para campos de destino de sinalização que indicam probabilidade de resultado "true" para o campo.
emeans	*structure*	Para cada campo categórico na lista de efeitos fixos, especifica se deve produzir médias marginais estimadas.

Tabela 130. Propriedades de glmmnode (continuação)		
Propriedades glmmnode	**Valores**	**Descrição da propriedade**
covariance_list	*structure*	Para cada campo contínuo da lista de efeitos fixos, especifica se deve ser utilizada a média ou um valor customizado quando calcular médias marginais estimadas.
mean_scale	Original Transformed	Especifica se médias marginais estimadas devem ser calculadas com base na escala original do destino (padrão) ou na transformação da função de ligação.
comparison_adjustment_method	LSD SEQBONFERRONI SEQSIDAK	Método de ajuste a ser utilizado ao executar testes de hipótese com diversos contrastes.

Propriedades gle

Um GLE estende o modelo linear para que o destino possa ter uma distribuição não normal, esteja linearmente relacionado aos fatores e covariáveis por meio de uma função de ligação especificada, e para que as observações possam ser correlacionadas. Os modelos lineares generalizados mistos abrangem uma ampla variedade de modelos, desde regressão linear simples até modelos multiníveis complexos para dados longitudinais não normais.

Tabela 131. Propriedades gle		
Propriedades gle	**Valores**	**Descrição da propriedade**
custom_target	*sinalização*	Indica se deve usar o destino definido em nó de envio de dados (false) ou o destino customizado especificado por target_field (true).
target_field	*campo*	Campo para usar como destino se custom_target for true.
use_trials	*sinalização*	Indica se um campo ou um valor adicional que especifica o número de avaliações deve ser utilizado quando a resposta de destino for um número de eventos que ocorrem em um conjunto de avaliações. O padrão é false.
use_trials_field_or_value	Field Value	Indica se um campo (padrão) ou um valor é utilizado para especificar o número de avaliações.
trials_field	*campo*	Campo a ser utilizado para especificar o número de avaliações.

Tabela 131. Propriedades gle (continuação)

Propriedades gle	**Valores**	**Descrição da propriedade**
trials_value	*Número inteiro*	Valor a ser utilizado para especificar o número de avaliações. Se especificado, o valor mínimo é 1.
use_custom_target_reference	*sinalização*	Indica se a categoria de referência customizada deve ser utilizada para uma variável resposta categórica. O padrão é false.
target_reference_value	*sequência*	Categoria de referência para usar se use_custom_target_reference for true.
dist_link_combination	NormalIdentity GammaLog PoissonLog NegbinLog TweedieIdentity NominalLogit BinomialLogit BinomialProbit BinomialLogC CUSTOM	Modelos comuns para a distribuição de valores para o destino. Escolha CUSTOM para especificar uma distribuição da lista fornecida por target_distribution.
target_distribution	Normal Binomial Multinomial Gamma INVERSE_GAUSS NEG_BINOMIAL Poisson TWEEDIE UNKNOWN	Distribuição de valores para destino quando dist_link_combination é Custom.

Tabela 131. Propriedades gle (continuação)

Propriedades gle	Valores	Descrição da propriedade
link_function_type	UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER NEG_BINOMIAL GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT	Função Link para relacionar valores de destino para preditores. Se target_distribution for Binomial, será possível usar: UNKNOWN IDENTITY LOG LOGIT PROBIT COMPL_LOG_LOG POWER LOG_COMPL NEG_LOG_LOG ODDS_POWER Se target_distribution for NEG_BINOMIAL, será possível usar: NEG_BINOMIAL. Se target_distribution for UNKNOWN, será possível usar: GEN_LOGIT CUMUL_LOGIT CUMUL_PROBIT CUMUL_COMPL_LOG_LOG CUMUL_NEG_LOG_LOG CUMUL_CAUCHIT

Tabela 131. Propriedades gle (continuação)

Propriedades gle	Valores	Descrição da propriedade
link_function_param	*number*	O valor de parâmetro tweedie a ser usado. Somente aplicável se normal_link_function ou link_function_type for POWER.
tweedie_param	*number*	Valor do parâmetro da função de ligação a ser utilizado. Somente aplicável se dist_link_combination for configurado como TweedieIdentity ou link_function_type for TWEEDIE.
use_predefined_inputs	*sinalização*	Indica se os campos de efeito de modelo devem ser aqueles definidos como envio de dados como campos de entrada (true) ou aqueles de fixed_effects_list (false).
model_effects_list	*estruturado*	Se use_predefined_inputs for false, especifica os campos de entrada para usar como campos de efeito de modelo.
use_intercept	*sinalização*	Se true (padrão), inclui o intercepto no modelo.
regression_weight_field	*campo*	Campo a ser utilizado como campo de ponderação da análise.
use_offset	None Value Variable	Indica como o deslocamento for especificado. Valor None significa que nenhum deslocamento é usado.
offset_value	*number*	Valor a ser usado para deslocamento se use_offset for configurado como offset_value.
offset_field	*campo*	Campo para usar para valor de deslocamento se use_offset for configurado como offset_field.
target_category_order	Ascending Descending	Ordenação de classificação para variáveis resposta categórica. O padrão é Ascending.
inputs_category_order	Ascending Descending	Ordenação de classificação para preditores categóricos. O padrão é Ascending.
max_iterations	*Número inteiro*	Número máximo de iterações que o algoritmo executará. Um número inteiro não negativo; o padrão é 100.
confidence_level	*number*	O nível de confiança utilizado para calcular estimativas de intervalo dos coeficientes do modelo. Um número inteiro não negativo; o máximo é 100 e o padrão é 95.

Tabela 131. Propriedades gle (continuação)

Propriedades gle	Valores	Descrição da propriedade
test_fixed_effects_coefficients	Model Robust	Método para calcular a matriz de covariância de estimativa de parâmetro.
detect_outliers	*sinalização*	Quando true, o algoritmo localiza valores discrepantes influentes para todas as distribuições, exceto a distribuição multinomial.
conduct_trend_analysis	*sinalização*	Quando true, o algoritmo conduz a análise de tendência para o gráfico de dispersão.
estimation_method	FISHER_SCORING NEWTON_RAPHSON HYBRID	Especifique o algoritmo de estimação de máxima verossimilhança.
max_fisher_iterations	*Número inteiro*	Se estiver usando o FISHER_SCORING estimation_method, o número máximo de iterações. Mínimo 0, máximo 20.
scale_parameter_method	MLE FIXED DEVIANCE PEARSON_CHISQUARE	Especifique o método a ser usado para a estimação do parâmetro de escala.
scale_value	*number*	Somente disponível se scale_parameter_method for configurado como Fixed.
negative_binomial_method	MLE FIXED	Especifique o método a ser usado para a estimação do parâmetro auxiliar de binomial negativa.
negative_binomial_value	*number*	Somente disponível se negative_binomial_method for configurado como Fixed.
non_neg_least_squares	*sinalização*	Seja para executar mínimos quadrados não negativos. O padrão é false.
use_p_converge	*sinalização*	Opção para a convergência de parâmetro.
p_converge	*number*	Em branco, ou qualquer valor positivo.
p_converge_type	*sinalização*	True = absoluto, False = relativo
use_l_converge	*sinalização*	Opção para convergência de log da verossimilhança.
l_converge	*number*	Em branco, ou qualquer valor positivo.
l_converge_type	*sinalização*	True = absoluto, False = relativo
use_h_converge	*sinalização*	Opção para convergência da Hessiana.
h_converge	*number*	Em branco, ou qualquer valor positivo.
h_converge_type	*sinalização*	True = absoluto, False = relativo

Tabela 131. Propriedades gle (continuação)

Propriedades gle	Valores	Descrição da propriedade
max_iterations	*Número inteiro*	Número máximo de iterações que o algoritmo executará. Um número inteiro não negativo; o padrão é 100.
sing_tolerance	*Número inteiro*	
use_model_selection	*sinalização*	Ativa os controles de limite de parâmetro e de método de seleção do modelo.

method	`LASSO`
	`ELASTIC_NET`
	`FORWARD_STEPWISE`
	`RIDGE`
		Determina o método de seleção do modelo usado ou, se estiver usando `Ridge`, o método de regularização.

detect_two_way_interactions	*sinalização*	Quando `True`, o modelo detectará automaticamente interações de duas vias entre campos de entrada. Esse controle só deverá ser ativado se o modelo for efeitos principais apenas (ou seja, em que o usuário não tenha criado nenhum efeito de ordem superior) e se o `method` selecionado for Forward stepwise, Laço ou Rede elástica.
automatic_penalty_params	*sinalização*	Somente disponível se a seleção de modelo `method` for Laço ou Rede elástica. Use esta função para inserir parâmetros de penalidade associados aos métodos de seleção de variáveis Lasso ou Elastic Net. Se `True`, os valores padrão serão usados. Se `False`, os parâmetros de pênalti são valores customizados ativados que podem ser inseridos.
lasso_penalty_param	*number*	Somente disponível se a seleção de modelo `method` for Laço ou Rede elástica e `automatic_penalty_params` for `False`. Especifique o valor de parâmetro de penalidade para Lasso.
elastic_net_penalty_param1	*number*	Somente disponível se a seleção de modelo `method` for Laço ou Rede elástica e `automatic_penalty_params` for `False`. Especifique o valor de parâmetro de penalidade para parâmetro 1 do Elastic Net.
elastic_net_penalty_param2	*number*	Somente disponível se a seleção de modelo `method` for Laço ou Rede elástica e `automatic_penalty_params` for `False`. Especifique o valor de parâmetro de penalidade para parâmetro 2 de Elastic Net.

Tabela 131. Propriedades gle (continuação)		
Propriedades gle	**Valores**	**Descrição da propriedade**
probability_entry	*number*	Somente disponível se o method selecionado for Forward stepwise. Especifique o nível de significância do critério de estatística f para inclusão de efeito.
probability_removal	*number*	Somente disponível se o method selecionado for Forward stepwise. Especifique o nível de significância do critério de estatística f para remoção do efeito.
use_max_effects	*sinalização*	Somente disponível se o method selecionado for Forward stepwise. Ativa o controle max_effects. Quando False, o número padrão de efeitos incluídos deve igualar o número total de efeitos fornecidos ao modelo, menos o intercepto.
max_effects	*Número inteiro*	Especifique o número máximo de efeitos ao usar o método de construção forward stepwise.
use_max_steps	*sinalização*	Ativa o controle max_steps. Quando False, o número padrão de etapas deve igualar três vezes o número de efeitos fornecidos ao modelo, exceto o intercepto.
max_steps	*Número inteiro*	Especifique o número máximo de etapas a serem tomadas ao usar a construção Forward stepwise method.
use_model_name	*sinalização*	Indica se deve especificar um nome customizado para o modelo (true) ou para usar o nome gerado pelo sistema (false). O padrão é false.
model_name	*sequência*	Se use_model_name for true, especifica o nome do modelo a ser usado.
usePI	*sinalização*	Se true, a importância do preditor é calculada.

Propriedades kmeansnode

O nó K-médias armazena em cluster os dados configurados em grupos distintos (ou clusters). O método define um número fixo de clusters, designa registros aos clusters iterativamente e ajusta os centros de cluster até que os refinamentos adicionais não possam mais melhorar o modelo. Em vez de tentar prever um resultado, o *k*-médias usa um processo conhecido como aprendizado não supervisionado para descobrir padrões no conjunto de campos de entrada.

Exemplo

```
node = stream.create("kmeans", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Cholesterol", "BP", "Drug", "Na", "K",
"Age"])
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Kmeans_allinputs")
node.setPropertyValue("num_clusters", 9)
node.setPropertyValue("gen_distance", True)
node.setPropertyValue("cluster_label", "Number")
node.setPropertyValue("label_prefix", "Kmeans_")
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("stop_on", "Custom")
node.setPropertyValue("max_iterations", 10)
node.setPropertyValue("tolerance", 3.0)
node.setPropertyValue("encoding_value", 0.3)
```

Tabela 132. Propriedades kmeansnode

Propriedades kmeansnode	**Valores**	**Descrição da propriedade**
inputs	[*field1 ... fieldN*]	Os modelos de K-médias executam a análise de cluster em um conjunto de campos de entrada, mas não utilizam um campo de destino. Os campos de peso e frequência não são usados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
num_clusters	*number*	
gen_distance	*sinalização*	
cluster_label	String Number	
label_prefix	*sequência*	
mode	Simple Expert	
stop_on	Default Custom	
max_iterations	*number*	
tolerance	*number*	
encoding_value	*number*	
optimize	Speed Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.

propriedades kmeansasnode

K-Médias é um dos algoritmos de armazenamento em cluster mais comumente usados. Ele armazena em cluster pontos de dados em um número predefinido de clusters. O nó K-Means-AS no SPSS Modeler é implementado no Spark. Para obter detalhes sobre algoritmos K-Means, consulte https://spark.apache.org/docs/2.2.0/ml-clustering.html. Observe que o nó K-Means-AS executa a codificação one-hot automaticamente para variáveis categóricas.

Tabela 133. propriedades kmeansasnode

Propriedades kmeansasnode	Valores	Descrição da propriedade
roleUse	*sequência*	Especifique predefined para usar funções predefinidas ou custom para usar designações de campo customizadas. O padrão é predefined.
autoModel	*Booleano*	Especifique true para usar o nome padrão ($S-prediction) para o novo campo de pontuação gerado ou false para usar um nome customizado. O padrão é true.
features	*campo*	Lista dos nomes de campo para entrada quando a propriedade roleUse é configurada como custom.
name	*sequência*	O nome do novo campo de pontuação do gerado quando a propriedade autoModel é configurada como false.
clustersNum	*Número inteiro*	O número de clusters a serem criados. O padrão é 5.
initMode	*sequência*	O algoritmo de inicialização. Os valores possíveis são k-means|| ou random. O padrão é k-means||.
initSteps	*Número inteiro*	O número de etapas de inicialização quando initMode é configurado como k-means||. O padrão é 2.
advancedSettings	*Booleano*	Especifique true para disponibilizar as quatro propriedades a seguir. O patrão é false.
maxIteration	*Número inteiro*	Número máximo de iterações para clusterização. O padrão é 20.
tolerance	*sequência*	A tolerância para parar as iterações. As configurações possíveis são 1.0E-1, 1.0E-2, ..., 1.0E-6. O padrão é 1.0E-4.
setSeed	*Booleano*	Especifique true para usar um valor inicial aleatório customizado. O patrão é false.

Tabela 133. propriedades kmeansasnode (continuação)		
Propriedades kmeansasnode	**Valores**	**Descrição da propriedade**
randomSeed	*Número inteiro*	O valor inicial aleatório customizado quando a propriedade setSeed é true.

Propriedades de knnnode

O nó *k*-Nearest Neighbor (KNN) associa um novo caso à categoria ou valor dos objetos *k* mais próximos dele no espaço do preditor, em que *k* é um número inteiro. Casos semelhantes ficam próximos uns dos outros e os casos diferentes ficam distantes uns dos outros.

Exemplo

```
node = stream.create("knn", "My node")
# Objectives tab
node.setPropertyValue("objective", "Custom")
# Settings tab - Neighbors panel
node.setPropertyValue("automatic_k_selection", False)
node.setPropertyValue("fixed_k", 2)
node.setPropertyValue("weight_by_importance", True)
# Settings tab - Analyze panel
node.setPropertyValue("save_distances", True)
```

Tabela 134. Propriedades de knnnode		
Propriedades knnnode	**Valores**	**Descrição da propriedade**
analysis	PredictTarget IdentifyNeighbors	
objective	Balance Speed Accuracy Custom	
normalize_ranges	*sinalização*	
use_case_labels	*sinalização*	Caixa de seleção para ativar a próxima opção.
case_labels_field	*campo*	
identify_focal_cases	*sinalização*	Caixa de seleção para ativar a próxima opção.
focal_cases_field	*campo*	
automatic_k_selection	*sinalização*	
fixed_k	*Número inteiro*	Ativado apenas se automatic_k_selectio for False.

Tabela 134. Propriedades de knnnode (continuação)

Propriedades knnnode	Valores	Descrição da propriedade
minimum_k	*Número inteiro*	Ativado apenas se automatic_k_selectio for True.
maximum_k	*Número inteiro*	
distance_computation	Euclidean CityBlock	
weight_by_importance	*sinalização*	
range_predictions	Mean Median	
perform_feature_selection	*sinalização*	
forced_entry_inputs	[*field1 ... fieldN*]	
stop_on_error_ratio	*sinalização*	
number_to_select	*Número inteiro*	
minimum_change	*number*	
validation_fold_assign_by_field	*sinalização*	
number_of_folds	*Número inteiro*	Ativado apenas se validation_fold_assign_by_field for False
set_random_seed	*sinalização*	
random_seed	*number*	
folds_field	*campo*	Ativado apenas se validation_fold_assign_by_field for True
all_probabilities	*sinalização*	
save_distances	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades de kohonennode

O nó Kohonen gera um tipo de rede neural que pode ser usada para agrupar o conjunto de dados em grupos distintos. Quando a rede é totalmente treinada, os registros que são semelhantes devem estar próximos no mapa de saída, enquanto os registros que forem diferentes estarão distantes. É possível verificar o número de observações capturadas por cada unidade no nugget do modelo para identificar as unidades fortes. Isso pode dar uma noção do número apropriado de clusters.

Exemplo

```
node = stream.create("kohonen", "My node")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Symbolic Cluster")
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("time", 1)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
node.setPropertyValue("optimize", "Speed")
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("width", 3)
node.setPropertyValue("length", 3)
node.setPropertyValue("decay_style", "Exponential")
node.setPropertyValue("phase1_neighborhood", 3)
node.setPropertyValue("phase1_eta", 0.5)
node.setPropertyValue("phase1_cycles", 10)
node.setPropertyValue("phase2_neighborhood", 1)
node.setPropertyValue("phase2_eta", 0.2)
node.setPropertyValue("phase2_cycles", 75)
```

Tabela 135. Propriedades de kohonennode

Propriedades kohonennode	Valores	Descrição da propriedade
inputs	[*field1 ... fieldN*]	Os modelos de Kohonen utilizam uma lista de campos de entrada, mas não de destino. Os campos de frequência e de ponderação não são utilizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
continue	*sinalização*	
show_feedback	*sinalização*	
stop_on	Default Time	
time	*number*	
optimize	Speed Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.
cluster_label	*sinalização*	

Tabela 135. Propriedades de kohonennode (continuação)		
Propriedades kohonennode	**Valores**	**Descrição da propriedade**
mode	Simple Expert	
width	*number*	
length	*number*	
decay_style	Linear Exponential	
phase1_neighborhood	*number*	
phase1_eta	*number*	
phase1_cycles	*number*	
phase2_neighborhood	*number*	
phase2_eta	*number*	
phase2_cycles	*number*	

Propriedades de linearnode

Os modelos de regressão linear preveem um alvo contínuo baseado em relações lineares entre o alvo e um ou mais preditores.

Exemplo

```
node = stream.create("linear", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Model Selection panel
node.setPropertyValue("model_selection", "BestSubsets")
node.setPropertyValue("criteria_best_subsets", "ASE")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tabela 136. Propriedades de linearnode		
Propriedades linearnode	**Valores**	**Descrição da propriedade**
target	*campo*	Especifica um campo de destino único.
inputs	[*field1 ... fieldN*]	Campos do preditor utilizados pelo modelo.
continue_training_existing_model	*sinalização*	

Tabela 136. Propriedades de linearnode (continuação)

Propriedades linearnode	**Valores**	**Descrição da propriedade**
objective	Standard Bagging Boosting psm	O psm é usado para conjuntos de dados muito grandes e requer uma conexão do servidor.
use_auto_data_preparation	*sinalização*	
confidence_level	*number*	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	
probability_entry	*number*	
probability_removal	*number*	
use_max_effects	*sinalização*	
max_effects	*number*	
use_max_steps	*sinalização*	
max_steps	*number*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
use_random_seed	*sinalização*	
random_seed	*number*	
use_custom_model_name	*sinalização*	

Tabela 136. Propriedades de linearnode (continuação)		
Propriedades linearnode	**Valores**	**Descrição da propriedade**
custom_model_name	*sequência*	
use_custom_name	*sinalização*	
custom_name	*sequência*	
tooltip	*sequência*	
keywords	*sequência*	
annotation	*sequência*	

Propriedades de linearasnode

Os modelos de regressão linear preveem um alvo contínuo baseado em relações lineares entre o alvo e um ou mais preditores.

Tabela 137. Propriedades de linearasnode		
Propriedades linearasnode	**Valores**	**Descrição da propriedade**
target	*campo*	Especifica um campo de destino único.
inputs	[*field1 ... fieldN*]	Campos do preditor utilizados pelo modelo.
weight_field	*campo*	Campo de análise utilizado pelo modelo.
custom_fields	*sinalização*	O valor padrão é TRUE.
intercept	*sinalização*	O valor padrão é TRUE.
detect_2way_interaction	*sinalização*	Especifica se uma interação bilateral deve ser considerada ou não. O valor padrão é TRUE.
cin	*number*	O intervalo de confiança usado para calcular estimativas dos coeficientes do modelo. Especifique um valor maior que 0 e menor que 100. O valor padrão é 95.
factor_order	ascending descending	A ordem de classificação para preditores categóricos. O valor padrão é ascending.
var_select_method	ForwardStepwise BestSubsets none	O método de seleção do modelo a ser utilizado. O valor padrão é ForwardStepwise.

Tabela 137. Propriedades de linearasnode (continuação)		
Propriedades `linearasnode`	**Valores**	**Descrição da propriedade**

`criteria_for_forward_stepwise`	`AICC`
	
	`Fstatistics`
	
	`AdjustedRSquare`
	
	`ASE`
		A estatística utilizada para determinar se um efeito deve ser incluído ou removido do modelo. O valor padrão é `AdjustedRSquare`.

`pin`	*number*	O efeito que tem o menor valor p inferior a este limite de `pin` especificado é incluído no modelo. O valor padrão é `0.05`.
`pout`	*number*	Quaisquer efeitos no modelo com um valor p superior a este limite de `pout` especificado são removidos. O valor padrão é `0.10`.
`use_custom_max_effects`	*sinalização*	Especifica se o número máximo de efeitos deve ser usado no modelo final. O valor padrão é `FALSE`.
`max_effects`	*number*	O número máximo de efeitos a ser usado no modelo final. O valor padrão é `1`.
`use_custom_max_steps`	*sinalização*	Especifica se o número máximo de etapas deve ser usado. O valor padrão é `FALSE`.
`max_steps`	*number*	O número máximo de etapas antes de o algoritmo stepwise parar. O valor padrão é `1`.

`criteria_for_best_subsets`	`AICC`
	
	`AdjustedRSquare`
	
	`ASE`
		O modo de critérios a ser usado. O valor padrão é `AdjustedRSquare`.

Propriedades de logregnode

Regressão logística é uma técnica estatística para classificar registros com base em valores de campos de entrada. É análoga à regressão linear mas leva um campo de destino categórico em vez de um intervalo numérico.

Exemplo multinomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["BP", "Cholesterol", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
```

```
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Log_reg Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("logistic_procedure", "Multinomial")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("model_type", "FullFactorial")
node.setPropertyValue("custom_terms", [["BP", "Sex"], ["Age"], ["Na", "K"]])
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("max_steps", 3)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
node.setPropertyValue("delta", 0.03)
# "Output..." section
node.setPropertyValue("summary", True)
node.setPropertyValue("likelihood_ratio", True)
node.setPropertyValue("asymptotic_correlation", True)
node.setPropertyValue("goodness_fit", True)
node.setPropertyValue("iteration_history", True)
node.setPropertyValue("history_steps", 3)
node.setPropertyValue("parameters", True)
node.setPropertyValue("confidence_interval", 90)
node.setPropertyValue("asymptotic_covariance", True)
node.setPropertyValue("classification_table", True)
# "Stepping" options
node.setPropertyValue("min_terms", 7)
node.setPropertyValue("use_max_terms", True)
node.setPropertyValue("max_terms", 10)
node.setPropertyValue("probability_entry", 3)
node.setPropertyValue("probability_removal", 5)
node.setPropertyValue("requirements", "Containment")
```

Exemplo Binomial

```
node = stream.create("logreg", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Cholesterol")
node.setPropertyValue("inputs", ["BP", "Drug", "Age"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "Log_reg Cholesterol")
node.setPropertyValue("multinomial_base_category", "BP")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("binomial_method", "Forwards")
node.setPropertyValue("logistic_procedure", "Binomial")
node.setPropertyValue("binomial_categorical_input", "Sex")
node.setKeyedPropertyValue("binomial_input_contrast", "Sex", "Simple")
node.setKeyedPropertyValue("binomial_input_category", "Sex", "Last")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("scale", "Pearson")
node.setPropertyValue("scale_value", 3.0)
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("tolerance", "1.0E-7")
```

```
# "Convergence..." section
node.setPropertyValue("max_iterations", 50)
node.setPropertyValue("l_converge", "1.0E-3")
node.setPropertyValue("p_converge", "1.0E-7")
# "Output..." section
node.setPropertyValue("binomial_output_display", "at_each_step")
node.setPropertyValue("binomial_goodness_of_fit", True)
node.setPropertyValue("binomial_iteration_history", True)
node.setPropertyValue("binomial_parameters", True)
node.setPropertyValue("binomial_ci_enable", True)
node.setPropertyValue("binomial_ci", 85)
# "Stepping" options
node.setPropertyValue("binomial_removal_criterion", "LR")
node.setPropertyValue("binomial_probability_removal", 0.2)
```

Tabela 138. Propriedades de logregnode

Propriedades logregnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos de regressão logística requerem um único campo de destino e um ou mais campos de entrada. Os campos de frequência e de ponderação não são utilizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
logistic_procedure	Binomial Multinomial	
include_constant	*sinalização*	
mode	Simple Expert	
method	Enter Stepwise Forwards Backwards BackwardsStepwise	
binomial_method	Enter Forwards Backwards	

Tabela 138. Propriedades de logregnode (continuação)		
Propriedades logregnode	**Valores**	**Descrição da propriedade**
model_type	MainEffects FullFactorial Custom	Quando FullFactorial é especificado como o tipo de modelo, os métodos de progresso não serão executados, mesmo se especificados. Em vez disso, Enter será o método usado. Se o tipo de modelo for configurado como Custom, mas nenhum campo customizado for especificado, um modelo de efeitos principais será construído.
custom_terms	[*[BP Sex][BP][Age]*]	
multinomial_base_category	*sequência*	Especifica como a categoria de referência é determinada.
binomial_categorical_input	*sequência*	
binomial_input_contrast	Indicator Simple Difference Helmert Repeated Polynomial Deviation	Propriedade definida como chave para entrada categórica que especifica como o contraste é determinado.
binomial_input_category	First Last	Propriedade definida como chave para entrada categórica que especifica como a categoria de referência é determinada.
scale	None UserDefined Pearson Deviance	
scale_value	*number*	
all_probabilities	*sinalização*	

Tabela 138. Propriedades de logregnode (continuação)

Propriedades logregnode	Valores	Descrição da propriedade
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*number*	
use_max_terms	*sinalização*	
max_terms	*number*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*number*	
probability_removal	*number*	
binomial_probability_entry	*number*	
binomial_probability_removal	*number*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*number*	
max_steps	*number*	

Tabela 138. Propriedades de logregnode (continuação)		
Propriedades logregnode	**Valores**	**Descrição da propriedade**
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*number*	
iteration_history	*sinalização*	
history_steps	*number*	
summary	*sinalização*	
likelihood_ratio	*sinalização*	
asymptotic_correlation	*sinalização*	
goodness_fit	*sinalização*	
parameters	*sinalização*	
confidence_interval	*number*	
asymptotic_covariance	*sinalização*	
classification_table	*sinalização*	
stepwise_summary	*sinalização*	
info_criteria	*sinalização*	
monotonicity_measures	*sinalização*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*sinalização*	

Tabela 138. Propriedades de logregnode (continuação)		
Propriedades logregnode	**Valores**	**Descrição da propriedade**
binomial_parameters	*sinalização*	
binomial_iteration_history	*sinalização*	
binomial_classification_plots	*sinalização*	
binomial_ci_enable	*sinalização*	
binomial_ci	*number*	
binomial_residual	outliers all	
binomial_residual_enable	*sinalização*	
binomial_outlier_threshold	*number*	
binomial_classification_cutoff	*number*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	

Propriedades de lsvmnode

O nó Support Vector Machine (LSVM) linear permite ordenar dados em um de dois grupos sem causar super ajuste. O LSVM é linear e trabalha bem com conjuntos de dados grandes, como aqueles com um número muito grande de registros.

Tabela 139. Propriedades de lsvmnode		
Propriedades lsvmnode	**Valores**	**Descrição da propriedade**
intercept	*sinalização*	Inclui o intercepto no modelo. O valor padrão é True.
target_order	Ascending Descending	Especifica a ordem de classificação para a variável resposta categórica. Ignorado para variáveis resposta contínuas. O padrão é Ascending.

Tabela 139. Propriedades de lsvmnode (continuação)		
Propriedades lsvmnode	**Valores**	**Descrição da propriedade**
precision	*number*	Usado apenas se o nível de medição do campo de destino for Continuous. Especifica o parâmetro relacionado à sensibilidade de perda para regressão. O mínimo é 0 e não há máximo. O valor padrão é 0.1.
exclude_missing_values	*sinalização*	Quando True, um registro é excluído se algum valor único estiver faltando. O valor padrão é False.
penalty_function	L1 L2	Especifica o tipo de função de penalidade usada. O valor padrão é L2.
lambda	*number*	Parâmetro de penalidade (regularização).
calculate_variable_importance	*sinalização*	Para modelos que produzem uma medida de importância apropriada, esta opção exibe um gráfico que indica a importância relativa de cada preditor na estimativa do modelo. Observe que a importância variável pode demorar mais tempo para calcular para alguns modelos, particularmente ao trabalhar com conjuntos de dados grandes, e está desativada por padrão para alguns modelos como um resultado. A importância variável não está disponível para modelos de lista de decisão.

Propriedades de neuralnetnode

Importante: Uma versão mais recente do nó de modelagem Neural Net, com recursos aprimorados, está disponível nesta liberação e é descrita na próxima seção (*neuralnetwork*). Embora ainda seja possível criar e escorar um modelo com a versão anterior, recomenda-se atualizar seus scripts para utilizar a nova versão. Os detalhes da versão anterior são mantidos aqui para referência.

Exemplo

```
node = stream.create("neuralnet", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("targets", ["Drug"])
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
# "Model" tab
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Dynamic")
node.setPropertyValue("train_pct", 30)
node.setPropertyValue("set_random_seed", True)
node.setPropertyValue("random_seed", 12345)
```

```
node.setPropertyValue("stop_on", "Time")
node.setPropertyValue("accuracy", 95)
node.setPropertyValue("cycles", 200)
node.setPropertyValue("time", 3)
node.setPropertyValue("optimize", "Speed")
# "Multiple Method Expert Options" section
node.setPropertyValue("m_topologies", "5 30 5; 2 20 3, 1 10 1")
node.setPropertyValue("m_non_pyramids", False)
node.setPropertyValue("m_persistence", 100)
```

Tabela 140. Propriedades de neuralnetnode

Propriedades neuralnetnode	**Valores**	**Descrição da propriedade**
targets	[*field1 ... fieldN*]	O nó Rede Neural espera um ou mais campos de destino e um ou mais campos de entrada. Os campos de frequência e de ponderação são ignorados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
method	Quick Dynamic Multiple Prune ExhaustivePrune RBFN	
prevent_overtrain	*sinalização*	
train_pct	*number*	
set_random_seed	*sinalização*	
random_seed	*number*	
mode	Simple Expert	
stop_on	Default Accuracy Cycles Time	Modo de parada.
accuracy	*number*	Precisão da parada.
cycles	*number*	Ciclos para treinamento.
time	*number*	Tempo de treinamento (minutos).

Tabela 140. Propriedades de neuralnetnode (continuação)

Propriedades neuralnetnode	Valores	Descrição da propriedade
continue	*sinalização*	
show_feedback	*sinalização*	
binary_encode	*sinalização*	
use_last_model	*sinalização*	
gen_logfile	*sinalização*	
logfile_name	*sequência*	
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	One Two Three	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
persistence	*number*	
m_topologies	*sequência*	
m_non_pyramids	*sinalização*	
m_persistence	*number*	
p_hid_layers	One Two Three	
p_hl_units_one	*number*	
p_hl_units_two	*number*	
p_hl_units_three	*number*	
p_persistence	*number*	
p_hid_rate	*number*	
p_hid_pers	*number*	
p_inp_rate	*number*	
p_inp_pers	*number*	

Tabela 140. Propriedades de neuralnetnode (continuação)		
Propriedades neuralnetnode	**Valores**	**Descrição da propriedade**
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*sinalização*	
r_alpha	*number*	
r_eta	*number*	
optimize	Speed Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.
calculate_variable_importance	*sinalização*	Nota: A propriedade sensitivity_analysis utilizada em liberações anteriores foi descontinuada a favor desta propriedade. A propriedade antiga ainda é suportada, porém calculate_variable_importance é recomendado.
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades neuralnetworknode

O nó Rede Neural usa um modelo simplificado da maneira como o cérebro humano processa informações. Ele funciona simulando um grande número de unidades de processamento simples interconectadas que se assemelham a versões abstratas de neurônios. As redes neurais são poderosos estimadores de função geral e requerem um conhecimento estatístico ou matemático mínimo para treinar ou aplicar.

Exemplo

```
node = stream.create("neuralnetwork", "My node")
# Build Options tab - Objectives panel
node.setPropertyValue("objective", "Standard")
# Build Options tab - Ensembles panel
node.setPropertyValue("combining_rule_categorical", "HighestMeanProbability")
```

Tabela 141. Propriedades neuralnetworknode		
Propriedades neuralnetworknode	**Valores**	**Descrição da propriedade**
targets	[*field1 ... fieldN*]	Especifica campos de destino.

Tabela 141. Propriedades neuralnetworknode (continuação)		
Propriedades neuralnetworknode	**Valores**	**Descrição da propriedade**
inputs	[*field1 ... fieldN*]	Campos do preditor utilizados pelo modelo.
splits	*[field1 ... fieldN*	Especifica o campo ou campos a serem utilizados para modelagem de divisão.
use_partition	*sinalização*	Se um campo de partição for definido, essa opção assegurará que apenas os dados da partição de treinamento sejam utilizados para construir o modelo.
continue	*sinalização*	Continua treinando o modelo existente.
objective	Standard Bagging Boosting psm	O psm é usado para conjuntos de dados muito grandes e requer uma conexão do servidor.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*sinalização*	
first_layer_units	*number*	
second_layer_units	*number*	
use_max_time	*sinalização*	
max_time	*number*	
use_max_cycles	*sinalização*	
max_cycles	*number*	
use_min_accuracy	*sinalização*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	*number*	

Tabela 141. Propriedades neuralnetworknode (continuação)		
Propriedades neuralnetworknode	**Valores**	**Descrição da propriedade**
overfit_prevention_pct	*number*	
use_random_seed	*sinalização*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	*Booleano*	
model_name	*sequência*	
confidence	onProbability onIncrease	
score_category_probabilities	*sinalização*	
max_categories	*number*	
score_propensity	*sinalização*	
use_custom_name	*sinalização*	
custom_name	*sequência*	
tooltip	*sequência*	
keywords	*sequência*	
annotation	*sequência*	

Propriedades de questnode

O nó QUEST fornece um método de classificação binário para construir árvores de decisão, projetadas para reduzir o tempo de processamento necessário para grandes análises de C e R, enquanto também reduz a tendência localizada nos métodos de árvore de classificação para favorecer entradas que permitam mais divisões. Campos de entrada podem ser intervalos numéricos (contínuos), mas o campo de resposta deve ser categórico. Todas as splits são binárias.

Exemplo

```
node = stream.create("quest", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Drug")
node.setPropertyValue("inputs", ["Age", "Na", "K", "Cholesterol", "BP"])
node.setPropertyValue("model_output_type", "InteractiveBuilder")
node.setPropertyValue("use_tree_directives", True)
node.setPropertyValue("max_surrogates", 5)
node.setPropertyValue("split_alpha", 0.03)
node.setPropertyValue("use_percentage", False)
node.setPropertyValue("min_parent_records_abs", 40)
node.setPropertyValue("min_child_records_abs", 30)
node.setPropertyValue("prune_tree", True)
```

```
node.setPropertyValue("use_std_err", True)
node.setPropertyValue("std_err_multiplier", 3)
```

Tabela 142. Propriedades de questnode

Propriedades questnode	Valores	Descrição da propriedade
target	*campo*	Os modelos QUEST requerem um único campo de destino e um ou mais campos de entrada. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
continue_training_existing_model	*sinalização*	
objective	Standard Boosting Bagging psm	O psm é usado para conjuntos de dados muito grandes e requer uma conexão do servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*sinalização*	
tree_directives	*sequência*	
use_max_depth	Default Custom	
max_depth	*Número inteiro*	Profundidade máxima da árvore, de 0 a 1000. Usado apenas se use_max_depth = Custom.
prune_tree	*sinalização*	Poda a árvore para evitar super ajuste.
use_std_err	*sinalização*	Utiliza a diferença máxima em risco (nos Erros Padrão).
std_err_multiplier	*number*	Diferença máxima.
max_surrogates	*number*	Máximo de substitutos.
use_percentage	*sinalização*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*sinalização*	

Tabela 142. Propriedades de questnode (continuação)

Propriedades questnode	Valores	Descrição da propriedade
costs	*estruturado*	Propriedade estruturada.
priors	Data Equal Custom	
custom_priors	*estruturado*	Propriedade estruturada.
adjust_priors	*sinalização*	
trails	*number*	Número de modelos de componente para boosting ou bagging.
set_ensemble_method	Voting HighestProbability HighestMeanProbability	Regra de combinação padrão para variáveis resposta categórica.
range_ensemble_method	Mean Median	Regra de combinação padrão para variáveis resposta contínua.
large_boost	*sinalização*	Aplica boosting em conjuntos de dados muito grandes.
split_alpha	*number*	Nível de significância para divisão.
train_pct	*number*	Conjunto de prevenção ao super ajuste
set_random_seed	*sinalização*	Replica a opção de resultados.
seed	*number*	
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades randomtrees

O Nó de Árvores Aleatórias é semelhante ao nó de C & RT existente; no entanto, o nó Random Trees é projetado para processar big data para criar uma única árvore e exibe o modelo resultante no visualizador de saída que foi adicionado em SPSS Modeler versão 17. O nó de Árvores Aleatórias gera uma árvore de decisão que é usada para prever ou classificar observações futuras. O método usa particionamento recursivo para dividir os registros de treinamento em segmentos, minimizando as impurezas de cada passo, em que um nó na árvore será considerado *puro* se 100% dos casos no nó estiverem dentro de uma categoria específica do campo de destino. Os campos de destino e de entrada podem ser intervalos numéricos ou categóricos (nominais, ordinais ou sinalizadores); todas as divisões são binárias (apenas dois subgrupos).

Tabela 143. Propriedades randomtrees

Propriedades randomtrees	**Valores**	**Descrição da propriedade**
target	*campo*	No nó Árvores aleatórias, os modelos requerem um único destino e um ou mais campos de entrada. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
number_of_models	*Número inteiro*	Determina o número de modelos a serem construídos como parte da modelagem de combinação.
use_number_of_predictors	*sinalização*	Determina se number_of_predictors é usado.
number_of_predictors	*Número inteiro*	Especifica o número de preditores a serem usados ao construir modelos de divisão.
use_stop_rule_for_accuracy	*sinalização*	Determina se a construção de modelo é interrompida quando a precisão não pode ser melhorada.
sample_size	*number*	Reduza esse valor para melhorar o desempenho quando processar conjuntos de dados muito grandes.
handle_imbalanced_data	*sinalização*	Se o destino do modelo for um resultado de flag específico e a razão do resultado desejado para um resultado não desejado for muito pequena, os dados estão desbalanceados e a amostragem de bootstrap que é conduzida pelo modelo poderá afetar a precisão do modelo. Ative a manipulação de dados desbalanceados para que o modelo capture uma proporção maior do resultado desejado e gere um modelo mais forte.

Tabela 143. Propriedades randomtrees (continuação)

Propriedades randomtrees	Valores	Descrição da propriedade
use_weighted_sampling	*sinalização*	Quando *False*, as variáveis de cada nó são selecionadas aleatoriamente com a mesma probabilidade. Quando *True*, as variáveis são ponderadas e selecionadas de modo apropriado.
max_node_number	*Número inteiro*	Número máximo de nós permitidos em árvores individuais. Se o número exceder na próxima divisão, o crescimento da árvore será interrompido.
max_depth	*Número inteiro*	Profundidade máxima da árvore antes de o crescimento parar.
min_child_node_size	*Número inteiro*	Determina o número mínimo de registros permitidos em um nó-filho após o nó pai ser dividido. Se um nó-filho contiver menos registros do que o especificado aqui, o nó pai não será dividido.
use_costs	*sinalização*	
costs	*estruturado*	Propriedade estruturada. O formato é uma lista de 3 valores: o valor real, o valor previsto e o custo se esta predição estiver errada. Por exemplo: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square custom	**Nota:** Ativado somente para destinos ordinais. Configure os valores padrão na matriz de custos.
max_pct_missing	*Número inteiro*	Se a porcentagem de valores omissos em qualquer entrada for maior que o valor especificado aqui, a entrada será excluída. Mínimo 0, máximo 100.
exclude_single_cat_pct	*Número inteiro*	Se um valor de categoria representar uma porcentagem de registros maior que a especificada aqui, o campo inteiro será excluído da construção de modelo. Mínimo 1, máximo 99.
max_category_number	*Número inteiro*	Se o número de categorias em um campo exceder esse valor, o campo será excluído da construção de modelo. O mínimo é 2.

Tabela 143. Propriedades randomtrees (continuação)		
Propriedades randomtrees	**Valores**	**Descrição da propriedade**
min_field_variation	*number*	Se o coeficiente de variação de um campo contínuo for menor que esse valor, o campo será excluído da construção de modelo.
num_bins	*Número inteiro*	Utilizado apenas se os dados forem compostos de entradas contínuas. Configure o número de categorias de frequência igual a ser utilizado para as entradas; as opções são: 2, 4, 5, 10, 20, 25, 50, ou 100.
topN	*Número inteiro*	Especifica o número de regras a relatar. O valor padrão é de 50, com um mínimo de 1 e um máximo de 1000.

Propriedades de regressionnode

A regressão linear é uma técnica estatística comum para resumir dados e fazer previsões ajustes de uma linha reta ou superfície que minimiza as discrepâncias entre valores de saída previstos e reais.

Nota: O nó Regressão deverá ser substituído pelo nó Linear em uma liberação futura. Recomenda-se usar Modelos Lineares para regressão linear de agora em diante.

Exemplo

```
node = stream.create("regression", "My node")
# "Fields" tab
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("target", "Age")
node.setPropertyValue("inputs", ["Na", "K"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_weight", True)
node.setPropertyValue("weight_field", "Drug")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Regression Age")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("method", "Stepwise")
node.setPropertyValue("include_constant", False)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("complete_records", False)
node.setPropertyValue("tolerance", "1.0E-3")
# "Stepping..." section
node.setPropertyValue("stepping_method", "Probability")
node.setPropertyValue("probability_entry", 0.77)
node.setPropertyValue("probability_removal", 0.88)
node.setPropertyValue("F_value_entry", 7.0)
node.setPropertyValue("F_value_removal", 8.0)
# "Output..." section
node.setPropertyValue("model_fit", True)
node.setPropertyValue("r_squared_change", True)
node.setPropertyValue("selection_criteria", True)
node.setPropertyValue("descriptives", True)
node.setPropertyValue("p_correlations", True)
```

```
node.setPropertyValue("collinearity_diagnostics", True)
node.setPropertyValue("confidence_interval", True)
node.setPropertyValue("covariance_matrix", True)
node.setPropertyValue("durbin_watson", True)
```

Tabela 144. Propriedades de regressionnode

Propriedades regressionnode	**Valores**	**Descrição da propriedade**
target	*campo*	Os modelos de regressão requerem um único campo de destino e um ou mais campos de entrada. Um campo de ponderação também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
method	Enter Stepwise Backwards Forwards	
include_constant	*sinalização*	
use_weight	*sinalização*	
weight_field	*campo*	
mode	Simple Expert	
complete_records	*sinalização*	

Tabela 144. Propriedades de regressionnode (continuação)

Propriedades regressionnode	**Valores**	**Descrição da propriedade**
tolerance	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10 1.0E-11 1.0E-12	Utiliza aspas duplas para argumentos.
stepping_method	useP useF	useP : usar probabilidade de F useF: use valor F
probability_entry	*number*	
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*sinalização*	
confidence_interval	*sinalização*	
covariance_matrix	*sinalização*	
collinearity_diagnostics	*sinalização*	
regression_coefficients	*sinalização*	
exclude_fields	*sinalização*	
durbin_watson	*sinalização*	
model_fit	*sinalização*	
r_squared_change	*sinalização*	
p_correlations	*sinalização*	

Tabela 144. Propriedades de regressionnode (continuação)		
Propriedades regressionnode	**Valores**	**Descrição da propriedade**
descriptives	*sinalização*	
calculate_variable_importance	*sinalização*	

Propriedades sequencenode

O nó de Sequência descobre regras de associação em dados sequenciais ou orientados por tempo. Uma sequência é uma lista de conjuntos de itens que tende a ocorrer em uma ordem previsível. Por exemplo, um cliente que compra um aparelho de barbear e uma loção pós-barba pode comprar um creme de barbear na próxima compra. O nó Sequência é baseado no algoritmo de regras de associação CARMA, que usa um método eficiente de duas passagens para localizar sequências.

Exemplo

```
node = stream.create("sequence", "My node")
# "Fields" tab
node.setPropertyValue("id_field", "Age")
node.setPropertyValue("contiguous", True)
node.setPropertyValue("use_time_field", True)
node.setPropertyValue("time_field", "Date1")
node.setPropertyValue("content_fields", ["Drug", "BP"])
node.setPropertyValue("partition", "Test")
# "Model" tab
node.setPropertyValue("use_model_name", True)
node.setPropertyValue("model_name", "Sequence_test")
node.setPropertyValue("use_partitioned_data", False)
node.setPropertyValue("min_supp", 15.0)
node.setPropertyValue("min_conf", 14.0)
node.setPropertyValue("max_size", 7)
node.setPropertyValue("max_predictions", 5)
# "Expert" tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("use_max_duration", True)
node.setPropertyValue("max_duration", 3.0)
node.setPropertyValue("use_pruning", True)
node.setPropertyValue("pruning_value", 4.0)
node.setPropertyValue("set_mem_sequences", True)
node.setPropertyValue("mem_sequences", 5.0)
node.setPropertyValue("use_gaps", True)
node.setPropertyValue("min_item_gap", 20.0)
node.setPropertyValue("max_item_gap", 30.0)
```

Tabela 145. Propriedades sequencenode		
Propriedades sequencenode	**Valores**	**Descrição da propriedade**
id_field	*campo*	Para criar um modelo de Sequência, é necessário especificar um campo de ID, um campo de tempo opcional e um ou mais campos de conteúdo. Os campos de peso e frequência não são usados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
time_field	*campo*	
use_time_field	*sinalização*	
content_fields	[*field1 ... fieldn*]	
contiguous	*sinalização*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple Expert	
use_max_duration	*sinalização*	
max_duration	*number*	
use_gaps	*sinalização*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*sinalização*	
pruning_value	*number*	
set_mem_sequences	*sinalização*	
mem_sequences	*Número inteiro*	

Propriedades de slrmnode

O nó Self-Learning Response Model (SLRM) permite construir um modelo no qual um único novo caso, ou um pequeno número de novos casos, pode ser usado para estimar novamente o modelo sem precisar treinar o modelo outra vez usando todos os dados.

Exemplo

```
node = stream.create("slrm", "My node")
node.setPropertyValue("target", "Offer")
```

```
node.setPropertyValue("target_response", "Response")
node.setPropertyValue("inputs", ["Cust_ID", "Age", "Ave_Bal"])
```

Tabela 146. Propriedades de slrmnode

Propriedades slrmnode	Valores	Descrição da propriedade
target	*campo*	O campo de destino deve ser um campo nominal ou de sinalização. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
target_response	*campo*	O Tipo deve ser sinalizador.
continue_training_existing_model	*sinalização*	
target_field_values	*sinalização*	Utiliza tudo: Usa todos os valores da origem. Especifique: Selecione valores necessários.
target_field_values_specify	*[field1 ... fieldN]*	
include_model_assessment	*sinalização*	
model_assessment_random_seed	*number*	Deve ser um número real.
model_assessment_sample_size	*number*	Deve ser um número real.
model_assessment_iterations	*number*	Número de iterações.
display_model_evaluation	*sinalização*	
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	Ascending Descending	Especifica se ofertas com as escoragens mais altas ou mais baixas serão exibidas primeiro.
model_reliability	*sinalização*	
calculate_variable_importance	*sinalização*	

Propriedades de statisticsmodelnode

O nó Modelo de Estatísticas permite analisar e trabalhar com seus dados executando os procedimentos do IBM SPSS Statistics que produzem o PMML. Este nó requer uma cópia licenciada de IBM SPSS Statistics.

As propriedades desse nó são descritas em “Propriedades de statisticsmodelnode” na página 432.

propriedades stpnode

O nó Spatio-Temporal Prediction (STP) usa dados que contêm dados de localização, campos de entrada para previsão (preditores), um campo de tempo e um campo de resposta. Cada local possui várias linhas nos dados que representam os valores de cada preditor em cada momento da medição. Após os dados serem analisados, elem podem ser usados para prever valores de destino em qualquer local dentro dos dados de forma que são usados na análise.

Tabela 147. propriedades stpnode

propriedades stpnode	**Tipo de dados**	**Descrição da propriedade**
Guia **Campos**		
`target`	*campo*	Este é o campo de destino.
`location`	*campo*	O campo de local para o modelo. Apenas campos geoespaciais são permitidos.
`location_label`	*campo*	O campo categórico a ser usado na saída para rotular os locais escolhidos em `location`
`time_field`	*campo*	O campo de tempo para o modelo. Apenas os campos com medição contínua são permitidos e o tipo de armazenamento deve ser hora, data, registro de data e hora ou número inteiro.
`inputs`	*[field1 ... fieldN]*	Uma lista de campos de entrada.
Guia **Intervalos de Tempo**		

`interval_type_timestamp`	`Years`
	`Quarters`
	`Months`
	`Weeks`
	`Days`
	`Hours`
	`Minutes`
	`Seconds`	

Tabela 147. propriedades stpnode (continuação)

propriedades stpnode	**Tipo de dados**	**Descrição da propriedade**
interval_type_date	Years Quarters Months Weeks Days	
interval_type_time	Hours Minutes Seconds	Limita o número de dias por semana que são levados em conta ao criar o índice de tempo que o STP utiliza para o cálculo
interval_type_integer	Periods (Apenas campos de índice de tempo, armazenamento Número Inteiro)	O intervalo no qual o conjunto de dados será convertido. A seleção disponível é dependente do tipo de armazenamento do campo que é escolhido como o time_field para o modelo.
period_start	*Número inteiro*	
start_month	January February March April May June July August September October November December	O mês em que o modelo começará a indexar (por exemplo, se configurado como March, mas o primeiro registro no conjunto de dados for January, o modelo pulará os dois primeiros registros e iniciará a indexação em março.

Tabela 147. propriedades stpnode (continuação)

propriedades stpnode	**Tipo de dados**	**Descrição da propriedade**
week_begins_on	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	O ponto de início para o índice de tempo criado pelo STP a partir dos dados
days_per_week	*Número inteiro*	O mínimo é 1 e o máximo é 7, em incrementos de 1.
hours_per_day	*Número inteiro*	O número de horas que o modelo conta para um dia. Se isso for configurado como 10, o modelo começará a indexação no horário day_begins_at e continuará a indexação por 10 horas, depois pulará para o próximo valor correspondendo o valor day_begins_at, etc.
day_begins_at	00:00 01:00 02:00 03:00 ... 23:00	Configura o valor de hora na qual o modelo inicia a indexação.

Tabela 147. propriedades stpnode (continuação)

propriedades stpnode	Tipo de dados	Descrição da propriedade
interval_increment	1 2 3 4 5 6 10 12 15 20 30	Essa configuração de incremento é para minutos ou segundos. Isso determina onde o modelo cria índices dos dados. Assim, com um incremento de 30 e tipo de intervalo seconds, o modelo criará um índice por meio dos dados a cada 30 segundos.
data_matches_interval	*Booleano*	Se configurado como N, a conversão dos dados para o interval_type regular ocorrerá antes de o modelo ser construído. Se os seus dados já estão no formato correto, e o interval_type e quaisquer configurações associadas corresponderem seus dados, configure isso para Y para evitar a conversão ou agregação de seus dados. Configurar isso para Y desativa todos os controles da Agregação.

Tabela 147. propriedades stpnode (continuação)		
propriedades stpnode	**Tipo de dados**	**Descrição da propriedade**
agg_range_default	Sum Mean Min Max Median 1stQuartile 3rdQuartile	Isto determina o método de agregação padrão utilizado para campos contínuos. Todos os campos contínuos que não estiverem especificamente incluídos na agregação customizada serão agregados utilizando o método especificado aqui.
custom_agg	[[field, aggregation method],[]..] Demonstração: [['x5' 'FirstQuartile']['x4' 'Sum']]	Propriedade estruturada: Script parameter: custom_agg Por exemplo: set :stpnode.custom_agg = [[field1 function] [field2 function]] Em que function é a função de agregação a ser usada com aquele campo.
Guia **Configurações Básicas**		
include_intercept	*sinalização*	
max_autoregressive_lag	*Número inteiro*	Mínimo 1, máximo 5, em incrementos de 1. Esse é o número de registros anteriores necessários para uma predição. Portanto, se configurado como 5, por exemplo, os cinco registros anteriores são usados para criar uma nova previsão. O número de registros especificado aqui a partir dos dados de construção é incorporado no modelo e, portanto, o usuário não precisa fornecer os dados novamente quando escorar o modelo.

Tabela 147. propriedades stpnode (continuação)		
propriedades stpnode	**Tipo de dados**	**Descrição da propriedade**
estimation_method	Parametric Nonparametric	O método para modelar a matriz de covariâncias espacial
parametric_model	Gaussian Exponential PoweredExponential	Parâmetro de ordem para modelo de covariância espacial Parametric
exponential_power	*number*	Nível de energia para modelo PoweredExponential. Mínimo 1, máximo 2.
Guia **Avançado**		
max_missing_values	*Número inteiro*	A porcentagem máxima permitida de registros com valores ausentes no modelo.
significance	*number*	O nível de significância para teste de hipóteses na construção de modelo. Especifica o valor de significância para todos os testes na estimativa do modelo do STP, incluindo dois testes de Qualidade do ajuste, testes de Efeito F e testes de coeficiente t.
Guia **Saída**		
model_specifications	*sinalização*	
temporal_summary	*sinalização*	
location_summary	*sinalização*	Determina se a tabela Resumo do Local é incluída na saída do modelo.
model_quality	*sinalização*	
test_mean_structure	*sinalização*	
mean_structure_coefficients	*sinalização*	
autoregressive_coefficients	*sinalização*	
test_decay_space	*sinalização*	
parametric_spatial_covariance	*sinalização*	
correlations_heat_map	*sinalização*	
correlations_map	*sinalização*	
location_clusters	*sinalização*	

Tabela 147. propriedades stpnode (continuação)		
propriedades stpnode	**Tipo de dados**	**Descrição da propriedade**
similarity_threshold	*number*	O limite no qual os clusters de saída são considerados semelhantes o suficiente para serem mesclados em um único cluster.
max_number_clusters	*Número inteiro*	O limite superior para o número de clusters que podem ser incluídos na saída de modelo.
Guia **Opções de Modelo**		
use_model_name	*sinalização*	
model_name	*sequência*	
uncertainty_factor	*number*	Mínimo 0, máximo 100. Determina o aumento de incerteza (erro) aplicado às predições no futuro. Ele representa os limites superior e inferior para as predições.

Propriedades de svmnode

O nó Support Vector Machine (SVM) permite ordenar dados em dois grupos sem super ajuste. SVM trabalha bem com conjuntos de dados grandes, como aqueles com um número muito grande de campos de entrada.

Exemplo

```
node = stream.create("svm", "My node")
# Expert tab
node.setPropertyValue("mode", "Expert")
node.setPropertyValue("all_probabilities", True)
node.setPropertyValue("kernel", "Polynomial")
node.setPropertyValue("gamma", 1.5)
```

Tabela 148. Propriedades de svmnode		
Propriedades svmnode	**Valores**	**Descrição da propriedade**
all_probabilities	*sinalização*	

Tabela 148. Propriedades de svmnode (continuação)

Propriedades svmnode	Valores	Descrição da propriedade
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (padrão) 1.0E-4 1.0E-5 1.0E-6	Determina quando parar o algoritmo de otimização.
regularization	*number*	Também conhecido como o parâmetro C.
precision	*number*	Usado apenas se o nível de medição do campo de destino for Continuous.
kernel	RBF(padrão) Polynomial Sigmoid Linear	Tipo de função de kernel usado para a transformação.
rbf_gamma	*number*	Usado apenas se kernel for RBF.
gamma	*number*	Usado apenas se kernel for Polynomial ou Sigmoid.
bias	*number*	
degree	*number*	Usado apenas se kernel for Polynomial.
calculate_variable_importance	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	
adjusted_propensity_partition	Test Validation	

Propriedades de tcmnode

A modelagem causal temporal tenta descobrir relacionamentos causais chave nos dados de séries temporais. Na modelagem causal temporal, você especifica um conjunto de séries de destino e um conjunto de entradas candidatas a esses destinos. Em seguida, o procedimento constrói um modelo de séries temporais autorregressivo para cada destino e inclui somente as entradas que tiverem o relacionamento causal mais significativo com o destino.

Tabela 149. Propriedades de tcmnode

Propriedades tcmnode	**Valores**	**Descrição da propriedade**
custom_fields	*Booleano*	
dimensionlist	[*dimension1 ... dimensionN*]	
data_struct	Multiple Single	
metric_fields	*campos*	
both_target_and_input	[*f1 ... fN*]	
targets	[*f1 ... fN*]	
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	
use_timestamp	Timestamp Period	

Tabela 149. Propriedades de tcmnode (continuação)		
Propriedades tcmnode	**Valores**	**Descrição da propriedade**
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*sequência*	
period_start_value	*Número inteiro*	
num_days_per_week	*Número inteiro*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*Número inteiro*	
start_hour_of_day	*Número inteiro*	
timestamp_increments	*Número inteiro*	

Tabela 149. Propriedades de tcmnode (continuação)

Propriedades tcmnode	**Valores**	**Descrição da propriedade**
cyclic_increments	*Número inteiro*	
cyclic_periods	*lista*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	Same Notsame	
cross_hour	*Booleano*	
aggregate_and_distribute	*lista*	
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	

Tabela 149. Propriedades de tcmnode (continuação)

Propriedades tcmnode	Valores	Descrição da propriedade
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend None	
k_mean_param	*Número inteiro*	
k_median_param	*Número inteiro*	
missing_value_threshold	*Número inteiro*	
conf_level	*Número inteiro*	
max_num_predictor	*Número inteiro*	
max_lag	*Número inteiro*	
epsilon	*number*	
threshold	*Número inteiro*	
is_re_est	*Booleano*	
num_targets	*Número inteiro*	
percent_targets	*Número inteiro*	
fields_display	*lista*	
series_display	*lista*	
network_graph_for_target	*Booleano*	
sign_level_for_target	*number*	
fit_and_outlier_for_target	*Booleano*	
sum_and_para_for_target	*Booleano*	
impact_diag_for_target	*Booleano*	

Tabela 149. Propriedades de tcmnode (continuação)

Propriedades tcmnode	Valores	Descrição da propriedade
impact_diag_type_for_target	Effect Cause Both	
impact_diag_level_for_target	*Número inteiro*	
series_plot_for_target	*Booleano*	
res_plot_for_target	*Booleano*	
top_input_for_target	*Booleano*	
forecast_table_for_target	*Booleano*	
same_as_for_target	*Booleano*	
network_graph_for_series	*Booleano*	
sign_level_for_series	*number*	
fit_and_outlier_for_series	*Booleano*	
sum_and_para_for_series	*Booleano*	
impact_diagram_for_series	*Booleano*	
impact_diagram_type_for_series	Effect Cause Both	
impact_diagram_level_for_series	*Número inteiro*	
series_plot_for_series	*Booleano*	
residual_plot_for_series	*Booleano*	
forecast_table_for_series	*Booleano*	
outlier_root_cause_analysis	*Booleano*	
causal_levels	*Número inteiro*	
outlier_table	Interactive Pivot Both	
rmsp_error	*Booleano*	

Tabela 149. Propriedades de tcmnode (continuação)		
Propriedades tcmnode	**Valores**	**Descrição da propriedade**
bic	*Booleano*	
r_square	*Booleano*	
outliers_over_time	*Booleano*	
series_transormation	*Booleano*	
use_estimation_period	*Booleano*	
estimation_period	Times Observation	
observations	*lista*	
observations_type	Latest Earliest	
observations_num	*Número inteiro*	
observations_exclude	*Número inteiro*	
extend_records_into_future	*Booleano*	
forecastperiods	*Número inteiro*	
max_num_distinct_values	*Número inteiro*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	*Booleano*	
aic	*Booleano*	
rmse	*Booleano*	

propriedades ts

O nó Séries Temporais estima modelos de suavização exponencial, Média Móvel Integrada AutoRegressiva (ARIMA) univariada e ARIMA multivariada (ou função de transferência) para dados de séries temporais e produz previsões do desempenho futuro. Este nó do Time Series é semelhante ao nó do Time Series anterior que foi descontinuado em SPSS Modeler versão 18. No entanto, este nó do Time Series mais recente foi projetado para aproveitar a potência de IBM SPSS Analytic Server para processar big data, e exibir o modelo resultante no visualizador de saída que foi adicionado em SPSS Modeler versão 17.

Tabela 150. propriedades ts		
Propriedades ts	**Valores**	**Descrição da propriedade**
targets	*campo*	O nó Séries Temporais prevê um ou mais destinos, utilizando, opcionalmente, um ou mais campos de entrada como preditores. Os campos de frequência e de ponderação não são utilizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
candidate_inputs	[*field1 ... fieldN*]	Campos de entrada ou de preditores usados pelo modelo.
use_period	*sinalização*	
date_time_field	*campo*	
input_interval	None Unknown Year Quarter Month Week Day Hour Hour_nonperiod Minute Minute_nonperiod Second Second_nonperiod	
period_field	*campo*	

Tabela 150. propriedades ts (continuação)

Propriedades ts	**Valores**	**Descrição da propriedade**
period_start_value	*Número inteiro*	
num_days_per_week	*Número inteiro*	
start_day_of_week	Sunday Monday Tuesday Wednesday Thursday Friday Saturday	
num_hours_per_day	*Número inteiro*	
start_hour_of_day	*Número inteiro*	
timestamp_increments	*Número inteiro*	
cyclic_increments	*Número inteiro*	
cyclic_periods	*lista*	
output_interval	None Year Quarter Month Week Day Hour Minute Second	
is_same_interval	*sinalização*	
cross_hour	*sinalização*	
aggregate_and_distribute	*lista*	

Tabela 150. propriedades ts (continuação)

Propriedades ts	**Valores**	**Descrição da propriedade**
aggregate_default	Mean Sum Mode Min Max	
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_median Linear_trend	
k_span_points	*Número inteiro*	
use_estimation_period	*sinalização*	
estimation_period	Observations Times	
date_estimation	*lista*	Só disponível se você usar date_time_field
period_estimation	*lista*	Só disponível se você usar use_period
observations_type	Latest Earliest	

Tabela 150. propriedades ts (continuação)

Propriedades ts	**Valores**	**Descrição da propriedade**
observations_num	*Número inteiro*	
observations_exclude	*Número inteiro*	
method	ExpertModeler Exsmooth Arima	
expert_modeler_method	ExpertModeler Exsmooth Arima	
consider_seasonal	*sinalização*	
detect_outliers	*sinalização*	
expert_outlier_additive	*sinalização*	
expert_outlier_level_shift	*sinalização*	
expert_outlier_innovational	*sinalização*	
expert_outlier_level_shift	*sinalização*	
expert_outlier_transient	*sinalização*	
expert_outlier_seasonal_additive	*sinalização*	
expert_outlier_local_trend	*sinalização*	
expert_outlier_additive_patch	*sinalização*	
consider_newesmodels	*sinalização*	

Tabela 150. propriedades ts (continuação)

Propriedades ts	**Valores**	**Descrição da propriedade**
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative DampedTrendAdditive DampedTrendMultiplicative MultiplicativeTrendAdditive MultiplicativeSeasonal MultiplicativeTrendMultiplicative MultiplicativeTrend	Especifica o método Suavização exponencial. O padrão é Simple.

Tabela 150. propriedades ts (continuação)

Propriedades ts	**Valores**	**Descrição da propriedade**

futureValue_type_method	Compute
	
	specify
		Se Compute for usado, o sistema calculará os Valores de futuro para o período previsto para cada preditor.
		
		Para cada preditor, é possível escolher em uma lista de funções (em branco, média de pontos recentes, valor mais recente) ou usar specify para inserir valores manualmente. Para especificar campos e propriedades individuais, use a propriedade extend_metric_values. Por exemplo:
		
		`set :ts.futureValue_type_method="specify"`
		`set :ts.extend_metric_values=[{'Market_1', 'USER_SPECIFY', [1,2,3]}, {'Market_2','MOST_RECENT_VALUE', ''}, {'Market_3','RECENT_POINTS_MEAN', ''}]`

exsmooth_transformation_type	None SquareRoot NaturalLog	
arima.p	*Número inteiro*	
arima.d	*Número inteiro*	
arima.q	*Número inteiro*	
arima.sp	*Número inteiro*	
arima.sd	*Número inteiro*	
arima.sq	*Número inteiro*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*sinalização*	

Tabela 150. propriedades ts (continuação)

Propriedades ts	**Valores**	**Descrição da propriedade**
tf_arima.p. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.d. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.q. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.sp. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.sd. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.sq. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.delay. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima.transformation_type. *fieldname*	None SquareRoot NaturalLog	Para funções de transferência.
arima_detect_outliers	*sinalização*	
arima_outlier_additive	*sinalização*	
arima_outlier_level_shift	*sinalização*	
arima_outlier_innovational	*sinalização*	
arima_outlier_transient	*sinalização*	
arima_outlier_seasonal_additive	*sinalização*	
arima_outlier_local_trend	*sinalização*	
arima_outlier_additive_patch	*sinalização*	
max_lags	*Número inteiro*	
cal_PI	*sinalização*	
conf_limit_pct	*real*	
events	*campos*	
continue	*sinalização*	
scoring_model_only	*sinalização*	Utilize para modelos com números muito grandes (dezenas de milhares) de séries temporais.
forecastperiods	*Número inteiro*	
extend_records_into_future	*sinalização*	

Tabela 150. propriedades ts (continuação)		
Propriedades ts	**Valores**	**Descrição da propriedade**
extend_metric_values	*campos*	Permite que você forneça valores futuros para preditores.
conf_limits	*sinalização*	
noise_res	*sinalização*	
max_models_output	*Número inteiro*	Controla quantos modelos são mostrados na saída. O padrão é 10. Os modelos não são mostrados na saída se o número total de modelos construídos exceder esse valor. Os modelos ainda estão disponíveis para pontuação.

propriedades timeseriesnode (descontinuado)

Nota: Esse nó de Séries Temporais original foi descontinuado na versão 18 do SPSS Modeler e substituído pelo novo nó de Séries Temporais que foi projetado para aproveitar a energia do IBM SPSS Analytic Server e processar big data.

O nó Séries Temporais estima modelos de suavização exponencial, Média Móvel Integrada AutoRegressiva (ARIMA) univariada e ARIMA multivariada (ou função de transferência) para dados de séries temporais e produz previsões do desempenho futuro. Um nó Séries Temporais deve ser sempre precedido por um nó Intervalos de Tempo.

Exemplo

```
node = stream.create("timeseries", "My node")
node.setPropertyValue("method", "Exsmooth")
node.setPropertyValue("exsmooth_model_type", "HoltsLinearTrend")
node.setPropertyValue("exsmooth_transformation_type", "None")
```

Tabela 151. Propriedades de timeseriesnode

Propriedades `timeseriesnode`	**Valores**	**Descrição da propriedade**
`targets`	*campo*	O nó Séries Temporais prevê um ou mais destinos, utilizando, opcionalmente, um ou mais campos de entrada como preditores. Os campos de frequência e de ponderação não são utilizados. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
`continue`	*sinalização*	

`method`	`ExpertModeler`
	
	`Exsmooth`
	
	`Arima`
	
	`Reuse`	

`expert_modeler_method`	*sinalização*	
`consider_seasonal`	*sinalização*	
`detect_outliers`	*sinalização*	
`expert_outlier_additive`	*sinalização*	
`expert_outlier_level_shift`	*sinalização*	
`expert_outlier_innovational`	*sinalização*	
`expert_outlier_level_shift`	*sinalização*	
`expert_outlier_transient`	*sinalização*	
`expert_outlier_seasonal_additive`	*sinalização*	
`expert_outlier_local_trend`	*sinalização*	
`expert_outlier_additive_patch`	*sinalização*	

Tabela 151. Propriedades de timeseriesnode (continuação)

Propriedades timeseriesnode	**Valores**	**Descrição da propriedade**
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*Número inteiro*	
arima_d	*Número inteiro*	
arima_q	*Número inteiro*	
arima_sp	*Número inteiro*	
arima_sd	*Número inteiro*	
arima_sq	*Número inteiro*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*sinalização*	
tf_arima_p. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima_d. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima_q. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima_sp. *fieldname*	*Número inteiro*	Para funções de transferência.
tf_arima_sd. *fieldname*	*Número inteiro*	Para funções de transferência.

Tabela 151. Propriedades de timeseriesnode (continuação)		
Propriedades `timeseriesnode`	**Valores**	**Descrição da propriedade**
`tf_arima_sq.` *fieldname*	*Número inteiro*	Para funções de transferência.
`tf_arima_delay.` *fieldname*	*Número inteiro*	Para funções de transferência.

`tf_arima_transformation_type.` *fieldname*	`None`
	
	`SquareRoot`
	
	`NaturalLog`
		Para funções de transferência.

`arima_detect_outlier_mode`	`None`
	
	`Automatic`	

`arima_outlier_additive`	*sinalização*	
`arima_outlier_level_shift`	*sinalização*	
`arima_outlier_innovational`	*sinalização*	
`arima_outlier_transient`	*sinalização*	
`arima_outlier_seasonal_additive`	*sinalização*	
`arima_outlier_local_trend`	*sinalização*	
`arima_outlier_additive_patch`	*sinalização*	
`conf_limit_pct`	*real*	
`max_lags`	*Número inteiro*	
`events`	*campos*	
`scoring_model_only`	*sinalização*	Utilize para modelos com números muito grandes (dezenas de milhares) de séries temporais.

Propriedades de treeas

O nó Tree-AS é semelhante ao nó CHAID existente; no entanto, o nó Tree-AS é projetado para processar big data para criar uma única árvore e exibe o modelo resultante no visualizador de saída que foi adicionado em SPSS Modeler versão 17. O nó gera uma árvore de decisão usando estatísticas qui-quadrado (CHAID) para identificar divisões ideais. Essa utilização do CHAID pode gerar árvores não binárias, o que significa que algumas divisões possuem mais de duas ramificações. Os campos de destino e de entrada podem ser um intervalo numérico (contínuo) ou categóricos. Um CHAID exaustivo é uma modificação de CHAID que faz um trabalho mais profundo de examinar todas as divisões possíveis, porém demora mais tempo para calcular.

Tabela 152. Propriedades de treeas

Propriedades treeas	**Valores**	**Descrição da propriedade**
target	*campo*	No nó Árvore do AS, os modelos CHAID requerem um único destino e um ou mais campos de entrada. Um campo de frequência também pode ser especificado. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
method	chaid exhaustive_chaid	
max_depth	*Número inteiro*	Profundidade máxima da árvore, de 0 a 20. O valor padrão é 5 segundos.
num_bins	*Número inteiro*	Utilizado apenas se os dados forem compostos de entradas contínuas. Configure o número de categorias de frequência igual a ser utilizado para as entradas; as opções são: 2, 4, 5, 10, 20, 25, 50, ou 100.
record_threshold	*Número inteiro*	O número de registros no qual o modelo alternará do uso de valores-p para tamanhos de Efeito ao construir a árvore. O padrão é 1.000.000; aumente ou diminua isso em incrementos de 10.000.
split_alpha	*number*	Nível de significância para divisão. O valor deve estar entre 0,01 e 0,99.
merge_alpha	*number*	Nível de significância para mesclagem. O valor deve estar entre 0,01 e 0,99.
bonferroni_adjustment	*sinalização*	Ajusta valores de significância usando o método de Bonferroni.
effect_size_threshold_cont	*number*	Configure o limite de tamanho do Efeito ao dividir os nós e mesclar as categorias quando usar uma variável resposta contínua. O valor deve estar entre 0,01 e 0,99.
effect_size_threshold_cat	*number*	Configure o limite de tamanho do Efeito ao dividir os nós e mesclar as categorias quando usar uma variável resposta categórica. O valor deve estar entre 0,01 e 0,99.
split_merged_categories	*sinalização*	Permite redivisão de categorias mescladas.
grouping_sig_level	*number*	Utilizado para determinar como os grupos de nós são formados ou como nós incomuns são identificados.

Tabela 152. Propriedades de treeas (continuação)

Propriedades treeas	Valores	Descrição da propriedade
chi_square	pearson likelihood_ratio	Método utilizado para calcular a estatística chi-quadrada: Razão de Verossimilhança ou Pearson
minimum_record_use	use_percentage use_absolute	
min_parent_records_pc	*number*	O valor padrão é 2. O mínimo é 1 e o máximo é 100, em incrementos de 1. O valor da ramificação pai deve ser maior do que a ramificação filha.
min_child_records_pc	*number*	O valor padrão é 1. O mínimo é 1 e o máximo é 100, em incrementos de 1.
min_parent_records_abs	*number*	O valor padrão é 100. O mínimo é 1 e o máximo é 100, em incrementos de 1. O valor da ramificação pai deve ser maior do que a ramificação filha.
min_child_records_abs	*number*	O valor padrão é 50. O mínimo é 1 e o máximo é 100, em incrementos de 1.
epsilon	*number*	Mudança mínima nas frequências de célula esperadas.
max_iterations	*number*	Iterações máximas para convergência.
use_costs	*sinalização*	
costs	*estruturado*	Propriedade estruturada. O formato é uma lista de 3 valores: o valor real, o valor previsto e o custo se esta predição estiver errada. Por exemplo: tree.setPropertyValue("costs", [["drugA", "drugB", 3.0], ["drugX", "drugY", 4.0]])
default_cost_increase	none linear square custom	**Nota:** Ativado somente para destinos ordinais. Configure os valores padrão na matriz de custos.
calculate_conf	*sinalização*	
display_rule_id	*sinalização*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.

Propriedades de twostepnode

O nó TwoStep usa um método de clusterização em dois passos. O primeiro passo faz uma simples passagem pelos dados para compactar os dados de entrada brutos em um conjunto gerenciável de subclusters. O segundo passo usa um método de armazenamento em cluster hierárquico para mesclar progressivamente os subclusters em clusters cada vez maiores. TwoStep tem a vantagem de estimar automaticamente o número ideal de clusters para dados de treinamento. Ele pode manipular tipos mistos de campos e conjuntos grandes de dados de forma eficiente.

Exemplo

```
node = stream.create("twostep", "My node")
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("inputs", ["Age", "K", "Na", "BP"])
node.setPropertyValue("partition", "Test")
node.setPropertyValue("use_model_name", False)
node.setPropertyValue("model_name", "TwoStep_Drug")
node.setPropertyValue("use_partitioned_data", True)
node.setPropertyValue("exclude_outliers", True)
node.setPropertyValue("cluster_label", "String")
node.setPropertyValue("label_prefix", "TwoStep_")
node.setPropertyValue("cluster_num_auto", False)
node.setPropertyValue("max_num_clusters", 9)
node.setPropertyValue("min_num_clusters", 3)
node.setPropertyValue("num_clusters", 7)
```

Tabela 153. Propriedades de twostepnode

Propriedades twostepnode	Valores	Descrição da propriedade
inputs	[*field1 ... fieldN*]	Os modelos de TwoStep utilizam uma lista de campos de entrada, mas não de destino. Os campos de peso e de frequência não são reconhecidos. Consulte o tópico "Propriedades comuns do nó de modelagem" na página 219 para obter informações adicionais.
standardize	*sinalização*	
exclude_outliers	*sinalização*	
percentage	*number*	
cluster_num_auto	*sinalização*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	String Number	
label_prefix	*sequência*	

Tabela 153. Propriedades de twostepnode (continuação)		
Propriedades twostepnode	**Valores**	**Descrição da propriedade**
distance_measure	Euclidean Loglikelihood	
clustering_criterion	AIC BIC	

Propriedades de twostepAS

O Cluster TwoStep é uma ferramenta exploratória projetada para revelar agrupamentos naturais (ou clusters) em um conjunto de dados que, de outra forma, não estariam aparentes. O algoritmo que é utilizado por este procedimento possui vários recursos desejáveis que o diferenciam das técnicas tradicionais de armazenamento em cluster, como manipulação de variáveis categóricas e contínuas, seleção automática do número de clusters e escalabilidade.

Tabela 154. Propriedades de twostepAS		
Propriedades twostepAS	**Valores**	**Descrição da propriedade**
inputs	[*f1 ... fN*]	Os modelos de TwoStepAS utilizam uma lista de campos de entrada, mas não de destino. Os campos de peso e de frequência não são reconhecidos.
use_predefined_roles	booleano	Default=True
use_custom_field_assignments	booleano	Default=False
cluster_num_auto	booleano	Default=True
min_num_clusters	número inteiro	Default=2
max_num_clusters	número inteiro	Default=15
num_clusters	número inteiro	Default=5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Minimum Maximum	

Tabela 154. Propriedades de twostepAS (continuação)

Propriedades twostepAS	**Valores**	**Descrição da propriedade**
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	booleano	
random_seed	número inteiro	
distance_measure	Euclidean Loglikelihood	
include_outlier_clusters	booleano	Default=True
num_cases_in_feature_tree_leaf_is_less_than	número inteiro	Default=10
top_perc_outliers	número inteiro	Default=5
initial_dist_change_threshold	número inteiro	Default=0
leaf_node_maximum_branches	número inteiro	Default=8
non_leaf_node_maximum_branches	número inteiro	Default=8
max_tree_depth	número inteiro	Default=3
adjustment_weight_on_measurement_level	número inteiro	Default=6
memory_allocation_mb	número	Default=512
delayed_split	booleano	Default=True
fields_to_standardize	[*f1 ... fN*]	
adaptive_feature_selection	booleano	Default=True
featureMisPercent	número inteiro	Default=70
coefRange	número	Default=0.05
percCasesSingleCategory	número inteiro	Default=95
numCases	número inteiro	Default=24
include_model_specifications	booleano	Default=True
include_record_summary	booleano	Default=True
include_field_transformations	booleano	Default=True
excluded_inputs	booleano	Default=True
evaluate_model_quality	booleano	Default=True
show_feature_importance bar chart	booleano	Default=True
show_feature_importance_ word_cloud	booleano	Default=True

Tabela 154. Propriedades de twostepAS (continuação)

Propriedades twostepAS	**Valores**	**Descrição da propriedade**
show_outlier_clusters interactive_table_and_chart	booleano	Default=True
show_outlier_clusters_pivot_table	booleano	Default=True
across_cluster_feature_importance	booleano	Default=True
across_cluster_profiles_pivot_table	booleano	Default=True
withinprofiles	booleano	Default=True
cluster_distances	booleano	Default=True
cluster_label	String Number	
label_prefix	String	

Capítulo 14. Propriedades do nó de nugget do Modelo

Os nós de nugget do modelo compartilham as mesmas propriedades comuns que outros nós. Consulte o tópico "Propriedades Comuns do Nó" na página 75 para obter informações adicionais.

Propriedades de applyanomalydetectionnode

Os nós de modelagem de Detecção de Anomalias podem ser utilizados para gerar um nugget do modelo de Detecção de Anomalias. O nome de script deste nugget do modelo é *applyanomalydetectionnode*. Para obter mais informações sobre o script do próprio nó de modelagem, "Propriedades anomalydetectionnode" na página 220

Tabela 155. Propriedades de applyanomalydetectionnode

Propriedades applyanomalydetectionnode	**Valores**	**Descrição da propriedade**

`anomaly_score_method`	`FlagAndScore`
	`FlagOnly`
	`ScoreOnly`
		Determina quais saídas são criadas para escoragem.

`num_fields`	*Número inteiro*	Campos para relatório.
`discard_records`	*sinalização*	Indica se os registros são descartados a partir da saída ou não.
`discard_anomalous_records`	*sinalização*	Indicador que determina se registros anômalos ou *não* anômalos devem ser descartados. O padrão é `off`, significando que registros *não* anômalos são descartados. Caso contrário, se `on`, registros anômalos serão descartados. Esta propriedade é ativada apenas se a propriedade `discard_records` estiver ativada.

Propriedades de applyapriorinode

Os nós de modelagem a priori podem ser utilizados para gerar um nugget do modelo a priori. O nome de script deste nugget do modelo é *applyapriorinode*. Para obter mais informações sobre o script do próprio nó de modelagem, "Propriedades de apriorinode" na página 221

Tabela 156. Propriedades de applyapriorinode

Propriedades applyapriorinode	**Valores**	**Descrição da propriedade**
`max_predictions`	*número (inteiro)*	
`ignore_unmatached`	*sinalização*	
`allow_repeats`	*sinalização*	

Tabela 156. Propriedades de applyapriorinode (continuação)		
Propriedades applyapriorinode	**Valores**	**Descrição da propriedade**
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Support RuleSupport Lift Deployability	

Propriedades de applyassociationrulesnode

O nó de modelagem Regras de Associação pode ser utilizado para gerar um nugget do modelo de regras de associação. O nome de script deste nugget do modelo é *applyassociationrulesnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de associationrulesnode" na página 223.

Tabela 157. Propriedades de applyassociationrulesnode		
propriedades applyassociationrulesnode	**Tipo de dados**	**Descrição da propriedade**
max_predictions	*Número inteiro*	O número máximo de regras que podem ser aplicadas a cada entrada na escoragem.
criterion	Confidence Rulesupport Lift Conditionsupport Deployability	Seleciona a medida usada para determinar a força das regras.
allow_repeats	*Booleano*	Determina se regras com a mesma predição são incluídas na escoragem.
check_input	NoPredictions Predictions NoCheck	

Propriedades de applyautoclassifiernode

Os nós de modelagem Classificador Automático podem ser utilizados para gerar um nugget do modelo de Classificador Automático. O nome do script desse nugget do modelo é *applyautoclassifiernode.*Para obter mais informações sobre o script do próprio nó de modelagem, "Propriedades de autoclassifiernode" na página 226

Tabela 158. Propriedades de applyautoclassifiernode

Propriedades applyautoclassifiernode	**Valores**	**Descrição da propriedade**
flag_ensemble_method	Voting EvaluationWeightedVoting ConfidenceWeightedVoting RawPropensityWeightedVoting HighestConfidence AverageRawPropensity	Especifica o método utilizado para determinar o escore de combinação. Essa configuração se aplicará apenas se o destino selecionado for um campo de sinalização.
flag_evaluation_selection	Accuracy AUC_ROC	Esta opção é apenas para a meta de sinalizador, para decidir qual medida de avaliação é escolhida para votação ponderada pela avaliação.
filter_individual_model_output	*sinalização*	Especifica se os resultados da escoragem de modelos individuais devem ser suprimidos.
is_ensemble_update	*sinalização*	Ativa o modo de aprendizado de máquina automático contínuo, que inclui novos modelos de componentes em um conjunto de modelos automáticos existentes em vez de substituir o modelo automático existente e reavalia as medidas de modelos de componentes existentes usando dados recém-disponíveis
is_auto_ensemble_weights_reevaluation	*sinalização*	Ativa a reavaliação automática de pesos do modelo
use_accumulated_factor	*sinalização*	O fator acumulado é usado para calcular as medidas acumuladas
accumulated_factor	*number (duplo)*	O valor máximo é 0.99e o valor mínimo é 0.85.
use_accumulated_reducing	*sinalização*	Executa a redução do modelo com base no limite acumulado durante a atualização do modelo

Tabela 158. Propriedades de applyautoclassifiernode (continuação)		
Propriedades applyautoclassifiernode	**Valores**	**Descrição da propriedade**
`accumulated_reducing_limit`	*number (duplo)*	O valor máximo é `0.7`e o valor mínimo é `0.1`.
`use_accumulated_weighted_evaluation`	*sinalização*	A medida de avaliação acumulada é usada para votar quando o método de votação ponderado pela avaliação é selecionado para o método de combinação.

`flag_voting_tie_selection`	`Random`
	
	`HighestConfidence`
	
	`RawPropensity`
		Se um método de votação for selecionado, especifica como os empates serão resolvidos. Essa configuração se aplicará apenas se o destino selecionado for um campo de sinalização.

`set_ensemble_method`	`Voting`
	
	`EvaluationWeightedVoting`
	
	`ConfidenceWeightedVoting`
	
	`HighestConfidence`
		Especifica o método utilizado para determinar o escore de combinação. Essa configuração se aplicará apenas se o destino selecionado for um campo de conjunto.

`set_voting_tie_selection`	`Random`
	
	`HighestConfidence`
		Se um método de votação for selecionado, especifica como os empates serão resolvidos. Essa configuração se aplicará apenas se o destino selecionado for um campo nominal.

Propriedades de applyautoclusternode

Os nós de modelagem de Cluster Automático podem ser utilizados para gerar um nugget do modelo de Cluster Automático. O nome de script deste nugget do modelo é *applyautoclusternode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de autoclusternode" na página 229.

Propriedades de applyautonumericnode

Os nós de modelagem Numeração Automática podem ser utilizados para gerar um nugget do modelo de Numeração Automática. O nome do script deste nugget do modelo é *applyautonumericnode.*Para obter mais informações sobre o script do próprio nó de modelagem, "Propriedades de autonumericnode" na página 231

Tabela 159. Propriedades de applyautonumericnode		
Propriedades applyautonumericnode	**Valores**	**Descrição da propriedade**
`calculate_standard_error`	*sinalização*	

Propriedades de applybayesnetnode

Os nós de modelagem Rede Bayesiana podem ser utilizados para gerar um nugget do modelo Rede Bayesiana. O nome de script neste nugget do modelo é *applybayesnetnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de bayesnetnode" na página 233.

Tabela 160. Propriedades de applybayesnetnode

Propriedades applybayesnetnode	Valores	Descrição da propriedade
all_probabilities	*sinalização*	
raw_propensity	*sinalização*	
adjusted_propensity	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applyc50node

Os nós de modelagem C5.0 podem ser utilizados para gerar um nugget do modelo C5.0. O nome de script deste nugget do modelo é *applyc50node*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de c50node" na página 235.

Tabela 161. Propriedades de applyc50node

Propriedades applyc50node	Valores	Descrição da propriedade
sql_generate	udf Never NoMissingValues	Usado para configurar as opções de geração de SQL durante a execução do conjunto de regras. O valor padrão é udf.
calculate_conf	*sinalização*	Disponível quando a geração de SQL está ativada; essa propriedade inclui cálculos de confiança na árvore gerada.
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applycarmanode

Os nós de modelagem CARMA podem ser utilizados para gerar um nugget do modelo CARMA. O nome de script deste nugget do modelo é *applycarmanode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades carmanode" na página 237.

Propriedades de applycartnode

Os nós de modelagem de árvore C&R podem ser utilizados para gerar um nugget do modelo C&R. O nome de script deste nugget do modelo é *applycartnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de cartnode" na página 238.

Tabela 162. Propriedades de applycartnode

Propriedades applycartnode	Valores	Descrição da propriedade
enable_sql_generation	Never MissingValues NoMissingValues	Usado para configurar as opções de geração de SQL durante a execução do conjunto de regras.
calculate_conf	*sinalização*	Disponível quando a geração de SQL está ativada; essa propriedade inclui cálculos de confiança na árvore gerada.
display_rule_id	*sinalização*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applychaidnode

Os nós de modelagem CHAID podem ser utilizados para gerar um nugget do modelo CHAID. O nome de script deste nugget do modelo é *applychaidnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades chaidnode" na página 241.

Tabela 163. Propriedades de applychaidnode

Propriedades applychaidnode	Valores	Descrição da propriedade
enable_sql_generation	Never MissingValues	Usado para configurar as opções de geração de SQL durante a execução do conjunto de regras.
calculate_conf	*sinalização*	
display_rule_id	*sinalização*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applycoxregnode

Os nós de modelagem Cox podem ser utilizados para gerar um nugget do modelo Cox. O nome de script deste nugget do modelo é *applycoxregnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades coxregnode" na página 244.

Tabela 164. propriedades applycoxregnode

Propriedades applycoxregnode	**Valores**	**Descrição da propriedade**
future_time_as	Intervals Fields	
time_interval	*number*	
num_future_times	*Número inteiro*	
time_field	*campo*	
past_survival_time	*campo*	
all_probabilities	*sinalização*	
cumulative_hazard	*sinalização*	

Propriedades de applydecisionlistnode

Os nós de modelagem Lista de Decisão podem ser utilizados para gerar um nugget do modelo Lista de Decisão. O nome de script deste nugget do modelo é *applydecisionlistnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades decisionlistnode" na página 246.

Tabela 165. Propriedades de applydecisionlistnode

Propriedades applydecisionlistnode	**Valores**	**Descrição da propriedade**
enable_sql_generation	*sinalização*	Quando é true, o IBM SPSS Modeler tenta enviar por push o modelo de Lista de Decisão de volta para SQL.
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applydiscriminantnode

Os nós de modelagem Discriminante podem ser utilizados para gerar um nugget do modelo Discriminante. O nome de script deste nugget do modelo é *applydiscriminantnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades discriminantnode" na página 248.

Tabela 166. Propriedades de applydiscriminantnode		
Propriedades applydiscriminantnode	**Valores**	**Descrição da propriedade**
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

propriedades applyextension

Os nós do Modelo de Extensão podem ser usados para gerar um nugget do modelo de Extensão O nome de script desse nugget de modelo é *applyextension*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades extensionmodelnode" na página 250.

Exemplo de Python for Spark

```
#### script example for Python for Spark
applyModel = stream.findByType("extension_apply", None)

score_script = """
import json
import spss.pyspark.runtime
from pyspark.mllib.regression import LabeledPoint
from pyspark.mllib.linalg import DenseVector
from pyspark.mllib.tree import DecisionTreeModel
from pyspark.sql.types import StringType, StructField

cxt = spss.pyspark.runtime.getContext()

if cxt.isComputeDataModelOnly():
    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    cxt.setSparkOutputSchema(_schema)
else:
    df = cxt.getSparkInputData()

    _modelPath = cxt.getModelContentToPath("TreeModel")
    metadata = json.loads(cxt.getModelContentToString("model.dm"))

    schema = df.dtypes[:]
    target = "Drug"
    predictors = ["Age","BP","Sex","Cholesterol","Na","K"]

    lookup = {}
    for i in range(0,len(schema)):
        lookup[schema[i][0]] = i

    def row2LabeledPoint(dm,lookup,target,predictors,row):
        target_index = lookup[target]
        tval = dm[target_index].index(row[target_index])
        pvals = []
        for predictor in predictors:
            predictor_index = lookup[predictor]
            if isinstance(dm[predictor_index],list):
                pval = row[predictor_index] in dm[predictor_index] and
dm[predictor_index].index(row[predictor_index]) or -1
            else:
                pval = row[predictor_index]
            pvals.append(pval)
        return LabeledPoint(tval, DenseVector(pvals))

    # convert dataframe to an RDD containing LabeledPoint
    lps = df.rdd.map(lambda row: row2LabeledPoint(metadata,lookup,target,predictors,row))
    treeModel = DecisionTreeModel.load(cxt.getSparkContext(), _modelPath);
```

```
    # score the model, produces an RDD containing just double values
    predictions = treeModel.predict(lps.map(lambda lp: lp.features))

    def addPrediction(x,dm,lookup,target):
        result = []
        for _idx in range(0, len(x[0])):
            result.append(x[0][_idx])
        result.append(dm[lookup[target]][int(x[1])])
        return result

    _schema = cxt.getSparkInputSchema()
    _schema.fields.append(StructField("Prediction", StringType(), nullable=True))
    rdd2 = df.rdd.zip(predictions).map(lambda x:addPrediction(x, metadata, lookup, target))
    outDF = cxt.getSparkSQLContext().createDataFrame(rdd2, _schema)

    cxt.setSparkOutputData(outDF)
"""
applyModel.setPropertyValue("python_syntax", score_script)
```

Exemplo de R

```
#### script example for R
applyModel.setPropertyValue("r_syntax", """
result<-predict(modelerModel,newdata=modelerData)
modelerData<-cbind(modelerData,result)
var1<-c(fieldName="NaPrediction",fieldLabel="",fieldStorage="real",fieldMeasure="",
fieldFormat="",fieldRole="")
modelerDataModel<-data.frame(modelerDataModel,var1)""")
```

Tabela 167. propriedades applyextension

Propriedades applyextension	Valores	Descrição da propriedade
r_syntax	*sequência*	Sintaxe do script R para escoragem de modelo.
python_syntax	*sequência*	Sintaxe de script Python para pontuação do modelo.
use_batch_size	*sinalização*	Ative o uso do processamento em lote.
batch_size	*Número inteiro*	Especifique o número de registros de dados a serem incluídos em cada lote.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opção para converter os campos de sinalização.
convert_missing	*sinalização*	Opção para converter valores ausentes para o valor NA do R.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.

Propriedades de applyfactornode

Os nós de modelagem PCA/Fator podem ser utilizados para gerar um nugget do modelo PCA/Fator. O nome de script deste nugget do modelo é *applyfactornode*. Nenhuma outra propriedade existe para este

nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “Propriedades factornode” na página 253.

Propriedades de applyfeatureselectionnode

Os nós de modelagem de Seleção de Recurso podem ser utilizados para gerar um nugget do modelo de Seleção de Recurso. O nome do script deste nugget do modelo é *applyfeatureselectionnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “propriedades de featureselectionnode” na página 255.

Tabela 168. Propriedades de applyfeatureselectionnode

Propriedades applyfeatureselectionnode	Valores	Descrição da propriedade
selected_ranked_fields		Especifica quais campos classificados são verificados no navegador do modelo.
selected_screened_fields		Especifica quais campos selecionados são verificados no navegador do modelo.

Propriedades de applygeneralizedlinearnode

Os nós de modelagem Linear Generalizado (genlin) podem ser utilizados para gerar um nugget do modelo Linear Generalizado. O nome de script deste nugget do modelo é *applygeneralizedlinearnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “Propriedades de genlinnode” na página 257.

Tabela 169. Propriedades de applygeneralizedlinearnode

Propriedades applygeneralizedlinearnode	Valores	Descrição da propriedade
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applyglmmnode

Os nós de modelagem GLMM podem ser utilizados para gerar um nugget do modelo GLMM. O nome de script deste nugget do modelo é *applyglmmnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “Propriedades de glmmnode” na página 263.

Tabela 170. Propriedades de applyglmmnode

Propriedades applyglmmnode	Valores	Descrição da propriedade
confidence	onProbability onIncrease	Base para calcular o valor de confiança de escoragem: a probabilidade prevista mais alta ou a diferença entre as probabilidades mais altas e a segunda probabilidade mais alta prevista.

Tabela 170. Propriedades de applyglmmnode (continuação)		
Propriedades applyglmmnode	**Valores**	**Descrição da propriedade**
score_category_probabilities	*sinalização*	Se configurado como True, produz as probabilidades previstas para destinos categóricos. Um campo é criado para cada categoria. O patrão é False.
max_categories	*Número inteiro*	Número máximo de categorias para as quais as probabilidades serão previstas. Usado apenas se score_category_probabilities for True.
score_propensity	*sinalização*	Se configurado como True, produz pontuações de propensão bruta (probabilidade de resultado "True") para modelos com destinos de sinalização. Se as partições estiverem em vigor, também produzirá escores de propensão ajustada com base na partição de teste. O patrão é False.
enable_sql_generation	udf native	Usado para configurar as opções de geração de SQL durante a execução do fluxo. As opções são para enviar por push o banco de dados e a pontuação usando um adaptador de pontuação do SPSS® Modeler Server (se conectado a um banco de dados com um adaptador de pontuação instalado) ou para pontuar no SPSS Modeler. O valor padrão é udf.

Propriedades applygle

O nó de modelagem GLE pode ser usado para gerar um nugget do modelo GLE. O nome do script deste nugget do modelo é *applygle*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “Propriedades gle” na página 268.

Tabela 171. Propriedades applygle		
Propriedades applygle	**Valores**	**Descrição da propriedade**
enable_sql_generation	udf native	Usado para configurar as opções de geração de SQL durante a execução do fluxo. Escolha para retroceder ao banco de dados e escorar utilizando um adaptador de escoragem do SPSS Modeler Server (se estiver conectado a um banco de dados com um adaptador de escoragem instalado) ou escorar dentro do SPSS Modeler.

propriedades applygmm

O nó Mistura Gaussiana pode ser usado para gerar um nugget do modelo Mistura Gaussiana. O nome de script desse nugget de modelo é *applygmm*. As propriedades na tabela a seguir estão disponíveis na versão 18.2.1.1 e mais recente Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades gmm" na página 435.

Tabela 172. propriedades applygmm

propriedades applygmm	Tipo de dados	Descrição da propriedade
centers		
item_count		
total		
dimension		
components		
partition		

Propriedades de applykmeansnode

Os nós de modelagem K-Médias podem ser utilizados para gerar um nugget do modelo K-Médias. O nome de script deste nugget do modelo é *applykmeansnode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades kmeansnode" na página 274.

Propriedades de applyknnnode

Os nós de modelagem KNN podem ser utilizados para gerar um nugget do modelo KNN. O nome de script deste nugget do modelo é *applyknnnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de knnnode" na página 277.

Tabela 173. Propriedades de applyknnnode

Propriedades applyknnnode	Valores	Descrição da propriedade
all_probabilities	*sinalização*	
save_distances	*sinalização*	

Propriedades de applykohonennode

Os nós de modelagem Kohonen podem ser utilizados para gerar um nugget do modelo Kohonen. O nome de script deste nugget do modelo é *applykohonennode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de c50node" na página 235.

Propriedades de applylinearnode

Os nós de modelagem Linear podem ser utilizados para gerar um nugget do modelo Linear. O nome de script deste nugget do modelo é *applylinearnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de linearnode" na página 280.

Tabela 174. Propriedades de applylinearnode		
Propriedades `linear`	**Valores**	**Descrição da propriedade**
`use_custom_name`	*sinalização*	
`custom_name`	*sequência*	

`enable_sql_generation`	`udf`
	
	`native`
	
	`puresql`
		Usado para configurar as opções de geração de SQL durante a execução do fluxo. As opções são para retroceder para o banco de dados e pontuar usando um adaptador de pontuação do SPSS® Modeler Server (se conectado a um banco de dados com um adaptador de pontuação instalado), para pontuar dentro do SPSS Modelerou para retroceder para o banco de dados e pontuar usando SQL.
		
		O valor padrão é `udf`.

Propriedades de applylinearasnode

Os nós de modelagem Linear do AS podem ser utilizados para gerar um nugget do modelo Linear do AS. O nome de script deste nugget do modelo é *applylinearasnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de linearasnode" na página 282.

Tabela 175. Propriedades de applylinearasnode		
Propriedade `applylinearasnode`	**Valores**	**Descrição da propriedade**

`enable_sql_generation`	`udf`
	
	`native`
		O valor padrão é `udf`.

Propriedades de applylogregnode

Os nós de modelagem Regressão Logística podem ser utilizados para gerar um nugget do modelo Regressão Logística. O nome de script deste nugget do modelo é *applylogregnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de logregnode" na página 283.

Tabela 176. propriedades applylogregnode		
Propriedades `applylogregnode`	**Valores**	**Descrição da propriedade**
`calculate_raw_propensities`	*sinalização*	
`calculate_conf`	*sinalização*	
`enable_sql_generation`	*sinalização*	

Propriedades de applylsvmnode

Os nós de modelagem LSVM podem ser utilizados para gerar um nugget do modelo LSVM. O nome de script deste nugget do modelo é *applylsvmnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de lsvmnode" na página 289.

Tabela 177. Propriedades de applylsvmnode

Propriedades applylsvmnode	Valores	Descrição da propriedade
calculate_raw_propensities	*sinalização*	Especifica se os escores de propensão bruta devem ser calculados.
enable_sql_generation	udf native	Especifica se escorar usando o Scoring Adapter (se instalado) ou no processo, ou escorar fora do banco de dados.

Propriedades de applyneuralnetnode

Os nós de modelagem Rede Neural podem ser utilizados para gerar um nugget do modelo Rede Neural. O nome de script deste nugget do modelo é *applyneuralnetnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de neuralnetnode" na página 290.

Cuidado: Uma versão mais recente do nugget Rede Neural, com recursos aprimorados, está disponível nesta liberação e é descrita na próxima seção (*applyneuralnetwork*). Embora a versão anterior ainda esteja disponível, recomenda-se atualizar seus scripts para utilizar a nova versão. Detalhes da versão anterior são mantidos aqui para referência, porém o suporte para ela será removido em uma liberação futura.

Tabela 178. Propriedades de applyneuralnetnode

Propriedades applyneuralnetnode	Valores	Descrição da propriedade
calculate_conf	*sinalização*	Disponível quando a geração de SQL está ativada; essa propriedade inclui cálculos de confiança na árvore gerada.
enable_sql_generation	*sinalização*	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applyneuralnetworknode

Os nós de modelagem Rede Neural podem ser utilizados para gerar um nugget do modelo Rede Neural. O nome de script deste nugget do modelo é *applyneuralnetworknode*. Para obter mais informações sobre o script do próprio nó de modelagem, Propriedades de neuralnetworknode.

Tabela 179. Propriedades de applyneuralnetworknode		
Propriedades applyneuralnetworknode	**Valores**	**Descrição da propriedade**
use_custom_name	*sinalização*	
custom_name	*sequência*	
confidence	onProbability onIncrease	
score_category_probabilities	*sinalização*	
max_categories	*number*	
score_propensity	*sinalização*	
enable_sql_generation	udf native puresql	Usado para configurar as opções de geração de SQL durante a execução do fluxo. As opções são para retroceder para o banco de dados e pontuar usando um adaptador de pontuação do SPSS® Modeler Server (se conectado a um banco de dados com um adaptador de pontuação instalado), para pontuar dentro do SPSS Modelerou para retroceder para o banco de dados e pontuar usando SQL. O valor padrão é udf.

propriedades applyocsvmnode

Os nós SVM de uma classe podem ser usados para gerar um nugget do modelo SVM de uma classe. O nome de script desse nugget de modelo é *applyocsvmnode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “propriedades ocsvmnode” na página 441.

Propriedades de applyquestnode

Os nós de modelagem QUEST podem ser utilizados para gerar um nugget do modelo QUEST. O nome de script deste nugget do modelo é *applyquestnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte “Propriedades de questnode” na página 295.

Tabela 180. Propriedades de applyquestnode		
Propriedades applyquestnode	**Valores**	**Descrição da propriedade**
enable_sql_generation	Never MissingValues NoMissingValues	Usado para configurar as opções de geração de SQL durante a execução do conjunto de regras.
calculate_conf	*sinalização*	

Tabela 180. Propriedades de applyquestnode (continuação)		
Propriedades applyquestnode	**Valores**	**Descrição da propriedade**
display_rule_id	*sinalização*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applyr

Os nós Construção R podem ser utilizados para gerar um nugget do modelo R. O nome de script deste nugget do modelo é *applyr*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de buildr" na página 235.

Tabela 181. Propriedades de applyr		
Propriedades applyr	**Valores**	**Descrição da propriedade**
score_syntax	*sequência*	Sintaxe do script R para escoragem de modelo.
convert_flags	StringsAndDoubles LogicalValues	Opção para converter os campos de sinalização.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	POSIXct POSIXlt	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.
convert_missing	*sinalização*	Opção para converter valores ausentes em valor NA do R.
use_batch_size	*sinalização*	Ativar uso do processamento em lote
batch_size	*Número inteiro*	Especifique o número de registros de dados a serem incluídos em cada lote

Propriedades de applyrandomtrees

O nó de modelagem Árvores aleatórias pode ser usado para gerar um nugget do modelo de Árvores aleatórias. O nome do script desse nugget do modelo é *applyrandomtrees*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades randomtrees" na página 298.

Tabela 182. propriedades applyrandomtrees		
Propriedades applyrandomtrees	**Valores**	**Descrição da propriedade**
calculate_conf	*sinalização*	Esta propriedade inclui cálculos confiança na árvore gerada.
enable_sql_generation	udf native	Usado para configurar as opções de geração de SQL durante a execução do fluxo. Escolha para retroceder ao banco de dados e escorar utilizando um adaptador de escoragem do SPSS Modeler Server (se estiver conectado a um banco de dados com um adaptador de escoragem instalado) ou escorar dentro do SPSS Modeler.

Propriedades de applyregressionnode

Os nós de modelagem Regressão Linear podem ser utilizados para gerar um nugget do modelo Regressão Linear. O nome de script deste nugget do modelo é *applyregressionnode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de regressionnode" na página 300.

Propriedades de applyselflearningnode

Os nós de modelagem Self-Learning Response Model (SLRM) podem ser utilizados para gerar um nugget do modelo SLRM. O nome de script deste nugget do modelo é *applyselflearningnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de slrmnode" na página 304.

Tabela 183. Propriedades de applyselflearningnode		
Propriedades applyselflearningnode	**Valores**	**Descrição da propriedade**
max_predictions	*number*	
randomization	*number*	
scoring_random_seed	*number*	
sort	ascending descending	Especifica se ofertas com as escoragens mais altas ou mais baixas serão exibidas primeiro.
model_reliability	*sinalização*	Leva em conta a opção de confiabilidade do modelo na guia Configurações.

Propriedades de applysequencenode

Os nós de modelagem Sequência podem ser utilizados para gerar um nugget do modelo Sequência. O nome de script deste nugget do modelo é *applysequencenode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades sequencenode" na página 303.

Propriedades de applysvmnode

Os nós de modelagem SVM podem ser utilizados para gerar um nugget do modelo SVM. O nome de script deste nugget do modelo é *applysvmnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de svmnode" na página 312.

Tabela 184. Propriedades de applysvmnode

Propriedades applysvmnode	Valores	Descrição da propriedade
all_probabilities	*sinalização*	
calculate_raw_propensities	*sinalização*	
calculate_adjusted_propensities	*sinalização*	

Propriedades de applystpnode

O nó de modelagem STP pode ser utilizado para gerar um nugget do modelo associado que exibe a saída do modelo no Visualizador de Saída. O nome do script deste nugget do modelo é *applystpnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades stpnode" na página 306.

Tabela 185. Propriedades de applystpnode

propriedades applystpnode	Tipo de dados	Descrição da propriedade
uncertainty_factor	*Booleano*	Mínimo 0, máximo 100.

Propriedades de applytcmnode

Os nós de modelagem Temporal Causal Modeling (TCM) podem ser utilizados para gerar um nugget do modelo TCM. O nome de script deste nugget do modelo é *applytcmnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de tcmnode" na página 314.

Tabela 186. Propriedades de applytcmnode

Propriedades applytcmnode	Valores	Descrição da propriedade
ext_future	*Booleano*	
ext_future_num	*Número inteiro*	
noise_res	*Booleano*	
conf_limits	*Booleano*	
target_fields	*lista*	
target_series	*lista*	

propriedades applyts

O nó de modelagem Séries Temporais pode ser usado para gerar um nugget do modelo Série Temporal. O nome de script desse nugget de modelo é *applyts*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades ts" na página 319.

Tabela 187. propriedades applyts		
Propriedades applyts	**Valores**	**Descrição da propriedade**
extend_records_into_future	*Booleano*	
ext_future_num	*Número inteiro*	
compute_future_values_input	*Booleano*	
forecastperiods	*Número inteiro*	
noise_res	*Booleano*	
conf_limits	*Booleano*	
target_fields	*lista*	
target_series	*lista*	
includeTargets	*campo*	

Propriedades applytimeseriesnode (descontinuado)

O nó de modelagem Séries Temporais pode ser usado para gerar um nugget do modelo Série Temporal. O nome de script deste nugget do modelo é *applytimeseriesnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades timeseriesnode (descontinuado)" na página 327.

Tabela 188. Propriedades de applytimeseriesnode		
Propriedades applytimeseriesnode	**Valores**	**Descrição da propriedade**
calculate_conf	*sinalização*	
calculate_residuals	*sinalização*	

Propriedades de applytreeas

Os nós de modelagem Árvore do AS podem ser utilizados para gerar um nugget do modelo Árvore do AS. O nome de script deste nugget do modelo é *applytreenas*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de treeas" na página 330.

Tabela 189. Propriedades de applytreeas		
Propriedades applytreeas	**Valores**	**Descrição da propriedade**
calculate_conf	*sinalização*	Esta propriedade inclui cálculos confiança na árvore gerada.
display_rule_id	*sinalização*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.

Tabela 189. Propriedades de applytreeas (continuação)		
Propriedades applytreeas	**Valores**	**Descrição da propriedade**
enable_sql_generation	udf native	Usado para configurar as opções de geração de SQL durante a execução do fluxo. Escolha para retroceder ao banco de dados e escorar utilizando um adaptador de escoragem do SPSS Modeler Server (se estiver conectado a um banco de dados com um adaptador de escoragem instalado) ou escorar dentro do SPSS Modeler.

Propriedades de applytwostepnode

Os nós de modelagem TwoStep podem ser utilizados para gerar um nugget do modelo TwoStep. O nome de script deste nugget do modelo é *applytwostepnode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de twostepnode" na página 333.

Propriedades de applytwostepAS

Os nós de modelagem TwoStep AS podem ser utilizados para gerar um nugget do modelo TwoStep AS. O nome de script deste nugget do modelo é *applytwostepAS*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de twostepAS" na página 334.

Tabela 190. Propriedades de applytwostepAS		
Propriedades applytwostepAS	**Valores**	**Descrição da propriedade**
enable_sql_generation	udf native	Usado para configurar as opções de geração de SQL durante a execução do fluxo. As opções são para enviar por push o banco de dados e a pontuação usando um adaptador de pontuação do SPSS® Modeler Server (se conectado a um banco de dados com um adaptador de pontuação instalado) ou para pontuar no SPSS Modeler. O valor padrão é udf.

propriedades applyxgboosttreenode

O nó Árvore XGBoost pode ser usado para gerar um nugget do modelo Árvore XGBoost. O nome de script desse nugget de modelo é *applyxgboosttreenode*. As propriedades na tabela a seguir foram incluídas em 18.2.1.1 Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades xgboosttreenode" na página 449.

Tabela 191. propriedades applyxgboosttreenode		
propriedades applyxgboosttreenode	**Tipo de dados**	**Descrição da propriedade**
use_model_name		

Tabela 191. propriedades applyxgboosttreenode (continuação)		
propriedades applyxgboosttreenode	**Tipo de dados**	**Descrição da propriedade**
model_name		

propriedades applyxgboostlinearnode

Os nós XGBoost Linear podem ser usados para gerar um nugget de modelo XGBoost Linear. O nome de script desse nugget de modelo é *applyxgboostlinearnode*. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades xgboostlinearnode" na página 448.

propriedades hdbscannugget

O nó HDBSCAN pode ser usado para gerar um nugget do modelo HDBSCAN.. O nome de script desse nugget de modelo é hdbscannugget. Nenhuma outra propriedade existe para este nugget do modelo. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "propriedades hdbscannode" na página 436.

propriedades kdeapply

O nó de Modelagem KDE pode ser usado para gerar um nugget do modelo KDE. O nome de script desse nugget de modelo é kdeapply. Para obter informações sobre script do próprio nó de modelagem, consulte "propriedades kdemodel" na página 438.

Tabela 192. propriedades kdeapply		
propriedades kdeapply	**Tipo de dados**	**Descrição da propriedade**
outLogDensity Renomeado para out_log_density a partir da versão 18.2.1.1	*Booleano*	Especifique True ou False para incluir ou excluir o valor da densidade de log na saída. O padrão é False.

Capítulo 15. Propriedades do Nó de Modelagem de Banco de Dados

O IBM SPSS Modeler suporta a integração com ferramentas de mineração e modelagem de dados disponíveis a partir de fornecedores de banco de dados, incluindo o Microsoft SQL Server Analysis Services, Oracle Data Mining, e IBM Netezza Analytics. É possível construir e escorar modelos utilizando algoritmos de banco de dados nativo a partir de dentro do aplicativo IBM SPSS Modeler. Os modelos de banco de dados também podem ser criados e manipulados por meio de script utilizando as propriedades descritas nesta seção.

Por exemplo, o fragmento de script a seguir ilustra a criação de um modelo do Microsoft Decision Trees utilizando a interface de script do IBM SPSS Modeler:

```
stream = modeler.script.stream()
msbuilder = stream.createAt("mstreenode", "MSBuilder", 200, 200)

msbuilder.setPropertyValue("analysis_server_name", 'localhost')
msbuilder.setPropertyValue("analysis_database_name", 'TESTDB')
msbuilder.setPropertyValue("mode", 'Expert')
msbuilder.setPropertyValue("datasource", 'LocalServer')
msbuilder.setPropertyValue("target", 'Drug')
msbuilder.setPropertyValue("inputs", ['Age', 'Sex'])
msbuilder.setPropertyValue("unique_field", 'IDX')
msbuilder.setPropertyValue("custom_fields", True)
msbuilder.setPropertyValue("model_name", 'MSDRUG')

typenode = stream.findByType("type", None)
stream.link(typenode, msbuilder)
results = []
msbuilder.run(results)
msapplier = stream.createModelApplierAt(results[0], "Drug", 200, 300)
tablenode = stream.createAt("table", "Results", 300, 300)
stream.linkBetween(msapplier, typenode, tablenode)
msapplier.setPropertyValue("sql_generate", True)
tablenode.run([])
```

Propriedades do Nó de Modelagem para Microsoft

Propriedades do Nó de Modelagem Microsoft

Propriedades Comuns

As propriedades a seguir são comuns para os nós de modelagem do banco de dados Microsoft.

Tabela 193. Propriedades comuns do nó Microsoft

Propriedades Comuns do Nó Microsoft	Valores	Descrição da propriedade
analysis_database_name	*sequência*	Nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host de Serviços de Análise.
use_transactional_data	*sinalização*	Especifica se os dados de entrada estão em formato tabular ou transacional.
inputs	*lista*	Campos de entrada de dados tabulares.

Tabela 193. Propriedades comuns do nó Microsoft (continuação)		
Propriedades Comuns do Nó Microsoft	**Valores**	**Descrição da propriedade**
`target`	*campo*	Campo previsto (não aplicável para os nós Armazenamento em Cluster da MS ou Armazenamento em Cluster de Sequência).
`unique_field`	*campo*	Campo chave.
`msas_parameters`	*estruturado*	Parâmetros de algoritmo. Consulte o tópico "Parâmetros do algoritmo" na página 361 para obter informações adicionais.
`with_drillthrough`	*sinalização*	Com a opção Drill through.

Árvore de decisão MS

Não há propriedades específicas definidas para nós do tipo `mstreenode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Cluster MS

Não há propriedades específicas definidas para nós do tipo `msclusternode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Regras de associação da MS

As propriedades específicas a seguir estão disponíveis para nós do tipo `msassocnode`:

Tabela 194. Propriedades de msassocnode		
Propriedades `msassocnode`	**Valores**	**Descrição da propriedade**
`id_field`	*campo*	Identifica cada transação nos dados.
`trans_inputs`	*lista*	Campos de entrada para dados transacionais.
`transactional_target`	*campo*	Campo previsto (dados transacionais).

Naive Bayes da MS

Não há propriedades específicas definidas para nós do tipo `msbayesnode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Regressão linear da MS

Não há propriedades específicas definidas para nós do tipo `msregressionnode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Rede Neural da MS

Não há propriedades específicas definidas para nós do tipo `msneuralnetworknode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Regressão logística da MS

Não há propriedades específicas definidas para nós do tipo `mslogisticnode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Séries temporais da MS

Não há propriedades específicas definidas para nós do tipo `mstimeseriesnode`. Consulte as propriedades comuns da Microsoft no início desta seção.

Cluster de Sequências da MS

As propriedades específicas a seguir estão disponíveis para nós do tipo `mssequenceclusternode`:

Tabela 195. Propriedades de mssequenceclusternode

Propriedades `mssequenceclusternode`	Valores	Descrição da propriedade
`id_field`	*campo*	Identifica cada transação nos dados.
`input_fields`	*lista*	Campos de entrada para dados transacionais.
`sequence_field`	*campo*	Identificador de sequência.
`target_field`	*campo*	Campo previsto (dados tabulares).

Parâmetros do algoritmo

Cada tipo de modelo de banco de dados Microsoft possui parâmetros específicos que podem ser configurados usando a propriedade `msas_parameters` -- por exemplo:

```
stream = modeler.script.stream()
msregressionnode = stream.findByType("msregression", None)
msregressionnode.setPropertyValue("msas_parameters",
[["MAXIMUM_INPUT_ATTRIBUTES", 255],
["MAXIMUM_OUTPUT_ATTRIBUTES", 255]])
```

Esses parâmetros são derivados do SQL Server. Para ver os parâmetros relevantes para cada nó:

1. Coloque um nó de origem do banco de dados na tela.
2. Abra o nó de origem do banco de dados.
3. Selecione uma origem válida na lista suspensa da **Origem de dados**.
4. Selecione uma tabela válida na lista **Nome da tabela**.
5. Clique em **OK** para fechar o nó de origem do banco de dados.
6. Anexe o nó de modelagem do banco de dados da Microsoft cujas propriedades você deseja listar.
7. Abra o nó de modelagem do banco de dados.
8. Selecione a guia **Especialista**.

As propriedades `msas_parameters` disponíveis para esse nó são exibidas.

Propriedades de Nugget do Modelo da Microsoft

As propriedades a seguir são para os nuggets do modelo criados utilizando os nós de modelagem do banco de dados da Microsoft.

Árvore de decisão MS

Tabela 196. Propriedades da Árvore de Decisão da MS		
Propriedades applymstreenode	**Valores**	**Descrição**
analysis_database_name	*sequência*	Esse nó pode ser pontuado diretamente em um fluxo. Essa propriedade é utilizada para identificar o nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host do servidor de Análise.
datasource	*sequência*	Nome da origem de dados (DSN) do ODBC do SQL Server.
sql_generate	*sinalização* udf	Ativa a geração de SQL.

Regressão linear da MS

Tabela 197. Propriedades da Regressão Linear da MS		
Propriedades applymsregressionnode	**Valores**	**Descrição**
analysis_database_name	*sequência*	Esse nó pode ser pontuado diretamente em um fluxo. Essa propriedade é utilizada para identificar o nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host do servidor de Análise.

Rede Neural da MS

Tabela 198. Propriedades de Rede Neural da MS		
Propriedades applymsneuralnetworknode	**Valores**	**Descrição**
analysis_database_name	*sequência*	Esse nó pode ser pontuado diretamente em um fluxo. Essa propriedade é utilizada para identificar o nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host do servidor de Análise.

Regressão logística da MS

Tabela 199. Propriedades da Regressão Logística da MS		
Propriedades applymslogisticnode	**Valores**	**Descrição**
analysis_database_name	*sequência*	Esse nó pode ser pontuado diretamente em um fluxo. Essa propriedade é utilizada para identificar o nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host do servidor de Análise.

Séries temporais da MS

Tabela 200. Séries Temporais da MS		
Propriedades applymstimeseriesnode	**Valores**	**Descrição**
analysis_database_name	*sequência*	Esse nó pode ser pontuado diretamente em um fluxo. Essa propriedade é utilizada para identificar o nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host do servidor de Análise.
start_from	new_prediction `historical_prediction`	Especifica se predições futuras ou predições históricas devem ser feitas
new_step	*number*	Define período de tempo inicial para predições futuras.
historical_step	*number*	Define período de tempo inicial para predições históricas.
end_step	*number*	Define período de tempo final para predições.

Cluster de Sequências da MS

Tabela 201. Propriedades de Armazenamento em Cluster de Sequências da MS		
Propriedades applymssequenceclusternode	**Valores**	**Descrição**
analysis_database_name	*sequência*	Esse nó pode ser pontuado diretamente em um fluxo. Essa propriedade é utilizada para identificar o nome do banco de dados de Serviços de Análise.
analysis_server_name	*sequência*	Nome do host do servidor de Análise.

Propriedades do Nó de Modelagem para Oracle

Propriedades do Nó de Modelagem Oracle

As propriedades a seguir são comuns para os nós de modelagem do banco de dados Oracle.

Tabela 202. Propriedades comuns do nó Oracle

Propriedades Comuns do Nó Oracle	**Valores**	**Descrição da propriedade**
`target`	*campo*	
`inputs`	*Lista de campos*	
`partition`	*campo*	Campo utilizado para particionar os dados em amostras separadas para os estágios de treinamento, de teste e de validação de construção de modelo.
`datasource`		
`username`		
`password`		
`epassword`		
`use_model_name`	*sinalização*	
`model_name`	*sequência*	Nome customizado para o novo modelo.
`use_partitioned_data`	*sinalização*	Se um campo de partição for definido, essa opção assegurará que apenas os dados da partição de treinamento sejam utilizados para construir o modelo.
`unique_field`	*campo*	
`auto_data_prep`	*sinalização*	Ativa ou desativa o recurso de preparação automática de dados da Oracle (apenas bancos de dados 11g).
`costs`	*estruturado*	Propriedade estruturada no formato: `[[drugA drugB 1.5] [drugA drugC 2.1]]`, em que os argumentos entre [] são custos previstos reais.

`mode`	`Simple`
	
	`Expert`
		Faz com que determinadas propriedades sejam ignorada se configurado para `Simple`, conforme mencionado nas propriedades de nó individual.

`use_prediction_probability`	*sinalização*	
`prediction_probability`	*sequência*	
`use_prediction_set`	*sinalização*	

Oracle Naive Bayes

As propriedades a seguir estão disponíveis para nós do tipo `oranbnode`.

Tabela 203. Propriedades de oranbnode		
Propriedades oranbnode	**Valores**	**Descrição da propriedade**
singleton_threshold	*number*	0.0–1.0.*
pairwise_threshold	*number*	0.0–1.0.*
priors	Data Equal Custom	
custom_priors	*estruturado*	Propriedade estruturada no formato: set :oranbnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* A propriedade será ignorada se mode for configurado para Simple.

Oracle Adaptive Bayes

As propriedades a seguir estão disponíveis para nós do tipo oraabnnode.

Tabela 204. Propriedades de oraabnnode		
Propriedades oraabnnode	**Valores**	**Descrição da propriedade**
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*sinalização*	*
execution_time_limit	*Número inteiro*	O valor deve ser maior que 0.*
max_naive_bayes_predictors	*Número inteiro*	O valor deve ser maior que 0.*
max_predictors	*Número inteiro*	O valor deve ser maior que 0.*
priors	Data Equal Custom	
custom_priors	*estruturado*	Propriedade estruturada no formato: set :oraabnnode.custom_priors = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* A propriedade será ignorada se mode for configurado para Simple.

Support Vector Machines da Oracle

As propriedades a seguir estão disponíveis para nós do tipo orasvmnode.

Tabela 205. Propriedades de orasvmnode

Propriedades orasvmnode	Valores	Descrição da propriedade
active_learning	Enable Disable	
kernel_function	Linear Gaussian System	
normalization_method	zscore minmax none	
kernel_cache_size	*Número inteiro*	Apenas kernel gaussiano. O valor deve ser maior que 0.*
convergence_tolerance	*number*	O valor deve ser maior que 0.*
use_standard_deviation	*sinalização*	Apenas kernel gaussiano.*
standard_deviation	*number*	O valor deve ser maior que 0.*
use_epsilon	*sinalização*	Apenas modelos de regressão.*
epsilon	*number*	O valor deve ser maior que 0.*
use_complexity_factor	*sinalização*	*
complexity_factor	*number*	*
use_outlier_rate	*sinalização*	Apenas variante de Classe Um.*
outlier_rate	*number*	Apenas variante de Classe Um. 0.0–1.0.*
weights	Data Equal Custom	
custom_weights	*estruturado*	Propriedade estruturada no formato: set :orasvmnode.custom_weights = [[drugA 1][drugB 2][drugC 3][drugX 4][drugY 5]]

* A propriedade será ignorada se mode for configurado para Simple.

Modelos lineares generalizados da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oraglmnode.

Tabela 206. Propriedades e oraglmnode

Propriedades oraglmnode	Valores	Descrição da propriedade
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*sinalização*	*
row_weights_field	*campo*	*
save_row_diagnostics	*sinalização*	*
row_diagnostics_table	*sequência*	*
coefficient_confidence	*number*	*
use_reference_category	*sinalização*	*
reference_category	*sequência*	*
ridge_regression	Auto Off On	*
parameter_value	*number*	*
vif_for_ridge	*sinalização*	*

* A propriedade será ignorada se mode for configurado para Simple.

Árvore de decisão da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oradecisiontreenode.

Tabela 207. Propriedades de oradecisiontreenode

Propriedades oradecisiontreenode	Valores	Descrição da propriedade
use_costs	*sinalização*	
impurity_metric	Entropy Gini	
term_max_depth	*Número inteiro*	2–20.*
term_minpct_node	*number*	0.0–10.0.*
term_minpct_split	*number*	0.0–20.0.*

Tabela 207. Propriedades de oradecisiontreenode (continuação)		
Propriedades oradecisiontreenode	**Valores**	**Descrição da propriedade**
term_minrec_node	*Número inteiro*	O valor deve ser maior que 0.*
term_minrec_split	*Número inteiro*	O valor deve ser maior que 0.*
display_rule_ids	*sinalização*	*

* A propriedade será ignorada se mode for configurado para Simple.

O-Cluster Oracle

As propriedades a seguir estão disponíveis para nós do tipo oraoclusternode.

Tabela 208. Propriedades de oraoclusternode		
Propriedades oraoclusternode	**Valores**	**Descrição da propriedade**
max_num_clusters	*Número inteiro*	O valor deve ser maior do que 0.
max_buffer	*Número inteiro*	O valor deve ser maior que 0.*
sensitivity	*number*	0.0–1.0.*

* A propriedade será ignorada se mode for configurado para Simple.

K-Médias da Oracle

As propriedades a seguir estão disponíveis para nós do tipo orakmeansnode.

Tabela 209. Propriedades de orakmeansnode		
Propriedades orakmeansnode	**Valores**	**Descrição da propriedade**
num_clusters	*Número inteiro*	O valor deve ser maior do que 0.
normalization_method	zscore minmax none	
distance_function	Euclidean Cosine	
iterations	*Número inteiro*	0–20.*
conv_tolerance	*number*	0.0–0.5.*
split_criterion	Variance Size	O padrão é Variance.*
num_bins	*Número inteiro*	O valor deve ser maior que 0.*
block_growth	*Número inteiro*	1–5.*
min_pct_attr_support	*number*	0.0–1.0.*

* A propriedade será ignorada se mode for configurado para Simple.

NMF da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oranmfnode.

Tabela 210. Propriedades de oranmfnode		
Propriedades oranmfnode	**Valores**	**Descrição da propriedade**
normalization_method	minmax none	
use_num_features	*sinalização*	*
num_features	*Número inteiro*	0–1. O valor padrão é estimado a partir dos dados pelo algoritmo.*
random_seed	*number*	*
num_iterations	*Número inteiro*	0–500.*
conv_tolerance	*number*	0.0–0.5.*
display_all_features	*sinalização*	*

* A propriedade será ignorada se mode for configurado para Simple.

A priori da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oraapriorinode.

Tabela 211. Propriedades de oraapriorinode		
Propriedades oraapriorinode	**Valores**	**Descrição da propriedade**
content_field	*campo*	
id_field	*campo*	
max_rule_length	*Número inteiro*	2–20.
min_confidence	*number*	0.0–1.0.
min_support	*number*	0.0–1.0.
use_transactional_data	*sinalização*	

Oracle Minimum Description Length (MDL)

Não há propriedades específicas definidas para nós do tipo oramdlnode. Consulte as propriedades comuns da Oracle no início desta seção.

Oracle Attribute Importance (AI)

As propriedades a seguir estão disponíveis para nós do tipo oraainode.

Tabela 212. Propriedades de oraainode		
Propriedades oraainode	**Valores**	**Descrição da propriedade**
custom_fields	*sinalização*	Se true, permite especificar campos de destino, de entrada e outros campos para o nó atual. Se false, as configurações atuais de um nó Tipo de envio de dados serão utilizadas.
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*sinalização*	Quando selection_mode é configurado como ImportanceLevel, especifica se selecionará campos importantes.
important_label	*sequência*	Especifica o rótulo para a classificação "importante".
select_marginal	*sinalização*	Quando selection_mode é configurado como ImportanceLevel, especifica se selecionará campos marginais.
marginal_label	*sequência*	Especifica o rótulo para a classificação "marginal".
important_above	*number*	0.0–1.0.
select_unimportant	*sinalização*	Quando selection_mode é configurado como ImportanceLevel, especifica se selecionará campos não importantes.
unimportant_label	*sequência*	Especifica o rótulo para a classificação "não importante".
unimportant_below	*number*	0.0–1.0.
importance_value	*number*	Quando selection_mode é configurado como ImportanceValue, especifica o valor de corte a ser usado. Aceita valores de 0 a 100.
top_n	*number*	Quando selection_mode é configurado como TopN, especifica o valor de corte a ser usado. Aceita valores de 0 a 1000.

Propriedades de Nugget do Modelo da Oracle

As propriedades a seguir são para os nuggets do modelo criados utilizando os modelos da Oracle.

Oracle Naive Bayes

Não há propriedades específicas definidas para nós do tipo applyoranbnode.

Oracle Adaptive Bayes

Não há propriedades específicas definidas para nós do tipo applyoraabnnode.

Support Vector Machines da Oracle

Não há propriedades específicas definidas para nós do tipo `applyorasvmnode`.

Árvore de decisão da Oracle

As propriedades a seguir estão disponíveis para nós do tipo `applyoradecisiontreenode`.

Tabela 213. Propriedades do applyoradecisiontreenode		
Propriedades `applyoradecisiontreenode`	**Valores**	**Descrição da propriedade**
`use_costs`	*sinalização*	
`display_rule_ids`	*sinalização*	

O-Cluster Oracle

Não há propriedades específicas definidas para nós do tipo `applyoraoclusternode`.

K-Médias da Oracle

Não há propriedades específicas definidas para nós do tipo `applyorakmeansnode`.

NMF da Oracle

A propriedade a seguir está disponível para nós do tipo `applyoranmfnode`:

Tabela 214. Propriedades de applyoranmfnode		
Propriedades `applyoranmfnode`	**Valores**	**Descrição da propriedade**
`display_all_features`	*sinalização*	

A priori da Oracle

O nugget do modelo não pode ser aplicado no script.

MDL da Oracle

O nugget do modelo não pode ser aplicado no script.

Propriedades do nó para IBM Netezza Analytics Modelagem

Propriedades do Nó de Modelagem Netezza

As propriedades a seguir são comuns para os nós de modelagem do banco de dados do IBM Netezza.

Tabela 215. Propriedades comuns do nó Netezza		
Propriedades Comuns do Nó Netezza	**Valores**	**Descrição da propriedade**
`custom_fields`	*sinalização*	Se true, permite especificar campos de destino, de entrada e outros campos para o nó atual. Se false, as configurações atuais de um nó Tipo de envio de dados serão utilizadas.
`inputs`	*[field1 ... fieldN]*	Campos de entrada ou de preditores usados pelo modelo.

Tabela 215. Propriedades comuns do nó Netezza (continuação)		
Propriedades Comuns do Nó Netezza	**Valores**	**Descrição da propriedade**
`target`	*campo*	Campo de destino (contínuo ou categórico).
`record_id`	*campo*	Campo a ser utilizado como identificador de registro exclusivo.
`use_upstream_connection`	*sinalização*	Se true (padrão), os detalhes da conexão especificados em um nó de envio de dados. Não utilizado se `move_data_to_connection` for especificado.
`move_data_connection`	*sinalização*	Se true, move os dados para o banco de dados especificado por `connection`. Não utilizado se `use_upstream_connection` for especificado.
`connection`	*estruturado*	A sequência de conexões com o banco de dados Netezza no qual o modelo é armazenado. Propriedade estruturada no formato: `['odbc' '<dsn>' '<username>' '<psw>' '<catname>' '<conn_attribs>' [true|false]]` em que: `<dsn>` é o nome da origem de dados `<username>` e `<psw>` são o nome do usuário e a senha para o banco de dados `<catname>` é o nome do catálogo `<conn_attribs>` são os atributos de conexão `true | false` indica se a senha é necessária.
`table_name`	*sequência*	Nome da tabela de banco de dados na qual o modelo deve ser armazenado.
`use_model_name`	*sinalização*	Se true, utilizará o nome especificado por `model_name` como o nome do modelo, caso contrário, o nome do modelo será criado pelo sistema.
`model_name`	*sequência*	Nome customizado para o novo modelo.
`include_input_fields`	*sinalização*	Se true, transmitirá todos os campos de entrada de recebimento de dados, caso contrário, transmitirá apenas o `record_id` e campos gerados pelo modelo.

Árvore de decisão Netezza

As propriedades a seguir estão disponíveis para nós do tipo `netezzadectreenode`.

Tabela 216. Propriedades de netezzadectreenode

Propriedades netezzadectreenode	**Valores**	**Descrição da propriedade**
impurity_measure	Entropy Gini	A medição de impureza, utilizada para avaliar o melhor local para dividir a árvore.
max_tree_depth	*Número inteiro*	Número máximo de níveis até o qual árvore pode crescer. O padrão é 62 (o máximo possível).
min_improvement_splits	*number*	Melhoria mínima na impureza para ocorrer a divisão. O padrão é 0,01.
min_instances_split	*Número inteiro*	Número mínimo de registros não divididos restantes antes que a divisão possa ocorrer. O padrão é 2 (o mínimo possível).
weights	*estruturado*	Peso relativo para classes. Propriedade estruturada no formato: `set :netezza_dectree.weights = [[drugA 0.3][drugB 0.6]]` O peso padrão é 1 para todas as classes.
pruning_measure	Acc wAcc	O padrão é Acc (precisão). Um wAcc (precisão ponderada) alternativo leva em conta os pesos de classe ao aplicar a remoção.
prune_tree_options	allTrainingData `partitionTrainingData` useOtherTable	O padrão é utilizar allTrainingData para estimar a precisão do modelo. Utilize partitionTrainingData para especificar uma porcentagem de dados de treinamento a serem utilizados ou useOtherTable para utilizar um conjunto de dados de treinamento de uma tabela de banco de dados especificada.
perc_training_data	*number*	Se prune_tree_options for configurado para partitionTrainingData, especifica a porcentagem de dados a serem utilizados no treinamento.

Tabela 216. Propriedades de netezzadectreenode (continuação)		
Propriedades netezzadectreenode	**Valores**	**Descrição da propriedade**
prune_seed	*Número inteiro*	Valor inicial aleatório a ser utilizado para replicar os resultados da análise quando prune_tree_options for configurado para partitionTrainingData; o padrão é 1.
pruning_table	*sequência*	Nome da tabela de um conjunto de remoção separado para estimar a precisão do modelo.
compute_probabilities	*sinalização*	Se true, produz um campo de nível de confiança (probabilidade), assim como o campo de predição.

K-Médias Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzakmeansnode.

Tabela 217. Propriedades de netezzakmeansnode		
Propriedades netezzakmeansnode	**Valores**	**Descrição da propriedade**
distance_measure	Euclidean Manhattan Canberra maximum	Método a ser utilizado para medir a distância entre pontos de dados.
num_clusters	*Número inteiro*	Número de clusters a serem criados; o padrão é 3.
max_iterations	*Número inteiro*	Número de iterações de algoritmo após o qual o treinamento do modelo é interrompido; o padrão é 5.
rand_seed	*Número inteiro*	Valor inicial aleatório a ser utilizado para replicar os resultados da análise; o padrão é 12345.

Rede bayesiana Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzabayesnode.

Tabela 218. Propriedades de netezzabayesnode		
Propriedades netezzabayesnode	**Valores**	**Descrição da propriedade**
base_index	*Número inteiro*	Identificador numérico designado ao primeiro campo de entrada para gerenciamento interno; o padrão é 777.

Tabela 218. Propriedades de netezzabayesnode (continuação)		
Propriedades netezzabayesnode	**Valores**	**Descrição da propriedade**
sample_size	*Número inteiro*	Tamanho da amostra a ser usado se o número de atributos for muito grande; o padrão é 10.000.
display_additional_information	*sinalização*	Se true, exibe informações adicionais do progresso em uma caixa de diálogo de mensagens.
type_of_prediction	best neighbors nn-neighbors	Tipos de algoritmo de predição a ser utilizado: best (vizinho mais correlacionado), neighbors (predição ponderada dos vizinhos) ou nn-neighbors (vizinhos não nulos).

Naive Bayes Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzanaivebayesnode.

Tabela 219. Propriedades de netezzanaivebayesnode		
Propriedades netezzanaivebayesnode	**Valores**	**Descrição da propriedade**
compute_probabilities	*sinalização*	Se true, produz um campo de nível de confiança (probabilidade), assim como o campo de predição.
use_m_estimation	*sinalização*	Se true, utiliza a técnica m-estimativa para evitar probabilidades zero durante a estimativa.

KNN Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzaknnnode.

Tabela 220. Propriedades de netezzaknnnode		
Propriedades netezzaknnnode	**Valores**	**Descrição da propriedade**
weights	*estruturado*	Propriedade estruturada utilizada para designar pesos para classes individuais. Exemplo: set :netezzaknnnode.weights = [[drugA 0.3][drugB 0.6]]
distance_measure	Euclidean Manhattan Canberra Maximum	Método a ser utilizado para medir a distância entre pontos de dados.
num_nearest_neighbors	*Número inteiro*	Número de vizinhos mais próximos para um caso específico; o padrão é 3.

Tabela 220. Propriedades de netezzaknnnode (continuação)		
Propriedades netezzaknnnode	**Valores**	**Descrição da propriedade**
standardize_measurements	*sinalização*	Se true, padroniza as medições para campos de entrada contínuos antes de calcular valores de distância.
use_coresets	*sinalização*	Se true, utiliza amostragem do conjunto principal para acelerar o cálculo de conjuntos de dados grandes.

Cluster de divisão Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzadivclusternode.

Tabela 221. Propriedades de netezzadivclusternode		
Propriedades netezzadivclusternode	**Valores**	**Descrição da propriedade**
distance_measure	Euclidean Manhattan Canberra Maximum	Método a ser utilizado para medir a distância entre pontos de dados.
max_iterations	*Número inteiro*	Número máximo de iterações de algoritmo para executar antes de o treinamento de modelo ser interrompido; o padrão é 5.
max_tree_depth	*Número inteiro*	Número máximo de níveis até o qual o conjunto de dados pode ser subdividido; o padrão é 3.
rand_seed	*Número inteiro*	Valor inicial aleatório utilizado para replicar análises; o padrão é 12345.
min_instances_split	*Número inteiro*	Número mínimo de registros que podem ser divididos, o padrão é 5.
level	*Número inteiro*	Nível de hierarquia no qual os registros devem ser escorados; o padrão é -1.

PCA Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzapcanode.

Tabela 222. Propriedades de netezzapcanode		
Propriedades netezzapcanode	**Valores**	**Descrição da propriedade**
center_data	*sinalização*	Se true (padrão), executa a centralização de dados (também conhecida como "subtração média") antes da análise.

Tabela 222. Propriedades de netezzapcanode (continuação)		
Propriedades netezzapcanode	**Valores**	**Descrição da propriedade**
perform_data_scaling	*sinalização*	Se true, executa ajuste de escala de dados antes da análise. Fazer isso pode tornar a análise menos arbitrária quando variáveis diferentes forem medidas em unidades diferentes.
force_eigensolve	*sinalização*	Se true, utiliza um método menos preciso, porém mais rápido de localizar componentes principais.
pc_number	*Número inteiro*	Número de componentes principais para o qual o conjunto de dados deve ser reduzido; o padrão é 1.

Árvore de regressão Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzaregtreenode.

Tabela 223. Propriedades de netezzaregtreenode		
Propriedades netezzaregtreenode	**Valores**	**Descrição da propriedade**
max_tree_depth	*Número inteiro*	Número máximo de níveis até o qual a árvore pode crescer abaixo do nó raiz; o padrão é 10.
split_evaluation_measure	Variance	Medida de impureza de classe utilizada para avaliar o melhor local para dividir a árvore; o padrão (e atualmente a única opção) é Variance.
min_improvement_splits	*number*	Quantia mínima para reduzir a impureza antes que uma nova divisão seja criada na árvore.
min_instances_split	*Número inteiro*	Número mínimo de registros que podem ser divididos.
pruning_measure	mse r2 pearson spearman	Método a ser utilizado para limpeza.

Tabela 223. Propriedades de netezzaregtreenode (continuação)		
Propriedades netezzaregtreenode	**Valores**	**Descrição da propriedade**
prune_tree_options	allTrainingData partitionTrainingData useOtherTable	O padrão é utilizar allTrainingData para estimar a precisão do modelo. Utilize partitionTrainingData para especificar uma porcentagem de dados de treinamento a serem utilizados ou useOtherTable para utilizar um conjunto de dados de treinamento de uma tabela de banco de dados especificada.
perc_training_data	*number*	Se prune_tree_options for configurado para PercTrainingData, especifica a porcentagem de dados a serem utilizados no treinamento.
prune_seed	*Número inteiro*	Valor inicial aleatório a ser utilizado para replicar os resultados da análise quando prune_tree_options for configurado para PercTrainingData; o padrão é 1.
pruning_table	*sequência*	Nome da tabela de um conjunto de remoção separado para estimar a precisão do modelo.
compute_probabilities	*sinalização*	Se true, especifica que as variações de classes designadas devem ser incluídas na saída.

Regressão linear Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzalineregressionnode.

Tabela 224. Propriedades de netezzalineregressionnode		
Propriedades netezzalineregressionnode	**Valores**	**Descrição da propriedade**
use_svd	*sinalização*	Se true, utiliza a matriz de Decomposição em Valores Singulares ao invés da matriz original para maior velocidade e precisão numérica.
include_intercept	*sinalização*	Se true (padrão), aumenta a precisão geral da solução.
calculate_model_diagnostics	*sinalização*	Se true, calcula os diagnósticos no modelo.

Séries temporais Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzatimeseriesnode.

Tabela 225. Propriedades de netezzatimeseriesnode

Propriedades netezzatimeseriesnode	**Valores**	**Descrição da propriedade**
time_points	*campo*	Campo de entrada que contém os valores de data ou hora para as séries temporais.
time_series_ids	*campo*	Campo de entrada que contém IDs de séries temporais; usado se a entrada contiver mais de uma série temporal.
model_table	*campo*	Nome da tabela de banco de dados na qual o modelo de série temporal Netezza será armazenado.
description_table	*campo*	Nome da tabela de entrada que contém nomes e descrições de séries temporais.
seasonal_adjustment_table	*campo*	Nome da tabela de saída na qual os valores ajustados sazonalmente calculados pelos algoritmos de suavização exponencial ou de decomposição de tendência sazonal serão armazenados.
algorithm_name	SpectralAnalysis ou spectral ExponentialSmoothing ou esmoothing ARIMA SeasonalTrendDecomposition ou std	Algoritmo a ser utilizado para modelagem de séries temporais.

Tabela 225. Propriedades de netezzatimeseriesnode (continuação)

Propriedades netezzatimeseriesnode	**Valores**	**Descrição da propriedade**
trend_name	N A DA M DM	Tipo de tendência para suavização exponencial: N - nenhum A - aditiva DA – aditiva amortecida M - multiplicativa DM - multiplicativa amortecida
seasonality_type	N A M	Tipo de sazonalidade para suavização exponencial: N - nenhum A - aditiva M - multiplicativa
interpolation_method	linear cubicspline exponentialspline	Método de interpolação a ser utilizado.
timerange_setting	SD SP	Configurando para o intervalo de tempo a ser utilizado: SD - determinado pelo sistema (utiliza o intervalo inteiro de dados de série temporal) SP – especificado pelo usuário por meio do earliest_time e latest_time

Tabela 225. Propriedades de netezzatimeseriesnode (continuação)

Propriedades netezzatimeseriesnode	**Valores**	**Descrição da propriedade**

`earliest_time`	*Número inteiro*
	
	Data
	
	time
	
	registro de data e hora
		Valores de início e de término, se `timerange_setting` for SP.
		
		O formato deve seguir o valor `time_points`.
		
		Por exemplo, se o campo `time_points` contiver uma data, isso também deverá ser uma data.
		
		Exemplo:
		
		`set NZ_DT1.timerange_setting = 'SP'`
		
		`set NZ_DT1.earliest_time = '1921-01-01'`
		
		`set NZ_DT1.latest_time = '2121-01-01'`

`latest_time`		

Tabela 225. Propriedades de netezzatimeseriesnode (continuação)

Propriedades netezzatimeseriesnode	Valores	Descrição da propriedade
arima_setting	SD SP	Configuração para o algoritmo ARIMA (usado apenas se algorithm_name for configurado para ARIMA): SD - determinado pelo sistema SP - especificado pelo usuário Se arima_setting = SP, use os parâmetros a seguir para configurar os valores sazonais e não sazonais Exemplo (apenas não sazonal): set NZ_DT1.algorithm_name = 'arima' set NZ_DT1.arima_setting = 'SP' set NZ_DT1.p_symbol = 'lesseq' set NZ_DT1.p = '4' set NZ_DT1.d_symbol = 'lesseq' set NZ_DT1.d = '2' set NZ_DT1.q_symbol = 'lesseq' set NZ_DT1.q = '4'
p_symbol d_symbol q_symbol sp_symbol sd_symbol sq_symbol	less eq lesseq	ARIMA – operador para os parâmetros p, d, q, sp, sd e sq: less - menor que eq - igual lesseq - menor ou igual a
p	*Número inteiro*	ARIMA - graus não sazonais de autocorrelação.

Tabela 225. Propriedades de netezzatimeseriesnode (continuação)		
Propriedades netezzatimeseriesnode	**Valores**	**Descrição da propriedade**
q	*Número inteiro*	ARIMA - valor de derivação não sazonal.
d	*Número inteiro*	ARIMA - número não sazonal de pedidos médios móveis no modelo.
sp	*Número inteiro*	ARIMA - graus sazonais de autocorrelação.
sq	*Número inteiro*	ARIMA - valor de derivação sazonal.
sd	*Número inteiro*	ARIMA - número sazonal de pedidos médios móveis no modelo.
advanced_setting	SD SP	Determina como as configurações avançadas devem ser manipuladas: SD - determinado pelo sistema SP - especificado pelo usuário por meio de period , units_period e forecast_setting. Exemplo: set NZ_DT1.advanced_setting = 'SP' set NZ_DT1.period = 5 set NZ_DT1.units_period = 'd'
period	*Número inteiro*	Comprimento do ciclo sazonal, especificado em conjunto com units_period. Não aplicável para análise espectral.

Tabela 225. Propriedades de netezzatimeseriesnode (continuação)

Propriedades netezzatimeseriesnode	**Valores**	**Descrição da propriedade**
units_period	ms s min h d wk q y	Unidades na qual period é expresso: ms – milissegundos s – segundos min - minutos h – horas d – dias wk - semanas q – trimestres y – anos Por exemplo, para uma série temporal semanal, utilize 1 para period e wk para units_period.
forecast_setting	forecasthorizon forecasttimes	Especifica como as previsões devem ser feitas.
forecast_horizon	*Número inteiro* *Data* *time* *registro de data e hora*	Se forecast_setting = forecasthorizon, especifica o valor do ponto de extremidade para previsão. O formato deve seguir o valor time_points. Por exemplo, se o campo time_points contiver uma data, isso também deverá ser uma data.

Tabela 225. Propriedades de netezzatimeseriesnode (continuação)		
Propriedades netezzatimeseriesnode	**Valores**	**Descrição da propriedade**
forecast_times	*Número inteiro* *Data* *time* *registro de data e hora*	Se forecast_setting = forecasttimes, especifica os valores a serem utilizados para fazer previsões. O formato deve seguir o valor time_points. Por exemplo, se o campo time_points contiver uma data, isso também deverá ser uma data.
include_history	*sinalização*	Identifica se valores históricos devem ser incluídos na saída.
include_interpolated_values	*sinalização*	Indica se valores interpolados devem ser incluídos na saída. Não aplicável se include_history for false.

Modelo linear generalizado Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzaglmnode.

Tabela 226. Propriedades de netezzaglmnode		
Propriedades netezzaglmnode	**Valores**	**Descrição da propriedade**
dist_family	bernoulli gaussian poisson negativebinomial wald gamma	O tipo de distribuição; o padrão é bernoulli
dist_params	*number*	Valor de parâmetro de distribuição a ser utilizado. Aplicável apenas se distribution for Negativebinomial.

Tabela 226. Propriedades de netezzaglmnode (continuação)

Propriedades netezzaglmnode	Valores	Descrição da propriedade
trials	*Número inteiro*	Aplicável apenas se distribution for Binomial. Quando a resposta de destino for um número de eventos que ocorrem em um conjunto de avaliações, o campo target conterá o número de eventos e o campo trials conterá o número de avaliações.
model_table	*campo*	Nome da tabela de banco de dados na qual o modelo linear generalizado Netezza será armazenado.
maxit	*Número inteiro*	Número máximo de iterações que o algoritmo deve executar; o padrão é 20.
eps	*number*	Valor máximo de erro (em notação científica) no qual o algoritmo deve parar ao localizar o melhor modelo de ajuste. O padrão é -3, significando 1E-3, ou 0,001.
tol	*number*	Valor (em notação científica) abaixo do qual os erros são tratados como tendo um valor de zero. O padrão é -7, o que significa que valores de erro abaixo de 1E-7 (ou 0,0000001) são contados como insignificantes.

Tabela 226. Propriedades de netezzaglmnode (continuação)

Propriedades netezzaglmnode	Valores	Descrição da propriedade
link_func	identity inverse invnegative invsquare sqrt power oddspower log clog loglog cloglog logit probit gaussit cauchit canbinom cangeom cannegbinom	A função Link a ser utilizada; o padrão é logit.
link_params	*number*	Valor do parâmetro da função de ligação a ser utilizado. Aplicável apenas se link_function for power ou oddspower.

Tabela 226. Propriedades de netezzaglmnode (continuação)		
Propriedades netezzaglmnode	**Valores**	**Descrição da propriedade**
interaction	[[[*colnames1*],[*levels1*]], [[*colnames2*],[*levels2*]], ...,[[*colnamesN*],[*levelsN*]],]	Especifica as interações entre os campos. *colnames* é uma lista de campos de entrada e *level* é sempre 0 para cada campo. Exemplo: `[[["K","BP","Sex","K"], [0,0,0,0]], [["Age","Na"],[0,0]]]`
intercept	*sinalização*	Se true, inclui a interceptação no modelo.

Propriedades de Nugget do Modelo Netezza

As propriedades a seguir são comuns para nuggets do modelo de banco de dados Netezza.

Tabela 227. Propriedades comuns do nugget do modelo Netezza		
Propriedades Comuns do Nugget do Modelo Netezza	**Valores**	**Descrição da propriedade**
connection	*sequência*	A sequência de conexões com o banco de dados Netezza no qual o modelo é armazenado.
table_name	*sequência*	Nome da tabela de banco de dados na qual o modelo é armazenado.

Outras propriedades de nugget do modelo são as mesmas que para o nó de modelagem correspondente.

Os nomes de script dos nuggets do modelo são os seguintes.

Tabela 228. Nomes de script de nuggets do modelo Netezza	
Nugget do Modelo	**Nome de Script**
Árvore de decisão	applynetezzadectreenode
K-Médias	applynetezzakmeansnode
Rede Bayes	applynetezzabayesnode
Naive Bayes	applynetezzanaivebayesnode
KNN	applynetezzaknnnode
Armazenamento em Cluster Decisivo	applynetezzadivclusternode
PCA	applynetezzapcanode
Árvore de Regressão	applynetezzaregtreenode
Regressão linear	applynetezzalineregressionnode
Séries temporais	applynetezzatimeseriesnode
Linear generalizado	applynetezzaglmnode

Capítulo 16. Propriedades do nó de saída

As propriedades do nó de saída diferem um pouco das propriedades de outros tipos de nós. Ao invés de referenciar uma opção de nó específica, as propriedades do nó de saída armazenam uma referência para o objeto de saída. Isso é útil ao selecionar um valor de uma tabela e, em seguida, configurá-lo como um parâmetro de fluxo.

Esta seção descreve as propriedades de script disponíveis para nós de saída.

Propriedades de analysisnode

O nó Análise avalia a capacidade de modelos preditivos de gerar predições exatas. Os nós de análise executam várias comparações entre os valores preditos e os valores reais para um ou mais nuggets do modelo. Eles também podem comparar modelos preditivos uns aos outros.

Exemplo

```
node = stream.create("analysis", "My node")
# "Analysis" tab
node.setPropertyValue("coincidence", True)
node.setPropertyValue("performance", True)
node.setPropertyValue("confidence", True)
node.setPropertyValue("threshold", 75)
node.setPropertyValue("improve_accuracy", 3)
node.setPropertyValue("inc_user_measure", True)
# "Define User Measure..."
node.setPropertyValue("user_if", "@TARGET = @PREDICTED")
node.setPropertyValue("user_then", "101")
node.setPropertyValue("user_else", "1")
node.setPropertyValue("user_compute", ["Mean", "Sum"])
node.setPropertyValue("by_fields", ["Drug"])
# "Output" tab
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/analysis_out.html")
```

Tabela 229. Propriedades de analysisnode

propriedades analysisnode	**Tipo de dados**	**Descrição da propriedade**
output_mode	Screen File	Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser usado.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída.
by_fields	*lista*	

Tabela 229. Propriedades de analysisnode (continuação)		
propriedades analysisnode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	Se for saída de disco, de dados ou HTML, o nome do arquivo de saída.
coincidence	*sinalização*	
performance	*sinalização*	
evaluation_binary	*sinalização*	
confidence	*sinalização*	
threshold	*number*	
improve_accuracy	*number*	
field_detection_method	Metadata Name	Determina como campos preditos são correspondidos com o campo de destino original. Especifique Metadata ou Name.
inc_user_measure	*sinalização*	
user_if	*expr*	
user_then	*expr*	
user_else	*expr*	
user_compute	[Mean Sum Min Max SDev]	

Propriedades de dataauditnode

O nó Auditoria de Dados fornece uma primeira visão abrangente dos dados, incluindo estatísticas de resumo, histogramas e distribuição para cada campo, bem como informações sobre valores discrepantes, valores omissos e valores extremos. Os resultados são exibidos em uma matriz de fácil leitura que pode ser classificada e utilizada para gerar gráficos de tamanho completo e nós de preparação de dados.

Exemplo

```
filenode = stream.createAt("variablefile", "File", 100, 100)
filenode.setPropertyValue("full_filename", "$CLEO_DEMOS/DRUG1n")
node = stream.createAt("dataaudit", "My node", 196, 100)
stream.link(filenode, node)
node.setPropertyValue("custom_fields", True)
node.setPropertyValue("fields", ["Age", "Na", "K"])
node.setPropertyValue("display_graphs", True)
node.setPropertyValue("basic_stats", True)
node.setPropertyValue("advanced_stats", True)
node.setPropertyValue("median_stats", False)
node.setPropertyValue("calculate", ["Count", "Breakdown"])
node.setPropertyValue("outlier_detection_method", "std")
node.setPropertyValue("outlier_detection_std_outlier", 1.0)
node.setPropertyValue("outlier_detection_std_extreme", 3.0)
node.setPropertyValue("output_mode", "Screen")
```

Tabela 230. Propriedades de dataauditnode		
propriedades dataauditnode	**Tipo de dados**	**Descrição da propriedade**
custom_fields	*sinalização*	
fields	*[field1 ... fieldN]*	
overlay	*campo*	
display_graphs	*flag*	Utilizado para desativar a exibição de gráficos na matriz de saída ativada ou desativada.
basic_stats	*sinalização*	
advanced_stats	*sinalização*	
median_stats	*sinalização*	
calculate	Count Breakdown	Utilizado para calcular valores ausentes. Selecione para usar um dos, ambos ou nenhum método de cálculo.
outlier_detection_method	std iqr	Usado para especificar o método de detecção para valores discrepantes e valores extremos.
outlier_detection_std_outlier	*number*	Se outlier_detection_method for std, especifica o número a ser usado para definir valores discrepantes.
outlier_detection_std_extreme	*number*	Se outlier_detection_method for std, especifica o número a ser usado para definir valores extremos.
outlier_detection_iqr_outlier	*number*	Se outlier_detection_method for iqr, especifica o número a ser usado para definir valores discrepantes.
outlier_detection_iqr_extreme	*number*	Se outlier_detection_method for iqr, especifica o número a ser usado para definir valores extremos.
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser usado.
output_mode	Screen File	Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.

Tabela 230. Propriedades de dataauditnode (continuação)		
propriedades dataauditnode	**Tipo de dados**	**Descrição da propriedade**
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída.
paginate_output	*sinalização*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*number*	Quando usado com paginate_output, especifica as linhas por página de saída.
full_filename	*sequência*	

propriedades extensionoutputnode

O nó de Saída de Extensão permite analisar dados e os resultados de escoragem de modelo usando seu próprio script customizado R ou Python para Spark A saída da análise pode ser texto ou gráfico. A saída é incluída na guia **Saída** da área de janela do gerenciador; como alternativa, a saída pode ser redirecionada para um arquivo.

Exemplo de Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_output", "extension_output")
node.setPropertyValue("syntax_type", "Python")

python_script = """
import json
import spss.pyspark.runtime

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
schema = df.dtypes[:]
print df
"""

node.setPropertyValue("python_syntax", python_script)
```

Exemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", "print(modelerData$Age)")
```

Tabela 231. propriedades extensionoutputnode		
propriedades extensionoutputnode	**Tipo de dados**	**Descrição da propriedade**
syntax_type	*R* *Python*	Especifique qual script é executado - R ou Python (R é o padrão).
r_syntax	*sequência*	Sintaxe do script R para escoragem de modelo.
python_syntax	*sequência*	Sintaxe de script Python para pontuação do modelo.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opção para converter os campos de sinalização.
convert_missing	*sinalização*	Opção para converter valores ausentes para o valor NA do R.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.
output_to	`Screen` `File`	Especifique o tipo de saída (`Screen` ou `File`).
output_type	`Graph` `Text`	Especifique se deve produzir saída gráfica ou de texto.
full_filename	*sequência*	Nome do arquivo a ser usado para a saída gerada.
graph_file_type	`HTML` `COU`	Tipo de arquivo para o arquivo de saída (`.html` ou `.cou`).

text_file_type	`HTML`
	`TEXT`
	`COU`
		Especifique o tipo de arquivo para saída de texto (`.html`, `.txt` ou `.cou`).

propriedades kdeexport

Estimativa De Densidade Do Kernel (KDE) © utiliza os algoritmos Ball Tree ou KD Tree para consultas eficientes, e combina conceitos a partir de aprendizado não supervisionado, engenharia de recursos e modelagem de dados. Abordagens baseadas em vizinhos, como o KDE, são algumas das técnicas de estimativa de densidade mais populares e úteis. Os Nós KDE Modelagem e Simulação KDE em SPSS Modeler expõem os principais recursos e parâmetros comumente usados da biblioteca KDE. Os nós são implementados em Python.

Tabela 232. propriedades kdeexport		
propriedades kdeexport	**Tipo de dados**	**Descrição da propriedade**
bandwidth	*double*	O padrão é 1.
kernel	*sequência*	O kernel a ser usado: gaussian ou tophat. O padrão é gaussian.
algorithm	*sequência*	O algoritmo da árvore a ser usado: kd_tree, ball_tree ou auto. O padrão é auto.
metric	*sequência*	A métrica a ser usada ao calcular distância. Para o algoritmo kd_tree, escolha entre: Euclidean, Chebyshev, Cityblock, Minkowski, Manhattan, Infinity, P, L2 ou L1. Para o algoritmo ball_tree, escolha entre: Euclidian, Braycurtis, Chebyshev, Canberra, Cityblock, Dice, Hamming, Infinity, Jaccard, L1, L2, Minkowski, Matching, Manhattan, P, Rogersanimoto, Russellrao, Sokalmichener, Sokalsneath ou Kulsinski. O padrão é Euclidean.
atol	*Valor flutuante*	A tolerância absoluta desejada do resultado. Uma tolerância maior geralmente levará a uma execução mais rápida. O padrão é 0.0.
rtol	*Valor flutuante*	A tolerância relativa desejada do resultado. Uma tolerância maior geralmente levará a uma execução mais rápida. O padrão é 1E-8.
breadthFirst	*Booleano*	Configure como True para usar uma abordagem de amplitude primeiro. Configure como False para usar uma abordagem de profundidade primeiro. O padrão é True.
LeafSize	*Número inteiro*	O tamanho de folha da árvore subjacente. O padrão é 40. A mudança deste valor pode impactar significativamente o desempenho.
pValue	*double*	Especifique o Valor P para usar se você estiver usando Minkowski para a métrica. O patrão é 1.5.

Propriedades matrixnode

O nó Matriz cria uma tabela que mostra relacionamentos entre campos. Ele é mais comumente usado para mostrar a relação entre dois campos simbólicos, mas também pode mostrar relações entre campos de sinalização ou campos numéricos.

Exemplo

```
node = stream.create("matrix", "My node")
# "Settings" tab
node.setPropertyValue("fields", "Numerics")
node.setPropertyValue("row", "K")
node.setPropertyValue("column", "Na")
node.setPropertyValue("cell_contents", "Function")
node.setPropertyValue("function_field", "Age")
node.setPropertyValue("function", "Sum")
# "Appearance" tab
node.setPropertyValue("sort_mode", "Ascending")
node.setPropertyValue("highlight_top", 1)
node.setPropertyValue("highlight_bottom", 5)
node.setPropertyValue("display", ["Counts", "Expected", "Residuals"])
node.setPropertyValue("include_totals", True)
# "Output" tab
node.setPropertyValue("full_filename", "C:/output/matrix_output.html")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabela 233. Propriedades matrixnode

propriedades matrixnode	**Tipo de dados**	**Descrição da propriedade**
fields	Selected Flags Numerics	
row	*campo*	
column	*campo*	
include_missing_values	*sinalização*	Especifica se os valores ausentes do usuário (em branco) e ausentes do sistema (nulos) são incluídos na saída da linha e da coluna.
cell_contents	CrossTabs Function	
function_field	*sequência*	
function	Sum Mean Min Max SDev	

Tabela 233. Propriedades matrixnode (continuação)

propriedades matrixnode	Tipo de dados	Descrição da propriedade
sort_mode	Unsorted Ascending Descending	
highlight_top	*number*	Se diferente de zero, então true.
highlight_bottom	*number*	Se diferente de zero, então true.
display	[Counts Expected Residuals RowPct ColumnPct TotalPct]	
include_totals	*sinalização*	
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser usado.
output_mode	Screen File	Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída. Ambos os formatos Formatted e Delimited podem levar o modificador transposed, que transpõe as linhas e colunas na tabela.
paginate_output	*sinalização*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*number*	Quando usado com paginate_output, especifica as linhas por página de saída.
full_filename	*sequência*	

Propriedades de meansnode

O nó Média compara a média entre grupos independentes ou entre pares de campos relacionados para testar se há uma diferença significativa. Por exemplo, é possível comparar receitas médias antes e depois de realizar uma promoção ou comparar receitas de clientes que não receberam a promoção com aqueles que receberam.

Exemplo

```
node = stream.create("means", "My node")
node.setPropertyValue("means_mode", "BetweenFields")
node.setPropertyValue("paired_fields", [["OPEN_BAL", "CURR_BAL"]])
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("output_view", "Advanced")
node.setPropertyValue("output_mode", "File")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/output/means_output.html")
```

Tabela 234. Propriedades de meansnode

propriedades meansnode	**Tipo de dados**	**Descrição da propriedade**
means_mode	BetweenGroups BetweenFields	Especifica o tipo de estatística de médias a ser executado nos dados.
test_fields	[field1 ... fieldn]	Especifica o campo de teste quando means_mode é configurado como BetweenGroups.
grouping_field	*campo*	Especifica o campo de agrupamento.
paired_fields	[[field1 field2] [field3 field4] ...]	Especifica os pares de campo a serem usados quando means_mode é configurado como BetweenFields.
label_correlations	*sinalização*	Especifica se os rótulos de correlação são mostrados na saída. Essa configuração se aplica apenas quando means_mode é configurado como BetweenFields.
correlation_mode	Probability Absolute	Especifica se as correlações devem ser rotuladas por probabilidade ou valor absoluto.
weak_label	*sequência*	
medium_label	*sequência*	
strong_label	*sequência*	

Tabela 234. Propriedades de meansnode (continuação)		
propriedades meansnode	**Tipo de dados**	**Descrição da propriedade**
weak_below_probability	*number*	Quando correlation_mode é configurado como Probability, especifica o valor de corte para correlações fracas. Este deve ser um valor entre 0 e 1 – por exemplo, 0,90.
strong_above_probability	*number*	O valor de corte para correlações fortes.
weak_below_absolute	*number*	Quando correlation_mode é configurado como Absolute, especifica o valor de corte para correlações fracas. Este deve ser um valor entre 0 e 1 – por exemplo, 0,90.
strong_above_absolute	*number*	O valor de corte para correlações fortes.
unimportant_label	*sequência*	
marginal_label	*sequência*	
important_label	*sequência*	
unimportant_below	*number*	O valor de corte para importância de campo baixa. Este deve ser um valor entre 0 e 1 – por exemplo, 0,90.
important_above	*number*	
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Nome a ser utilizado.
output_mode	Screen File	Especifica o local de destino para a saída gerada a partir do nó de saída.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Especifica o tipo de saída.
full_filename	*sequência*	
output_view	Simple Advanced	Especifica se a visualização simples ou avançada é exibida na saída.

Propriedades de reportnode

O nó Relatório cria relatórios formatados contendo texto fixo, bem como dados e outras expressões derivadas dos dados. Especifique o formato do relatório utilizando os modelos de texto para definir as construções de saída de texto e de dados fixos. É possível fornecer formatação de texto customizada utilizando tags HTML no modelo e configurando as opções na guia Saída. Você pode incluir valores de dados e outra saída condicional usando expressões CLEM no template.

Exemplo

```
node = stream.create("report", "My node")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("full_filename", "C:/report_output.html")
node.setPropertyValue("lines_per_page", 50)
node.setPropertyValue("title", "Report node created by a script")
node.setPropertyValue("highlights", False)
```

Tabela 235. Propriedades de reportnode

propriedades reportnode	**Tipo de dados**	**Descrição da propriedade**
output_mode	Screen File	Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.
output_format	HTML (*.html*) Text (*.txt*) Output (*.cou*)	Usado para especificar o tipo de saída de arquivo.
format	Auto Custom	Usado para escolher se a saída será formatada automaticamente ou será formatada usando o HTML incluído no modelo. Para usar a formatação HTML no modelo, especifique Custom.
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser usado.
text	*sequência*	
full_filename	*sequência*	
highlights	*sinalização*	
title	*sequência*	
lines_per_page	*number*	

Propriedades de routputnode

O nó Saída R permite analisar dados e os resultados da escoragem de modelo utilizando seu próprio script R customizado. A saída da análise pode ser texto ou gráfico. A saída é incluída na guia **Saída** da área de janela do gerenciador; como alternativa, a saída pode ser redirecionada para um arquivo.

Tabela 236. Propriedades de routputnode

propriedades routputnode	Tipo de dados	Descrição da propriedade
syntax	*sequência*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*sinalização*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*sinalização*	
output_name	Auto Custom	
custom_name	*sequência*	
output_to	Screen File	
output_type	Graph Text	
full_filename	*sequência*	
graph_file_type	HTML COU	
text_file_type	HTML TEXT COU	

Propriedades de setglobalsnode

O nó Configurar Globais varre os dados e calcula os valores de resumo que podem ser utilizados em expressões do CLEM. Por exemplo, você pode usar este nó para computar estatísticas para um campo chamado *age* e, em seguida, usar a média geral de *idade* em CLEM expressões inserindo a função `@GLOBAL_MEAN(age)`.

Exemplo

```
node = stream.create("setglobals", "My node")
node.setKeyedPropertyValue("globals", "Na", ["Max", "Sum", "Mean"])
```

```
node.setKeyedPropertyValue("globals", "K", ["Max", "Sum", "Mean"])
node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
node.setPropertyValue("clear_first", False)
node.setPropertyValue("show_preview", True)
```

Tabela 237. Propriedades de setglobalsnode

propriedades setglobalsnode	**Tipo de dados**	**Descrição da propriedade**
globals	[Sum Mean Min Max SDev]	Propriedade estruturada em que os campos a serem configurados devem ser referenciados com a sintaxe a seguir: node.setKeyedPropertyValue("globals", "Age", ["Max", "Sum", "Mean", "SDev"])
clear_first	*sinalização*	
show_preview	*sinalização*	

Propriedades de simevalnode

O nó Avaliação de Simulação avalia um campo de destino previsto especificado e apresenta informações de distribuição e de correlação sobre o campo de destino.

Tabela 238. Propriedades de simevalnode

propriedades simevalnode	**Tipo de dados**	**Descrição da propriedade**
target	*campo*	
iteration	*campo*	
presorted_by_iteration	*Booleano*	
max_iterations	*number*	
tornado_fields	*[field1...fieldN]*	
plot_pdf	*Booleano*	
plot_cdf	*Booleano*	
show_ref_mean	*Booleano*	
show_ref_median	*Booleano*	
show_ref_sigma	*Booleano*	
num_ref_sigma	*number*	
show_ref_pct	*Booleano*	
ref_pct_bottom	*number*	
ref_pct_top	*number*	
show_ref_custom	*Booleano*	

Tabela 238. Propriedades de simevalnode (continuação)		
propriedades simevalnode	**Tipo de dados**	**Descrição da propriedade**
ref_custom_values	*[number1...numberN]*	

category_values	`Category`
	`Probabilities`
	`Both`	

category_groups	`Categories` `Iterations`	
create_pct_table	*Booleano*	

pct_table	`Quartiles`
	`Intervals`
	`Custom`	

pct_intervals_num	*number*	
pct_custom_values	*[number1...numberN]*	

Propriedades de simfitnode

O nó Ajuste de Simulação examina a distribuição estatística dos dados em cada campo e gera (ou atualiza) um nó Gerar Simulação com a melhor distribuição de ajuste designada a cada campo. Em seguida, o nó Gerar Simulação poderá ser utilizado para gerar dados simulados.

Tabela 239. Propriedades de simfitnode		
propriedades simfitnode	**Tipo de dados**	**Descrição da propriedade**

build	`Node`
	`XMLExport`
	`Both`	

use_source_node_name	*Booleano*	
source_node_name	*sequência*	O nome customizado do nó de origem que está sendo gerado ou atualizado.
use_cases	`All` `LimitFirstN`	
use_case_limit	*Número inteiro*	
fit_criterion	`AndersonDarling` `KolmogorovSmirnov`	
num_bins	*Número inteiro*	
parameter_xml_filename	*sequência*	
generate_parameter_import	*Booleano*	

Propriedades de statisticsnode

O nó Estatísticas fornece informações de resumo básicas sobre campos numéricos. Ela calcula as estatísticas de resumo para campos individuais e correlações entre os campos.

Exemplo

```
node = stream.create("statistics", "My node")
# "Settings" tab
node.setPropertyValue("examine", ["Age", "BP", "Drug"])
node.setPropertyValue("statistics", ["mean", "sum", "sdev"])
node.setPropertyValue("correlate", ["BP", "Drug"])
# "Correlation Labels..." section
node.setPropertyValue("label_correlations", True)
node.setPropertyValue("weak_below_absolute", 0.25)
node.setPropertyValue("weak_label", "lower quartile")
node.setPropertyValue("strong_above_absolute", 0.75)
node.setPropertyValue("medium_label", "middle quartiles")
node.setPropertyValue("strong_label", "upper quartile")
# "Output" tab
node.setPropertyValue("full_filename", "c:/output/statistics_output.html")
node.setPropertyValue("output_format", "HTML")
```

Tabela 240. Propriedades de statisticsnode

propriedades statisticsnode	**Tipo de dados**	**Descrição da propriedade**
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser usado.
output_mode	Screen File	Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída.
full_filename	*sequência*	
examine	*lista*	
correlate	*lista*	
statistics	[count mean sum min max range variance sdev semean median mode]	
correlation_mode	Probability Absolute	Especifica se as correlações devem ser rotuladas por probabilidade ou valor absoluto.

Tabela 240. Propriedades de statisticsnode (continuação)		
propriedades statisticsnode	**Tipo de dados**	**Descrição da propriedade**
label_correlations	*sinalização*	
weak_label	*sequência*	
medium_label	*sequência*	
strong_label	*sequência*	
weak_below_probability	*number*	Quando correlation_mode é configurado como Probability, especifica o valor de corte para correlações fracas. Este deve ser um valor entre 0 e 1 – por exemplo, 0,90.
strong_above_probability	*number*	O valor de corte para correlações fortes.
weak_below_absolute	*number*	Quando correlation_mode é configurado como Absolute, especifica o valor de corte para correlações fracas. Este deve ser um valor entre 0 e 1 – por exemplo, 0,90.
strong_above_absolute	*number*	O valor de corte para correlações fortes.

Propriedades de statisticsoutputnode

O nó Saída de Estatísticas permite chamar um procedimento do IBM SPSS Statistics para analisar seus dados do IBM SPSS Modeler. Uma ampla variedade de procedimentos de analítica do IBM SPSS Statistics está disponível. Este nó requer uma cópia licenciada de IBM SPSS Statistics.

As propriedades desse nó são descritas em "Propriedades de statisticsoutputnode" na página 433.

Propriedades de tablenode

O nó de Tabela exibe os dados no formato de tabela, que também podem ser gravados em um arquivo. Isso é útil sempre que você precisar inspecionar seus valores de dados ou exportá-los em um formato facilmente legível.

Exemplo

```
node = stream.create("table", "My node")
node.setPropertyValue("highlight_expr", "Age > 30")
node.setPropertyValue("output_format", "HTML")
node.setPropertyValue("transpose_data", True)
node.setPropertyValue("full_filename", "C:/output/table_output.htm")
node.setPropertyValue("paginate_output", True)
node.setPropertyValue("lines_per_page", 50)
```

Tabela 241. Propriedades de tablenode

propriedades tablenode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	Se for saída de disco, de dados ou HTML, o nome do arquivo de saída.
use_output_name	*sinalização*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser usado.
output_mode	Screen File	Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.
output_format	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída.
transpose_data	*sinalização*	Transpõe os dados antes de exportar para que as linhas representem campos e as colunas representam registros.
paginate_output	*sinalização*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*number*	Quando usado com paginate_output, especifica as linhas por página de saída.
highlight_expr	*sequência*	
output	*sequência*	Uma propriedade somente leitura que contém uma referência à última tabela construída pelo nó.
value_labels	*[[Value LabelString]* *[Value LabelString] ...]*	Utilizado para especificar rótulos para pares de valores.
display_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exibido (aplica-se apenas aos campos com armazenamento REAL). Um valor de -1 utilizará o padrão de fluxo.
export_places	*Número inteiro*	Configura o número de casas decimais para o campo quando exportado (aplica-se apenas aos campos com armazenamento REAL). Um valor de -1 utilizará o padrão de fluxo.

Tabela 241. Propriedades de tablenode (continuação)

propriedades tablenode	Tipo de dados	Descrição da propriedade
decimal_separator	DEFAULT PERIOD COMMA	Configura o separador decimal para o campo (aplica-se apenas aos campos com armazenamento REAL).
date_format	"DDMMYY" "MMDDYY" "YYMMDD" "YYYYMMDD" "YYYYDDD" DAY MONTH "DD-MM-YY" "DD-MM-YYYY" "MM-DD-YY" "MM-DD-YYYY" "DD-MON-YY" "DD-MON-YYYY" "YYYY-MM-DD" "DD.MM.YY" "DD.MM.YYYY" "MM.DD.YYYY" "DD.MON.YY" "DD.MON.YYYY" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MON YYYY q Q YYYY ww WK YYYY	Configura o formato de data para o campo (aplica-se apenas aos campos com o armazenamento DATE ou TIMESTAMP).

Tabela 241. Propriedades de tablenode (continuação)		
propriedades tablenode	**Tipo de dados**	**Descrição da propriedade**
time_format	"HHMMSS" "HHMM" "MMSS" "HH:MM:SS" "HH:MM" "MM:SS" "(H)H:(M)M:(S)S" "(H)H:(M)M" "(M)M:(S)S" "HH.MM.SS" "HH.MM" "MM.SS" "(H)H.(M)M.(S)S" "(H)H.(M)M" "(M)M.(S)S"	Configura o formato de hora para o campo (aplica-se apenas aos campos com armazenamento TIME ou TIMESTAMP).
column_width	*Número inteiro*	Configura a largura da coluna para o campo. Um valor -1 configurará a largura da coluna para Auto.
justify	AUTO CENTER LEFT RIGHT	Configura a justificação da coluna para o campo.

Propriedades transformnode

O Transforma nó permite selecionar e visualizar visualmente os resultados de transformações antes de aplicá-los em campos selecionados.

Exemplo

```
node = stream.create("transform", "My node")
node.setPropertyValue("fields", ["AGE", "INCOME"])
node.setPropertyValue("formula", "Select")
node.setPropertyValue("formula_log_n", True)
node.setPropertyValue("formula_log_n_offset", 1)
```

Tabela 242. Propriedades transformnode

propriedades `transformnode`	**Tipo de dados**	**Descrição da propriedade**
`fields`	[*field1... fieldn*]	Os campos a serem utilizados na transformação.

`formula`	`All`
	
	`Select`
		Indica se todas ou apenas as transformações selecionadas devem ser calculadas.

`formula_inverse`	*flag*	Indica se a transformação inversa deve ser utilizada.
`formula_inverse_offset`	*número*	Indica um deslocamento de dados a ser utilizado para a fórmula. Configure como 0 por padrão, a menos que seja especificado pelo usuário.
`formula_log_n`	*flag*	Indica se a transformação $\log_n$ deve ser utilizada.
`formula_log_n_offset`	*número*	
`formula_log_10`	*flag*	Indica se a transformação $\log_{10}$ deve ser utilizada.
`formula_log_10_offset`	*número*	
`formula_exponential`	*flag*	Indica se a transformação exponencial (e^x) deve ser utilizada.
`formula_square_root`	*flag*	Indica se a transformação raiz quadrada deve ser utilizada.
`use_output_name`	*flag*	Especifica se um nome de saída customizado é usado.
`output_name`	*string*	Se use_output_name for true, especifica o nome a ser utilizado.

`output_mode`	`Screen`
	
	`File`
		Utilizado para especificar o local de destino da saída gerada a partir do nó de saída.

`output_format`	`HTML` (*.html*) `Output` (*.cou*)	Utilizado para especificar o tipo de saída.
`paginate_output`	*sinalização*	Quando o `output_format` é `HTML`, faz com que a saída seja separada em páginas.

Tabela 242. Propriedades transformnode (continuação)

propriedades `transformnode`	**Tipo de dados**	**Descrição da propriedade**
`lines_per_page`	*number*	Quando usado com `paginate_output`, especifica as linhas por página de saída.
`full_filename`	*string*	Indica o nome do arquivo a ser utilizado para a saída do arquivo.

Capítulo 17. Propriedades do Nó de Exportação

Propriedades Comuns do Nó Exportação

As propriedades a seguir são comuns a todos os nós de exportação.

Tabela 243. Propriedades comuns do nó de exportação

Propriedade	Valores	Descrição da propriedade
publish_path	*sequência*	Insere o nome raiz a ser utilizado para os arquivos de imagem e de parâmetro publicados.
publish_metadata	*sinalização*	Especifica se um arquivo de metadados é produzido, descrevendo as entradas e saídas da imagem e seus modelos de dados.
publish_use_parameters	*sinalização*	Especifica se os parâmetros de fluxo são incluídos no arquivo *.par.
publish_parameters	*lista de sequências*	Especifica os parâmetros a serem incluídos.
execute_mode	export_data publish	Especifica se o nó é executado sem publicar o fluxo ou se o fluxo é publicado automaticamente quando o nó é executado.

Propriedades de asexport

A exportação do Servidor Analítico permite executar um fluxo no Hadoop Distributed File System (HDFS).

Exemplo

```
node.setPropertyValue("use_default_as", False)
node.setPropertyValue("connection",
["false","9.119.141.141","9080","analyticserver","ibm","admin","admin","false
","","","","",""])
```

Tabela 244. Propriedades de asexport

propriedades asexport	Tipo de dados	Descrição da propriedade
data_source	*sequência*	O nome da origem de dados.
export_mode	*sequência*	Especifica anexar (**append**) dados exportados à origem de dados existente ou gravar (**overwrite**) a origem de dados existente.

Tabela 244. Propriedades de asexport (continuação)		
propriedades asexport	**Tipo de dados**	**Descrição da propriedade**
use_default_as	*Booleano*	Se configurado como True, usa a conexão Servidor Analítico padrão configurada no arquivo options.cfg do servidor. Se configurado como False, usa a conexão desse nó.
connection	["string","string","string", "string","string","string","string", "string" ,"string","string", "string" ,"string"]	Uma propriedade de lista contendo os detalhes da conexão do Servidor Analítico O formato é: ["is_secure_connect", "server_url", "server_port", "context_root", "consumer", "user_name", "password", "use-kerberos-auth", "kerberos-krb5-config-file-path", "kerberos-jaas-config-file-path", "kerberos-krb5-service-principal-name", "enable-kerberos-debug"] Em que: is_secure_connect: indica se a conexão segura é usada e é true ou false. use-kerberos-auth: indica se a autenticação do Kerberos é usada e é true ou false. enable-kerberos-debug: indica se o modo de depuração da autenticação do Kerberos é usado e se é true ou false

Propriedades de cognosexportnode

O nó do IBM Cognos Export exporta dados em um formato que pode ser lido por bancos de dados do Cognos.

Para este nó, deve-se definir uma conexão Cognos e uma conexão ODBC.

Conexão do Cognos

As propriedades para a conexão Cognos são as seguintes.

Tabela 245. Propriedades de cognosexportnode		
propriedades cognosexportnode	**Tipo de dados**	**Descrição da propriedade**
`cognos_connection`	*["string","flag","string","string","string"]*	Uma propriedade de lista que contém os detalhes de conexão com o servidor Cognos. O formato é: `["Cognos_server_URL", login_mode, "namespace", "username", "password"]` em que: `Cognos_server_URL` é a URL do servidor Cognos que contém a origem. `login_mode` indica se login anônimo é usado e é `true` ou `false`; se configurado para `true`, os campos a seguir deverão ser configurados para `""`. `namespace` especifica o provedor de autenticação de segurança utilizado para efetuar logon no servidor. `username` e `password` são aqueles utilizados para efetuar logon no servidor Cognos. Ao invés de `login_mode`, os modos a seguir também estão disponíveis: • `anonymousMode`. Por exemplo: `['Cognos_server_url', 'anonymousMode', "namespace", "username", "password"]` • `credentialMode`. Por exemplo: `['Cognos_server_url', 'credentialMode', "namespace", "username", "password"]`

Tabela 245. Propriedades de cognosexportnode (continuação)		
propriedades cognosexportnode	**Tipo de dados**	**Descrição da propriedade**
		• `storedCredentialMode`. Por exemplo: `['Cognos_server_url', 'storedCredentialMode', "stored_credential_name"]` Em que `stored_credential_name` é o nome de uma credencial do Cognos no repositório.
`cognos_package_name`	*sequência*	O caminho e o nome do pacote do Cognos para o qual você está exportando dados, por exemplo: `/Public Folders/MyPackage`
`cognos_datasource`	*sequência*	

`cognos_export_mode`	`Publish`
	
	`ExportFile`	

`cognos_filename`	*sequência*	

Conexão ODBC

As propriedades para a conexão ODBC são idênticas àquelas listadas para `databaseexportnode` na próxima seção, com a exceção de que a propriedade `datasource` não é válida.

Propriedades de databaseexportnode

O nó Exportação de Banco de Dados grava dados em uma origem de dados relacionais compatível com ODBC. Para escrever para uma fonte de dados ODBC , a fonte de dados deve existir e você deve ter permissão de gravação para ele.

Exemplo

```
'''
Assumes a datasource named "MyDatasource" has been configured
'''
stream = modeler.script.stream()
db_exportnode = stream.createAt("databaseexport", "DB Export", 200, 200)
applynn = stream.findByType("applyneuralnetwork", None)
stream.link(applynn, db_exportnode)

# Export tab
db_exportnode.setPropertyValue("username", "user")
db_exportnode.setPropertyValue("datasource", "MyDatasource")
db_exportnode.setPropertyValue("password", "password")
db_exportnode.setPropertyValue("table_name", "predictions")
db_exportnode.setPropertyValue("write_mode", "Create")
db_exportnode.setPropertyValue("generate_import", True)
```

```
db_exportnode.setPropertyValue("drop_existing_table", True)
db_exportnode.setPropertyValue("delete_existing_rows", True)
db_exportnode.setPropertyValue("default_string_size", 32)

# Schema dialog
db_exportnode.setKeyedPropertyValue("type", "region", "VARCHAR(10)")
db_exportnode.setKeyedPropertyValue("export_db_primarykey", "id", True)
db_exportnode.setPropertyValue("use_custom_create_table_command", True)
db_exportnode.setPropertyValue("custom_create_table_command", "My SQL Code")

# Indexes dialog
db_exportnode.setPropertyValue("use_custom_create_index_command", True)
db_exportnode.setPropertyValue("custom_create_index_command", "CREATE BITMAP
INDEX <index-name>
 ON <table-name> <(index-columns)>")
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields", ["id",
"region"]])
```

Tabela 246. Propriedades de databaseexportnode

propriedades databaseexportnode	**Tipo de dados**	**Descrição da propriedade**
datasource	*sequência*	
username	*sequência*	
password	*sequência*	
epassword	*sequência*	Este slot é somente leitura durante a execução. Para gerar uma senha codificada, utilize a Ferramenta de Senha disponível no menu Ferramentas. Consulte o tópico "Gerando uma Senha Codificada" na página 54 para obter informações adicionais.
table_name	*sequência*	
write_mode	Create Append Merge	
map	*sequência*	Mapeia um nome de campo de fluxo para um nome de coluna de banco de dados (válido apenas se write_mode for Merge). Para obter uma mesclagem, todos os campos devem ser mapeados para serem exportados. Os nomes de campos que não existem no banco de dados são incluídos como novas colunas.

Tabela 246. Propriedades de databaseexportnode (continuação)

propriedades databaseexportnode	**Tipo de dados**	**Descrição da propriedade**
key_fields	*lista*	Especifica o campo de fluxo que é utilizado para a chave; a propriedade map mostra a que isso corresponde no banco de dados.
join	Database Add	
drop_existing_table	*sinalização*	
delete_existing_rows	*sinalização*	
default_string_size	*Número inteiro*	
type		Propriedade estruturada utilizada para configurar o tipo de esquema.
generate_import	*sinalização*	
use_custom_create_table_command	*sinalização*	Utilize o slot *custom_create_table* para modificar o comando SQL CREATE TABLE padrão.
custom_create_table_command	*sequência*	Especifica um comando de sequência a ser utilizado ao invés do comando SQL CREATE TABLE padrão.
use_batch	*sinalização*	As propriedades a seguir são opções avançadas para o carregamento em massa do banco de dados. Um valor True para Use_batch desativa as confirmações linha por linha no banco de dados.
batch_size	*number*	Especifica o número de registros a serem enviados para o banco de dados antes de confirmar na memória.
bulk_loading	Off ODBC External	Especifica o tipo de carregamento em massa. Opções adicionais para ODBC e External são listadas abaixo.
not_logged	*sinalização*	
odbc_binding	Row Column	Especifica a ligação de linha ou a ligação de coluna para carregamento em massa por meio do ODBC.

Tabela 246. Propriedades de databaseexportnode (continuação)		
propriedades databaseexportnode	**Tipo de dados**	**Descrição da propriedade**
loader_delimit_mode	Tab Space Other	Para carregamento em massa por meio de um programa externo, especifique tipo de delimitador. Selecione Other em conjunto com a propriedade loader_other_delimiter para especificar delimitadores, como a vírgula (,).
loader_other_delimiter	*sequência*	
specify_data_file	*sinalização*	Um sinalizador True ativa a propriedade data_file abaixo, em que é possível especificar o nome e o caminho do arquivo no qual gravar durante o carregamento em massa no banco de dados.
data_file	*sequência*	
specify_loader_program	*sinalização*	Um sinalizador True ativa a propriedade loader_program abaixo, em que é possível especificar o nome e o local de um script ou programa carregador externo.
loader_program	*sequência*	
gen_logfile	*sinalização*	Um sinalizador True ativa o logfile_name abaixo, em que é possível especificar o nome de um arquivo no servidor para gerar um log de erros.
logfile_name	*sequência*	
check_table_size	*sinalização*	Um flag verdadeiro permite verificação de tabela para assegurar que o aumento do tamanho da tabela de banco de dados corresponda ao número de linhas exportadas a partir do IBM SPSS Modeler.
loader_options	*sequência*	Especifica argumentos adicionais, como -comment e -specialdir, para o programa carregador.
export_db_primarykey	*sinalização*	Especifica se um determinado campo é uma chave primária.

Tabela 246. Propriedades de databaseexportnode (continuação)		
propriedades databaseexportnode	**Tipo de dados**	**Descrição da propriedade**
use_custom_create_index_command	*sinalização*	Se true, ativa uma consulta SQL customizada para todos os índices.
custom_create_index_command	*sequência*	Especifica o comando SQL utilizado para criar índices quando uma consulta SQL customizada é ativada. (Esse valor pode ser substituído para índices específicos conforme indicado abaixo).
indexes.INDEXNAME.fields		Cria o índice especificado, se necessário, e lista nomes de campos a serem incluídos nesse índice.
INDEXNAME "use_custom_create_index_command"	*sinalização*	Utilizado para ativar ou desativar uma consulta SQL customizada para um índice específico. Veja os exemplos após a tabela a seguir.
INDEXNAME "custom_create_index_command"	*sequência*	Especifica a consulta SQL customizada utilizada para o índice especificado. Veja os exemplos após a tabela a seguir.
indexes.INDEXNAME.remove	*sinalização*	Se True, remove o índice especificado do conjunto de índices.
table_space	*sequência*	Especifica o espaço de tabelas que será criado.
use_partition	*sinalização*	Especifica que o campo hash de distribuição será utilizado.
partition_field	*sequência*	Especifica o conteúdo do campo hash de distribuição.

Nota: Para alguns bancos de dados, é possível especificar que as tabelas de banco de dados sejam criadas para exportação com compactação (por exemplo, o equivalente de `CREATE TABLE MYTABLE (...) COMPRESS YES;` em SQL). As propriedades `use_compression` e `compression_mode` são fornecidas para suportar esse recurso, conforme a seguir.

Tabela 247. Propriedades de databaseexportnode usando recursos de compactação		
propriedades databaseexportnode	**Tipo de dados**	**Descrição da propriedade**
use_compression	*Booleano*	Se configurado para True, cria tabelas para exportação com compactação.

Tabela 247. Propriedades de databaseexportnode usando recursos de compactação (continuação)		
propriedades databaseexportnode	**Tipo de dados**	**Descrição da propriedade**
compression_mode	Row Page	Configura o nível de compactação para bancos de dados SQL Server.
	Default Direct_Load_Operations All_Operations Basic OLTP Query_High Query_Low Archive_High Archive_Low	Define o nível de compactação para bancos de dados Oracle. Observe que os valores OLTP, Query_High, Query_Low, Archive_High e Archive_Low requerem no mínimo o Oracle 11gR2.

Exemplo mostrando como alterar o comando CREATE INDEX para um índice específico:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["use_custom_create_index_command",
 True])db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX",
["custom_create_index_command",
 "CREATE BITMAP INDEX <index-name> ON <table-name> <(index-columns)>"])
```

Como alternativamente, isso pode ser feito através de uma hashtable:

```
db_exportnode.setKeyedPropertyValue("indexes", "MYINDEX", ["fields":["id",
"region"],
 "use_custom_create_index_command":True,
"custom_create_index_command":"CREATE INDEX <index-name> ON
 <table-name> <(index-columns)>"])
```

Propriedades de datacollectionexportnode

O nó de exportação do Coleta de Dados gera dados no formato utilizado pelo software de pesquisa de mercado do Coleta de Dados. Uma Biblioteca de Dados Coleta de Dados deve ser instalada para utilizar este nó.

Exemplo

```
stream = modeler.script.stream()
datacollectionexportnode = stream.createAt("datacollectionexport", "Data
Collection", 200, 200)
```

```
datacollectionexportnode.setPropertyValue("metadata_file", "c:\\museums.mdd")
datacollectionexportnode.setPropertyValue("merge_metadata", "Overwrite")
datacollectionexportnode.setPropertyValue("casedata_file", "c:\
\museumdata.sav")
datacollectionexportnode.setPropertyValue("generate_import", True)
datacollectionexportnode.setPropertyValue("enable_system_variables", True)
```

Tabela 248. Propriedades de datacollectionexportnode

propriedades datacollectionexportnode	Tipo de dados	Descrição da propriedade
metadata_file	*sequência*	O nome do arquivo de metadados para exportar.
merge_metadata	Overwrite MergeCurrent	
enable_system_variables	*sinalização*	Especifica se o arquivo *.mdd* exportado deve incluir as variáveis do sistema Coleta de Dados.
casedata_file	*sequência*	O nome do arquivo *.sav* para o qual os dados do caso são exportados.
generate_import	*sinalização*	

Propriedades de excelexportnode

O nó de exportação do Excel saídas dados no Microsoft Excel.`xlsx` formato de arquivo. Opcionalmente, você pode optar por ativar o Excel automaticamente e abrir o arquivo exportado quando o nó for executado.

Exemplo

```
stream = modeler.script.stream()
excelexportnode = stream.createAt("excelexport", "Excel", 200, 200)
excelexportnode.setPropertyValue("full_filename", "C:/output/myexport.xlsx")
excelexportnode.setPropertyValue("excel_file_type", "Excel2007")
excelexportnode.setPropertyValue("inc_field_names", True)
excelexportnode.setPropertyValue("inc_labels_as_cell_notes", False)
excelexportnode.setPropertyValue("launch_application", True)
excelexportnode.setPropertyValue("generate_import", True)
```

Tabela 249. Propriedades de excelexportnode

propriedades excelexportnode	Tipo de dados	Descrição da propriedade
full_filename	*sequência*	
excel_file_type	Excel2007	
export_mode	Create Append	
inc_field_names	*sinalização*	Especifica se nomes de campos devem ser incluídos na primeira linha da planilha.

Tabela 249. Propriedades de excelexportnode (continuação)		
propriedades excelexportnode	**Tipo de dados**	**Descrição da propriedade**
start_cell	*sequência*	Especifica célula inicial para exportação.
worksheet_name	*sequência*	O nome da planilha a ser gravada.
launch_application	*sinalização*	Especifica se o Excel deve ser chamado no arquivo resultante. Observe que o caminho para ativação do Excel deve ser especificado na caixa de diálogo Aplicativos Auxiliares (menu Ferramentas, Aplicativos Auxiliares).
generate_import	*sinalização*	Especifica se um nó de importação do Excel deve ser gerado que lerá o arquivo de dados exportado.

propriedades extensionexportnode

Com o nó Exportação de extensão, é possível executar scripts R ou Python para Spark para exportar dados.

Exemplo de Python for Spark

```
#### script example for Python for Spark
import modeler.api
stream = modeler.script.stream()
node = stream.create("extension_export", "extension_export")
node.setPropertyValue("syntax_type", "Python")

python_script = """import spss.pyspark.runtime
from pyspark.sql import SQLContext
from pyspark.sql.types import *

cxt = spss.pyspark.runtime.getContext()
df = cxt.getSparkInputData()
print df.dtypes[:]
_newDF = df.select("Age","Drug")
print _newDF.dtypes[:]

df.select("Age", "Drug").write.save("c:/data/ageAndDrug.json", format="json")
"""

node.setPropertyValue("python_syntax", python_script)
```

Exemplo de R

```
#### script example for R
node.setPropertyValue("syntax_type", "R")
node.setPropertyValue("r_syntax", """write.csv(modelerData, "C:/export.csv")""")
```

Tabela 250. propriedades extensionexportnode		
propriedades extensionexportnode	**Tipo de dados**	**Descrição da propriedade**
syntax_type	*R* *Python*	Especifique qual script é executado - R ou Python (R é o padrão).
r_syntax	*sequência*	A sintaxe de script do R a ser executada.
python_syntax	*sequência*	A sintaxe de script Python a ser executada.
convert_flags	`StringsAndDoubles` `LogicalValues`	Opção para converter os campos de sinalização.
convert_missing	*sinalização*	Opção para converter valores ausentes para o valor NA do R.
convert_datetime	*sinalização*	Opção para converter variáveis com os formatos de data ou data/hora em formatos de data/hora R.
convert_datetime_class	`POSIXct` `POSIXlt`	Opções para especificar em qual formato as variáveis com os formatos de data ou data/hora serão convertidas.

Propriedades jsonexportnode

O nó de exportação JSON supera dados em formato JSON.

Tabela 251. Propriedades jsonexportnode		
propriedades jsonexportnode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	O nome do arquivo completo, incluindo o caminho.
string_format	*registros* *valores*	Especifique o formato da sequência JSON. O padrão é `records`.
generate_import	*sinalização*	Especifica se um nó de Importação JSON deve ser gerado para ler o arquivo de dados exportados. O padrão é `False`.

Propriedades de outputfilenode

O nó Flat File Export gera dados para um arquivo de texto delimitado. Ele é útil para exportar dados que podem ser lidos por outras análises ou software de planilha.

Exemplo

```
stream = modeler.script.stream()
outputfile = stream.createAt("outputfile", "File Output", 200, 200)
outputfile.setPropertyValue("full_filename", "c:/output/flatfile_output.txt")
outputfile.setPropertyValue("write_mode", "Append")
outputfile.setPropertyValue("inc_field_names", False)
outputfile.setPropertyValue("use_newline_after_records", False)
outputfile.setPropertyValue("delimit_mode", "Tab")
outputfile.setPropertyValue("other_delimiter", ",")
outputfile.setPropertyValue("quote_mode", "Double")
outputfile.setPropertyValue("other_quote", "*")
outputfile.setPropertyValue("decimal_symbol", "Period")
outputfile.setPropertyValue("generate_import", True)
```

Tabela 252. Propriedades de outputfilenode

propriedades outputfilenode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	Nome do arquivo de saída.
write_mode	Overwrite Append	
inc_field_names	*sinalização*	
use_newline_after_records	*sinalização*	
delimit_mode	Comma Tab Space Other	
other_delimiter	*char*	
quote_mode	None Single Double Other	
other_quote	*sinalização*	
generate_import	*sinalização*	

Tabela 252. Propriedades de outputfilenode (continuação)		
propriedades outputfilenode	**Tipo de dados**	**Descrição da propriedade**
encoding	StreamDefault SystemDefault "UTF-8"	

Propriedades de sasexportnode

O nó de exportação SAS gera dados em formato do SAS a serem lidos no SAS ou em um pacote de software compatível com o SAS. Três formatos de arquivo SAS estão disponíveis: SAS para Windows/OS2, SAS para UNIX ou SAS Versão 7/8.

Exemplo

```
stream = modeler.script.stream()
sasexportnode = stream.createAt("sasexport", "SAS Export", 200, 200)
sasexportnode.setPropertyValue("full_filename", "c:/output/
SAS_output.sas7bdat")
sasexportnode.setPropertyValue("format", "SAS8")
sasexportnode.setPropertyValue("export_names", "NamesAndLabels")
sasexportnode.setPropertyValue("generate_import", True)
```

Tabela 253. Propriedades de sasexportnode		
propriedades sasexportnode	**Tipo de dados**	**Descrição da propriedade**
format	Windows UNIX SAS7 SAS8	Campos de rótulo de propriedade variante
full_filename	*sequência*	
export_names	NamesAndLabels NamesAsLabels	Usado para mapear os nomes de campos a partir do IBM SPSS Modeler na exportação para o IBM SPSS Statistics ou nomes de variáveis do SAS.
generate_import	*sinalização*	

Propriedades de statisticsexportnode

O nó Exportação de Estatísticas gera dados no formato IBM SPSS Statistics *.sav* ou *.zsav*. Os arquivos `.sav` ou `.zsav` podem ser lidos pelo IBM SPSS Statistics Base e por outros produtos. Este também é o formato usado para arquivos de cache em IBM SPSS Modeler.

As propriedades desse nó são descritas em "Propriedades de statisticsexportnode" na página 434.

Propriedades do Nó tm1odataexport :

O nó de exportação do IBM Cognos TM1 exporta dados em um formato que pode ser lido por bancos de dados do Cognos TM1.

Tabela 254. Propriedades do nó tm1odataexport .

tm1odataexport propriedades do nó	**Tipo de dados**	**Descrição da propriedade**
credential_type	inputCredential ou storedCredential	Usado para indicar o tipo de credencial
input_credential	*lista*	Quando o credential_type for inputCredential; especifique o domínio, nome de usuário e senha.
stored_credential_name	*sequência*	Quando o credential_type for storedCredential; especifique o nome da credencial no servidor C & DS
selected_cube	*campo*	O nome do cubo para o qual você está exportando dados. Por exemplo: TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")

Tabela 254. Propriedades do nó tm1odataexport . (continuação)

tm1odataexport propriedades do nó	**Tipo de dados**	**Descrição da propriedade**
spss_field_to_tm1_element_mapping	*lista*	O elemento tm1 a ser mapeado deve fazer parte da dimensão da coluna para visualização de cubo selecionada. O formato é: [[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]] Há 2 listas para descrever as informações de mapeamento O mapeamento de um elemento folha para uma dimensão corresponde ao exemplo 2 abaixo: Exemplo 1: A primeira lista: ([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...]) é usada para as informações do mapa de dimensão do TM1 . Cada lista de valores de 3 indica informações de mapeamento de dimensão O terceiro valor booleano é usado para indicar se ele seleciona um elemento de uma dimensão Por exemplo: "[Field_1, Dimension_1, False]" significa que Field_1 é mapeado para Dimension_1; "[Element_1, Dimension_2, True]" significa que Element_1 é selecionado para Dimension_2. Exemplo 2: A segunda lista: ([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]) é usada para as informações do mapa do elemento de dimensão de medida do TM1 . Cada lista de valores 3 indica informações de mapeamento do elemento de medida. O terceiro valor booleano é usado para indicar a necessidade de criar um novo elemento "[Field_2, ExistMeasureElement, False]" significa que Field_2 é mapeado para ExistMeasureElement; "[Field_3, NewMeasureElement, True]" significa que NewMeasureElement precisa ser a dimensão de medida escolhida em selected_measure e que Field_3 é mapeado para ele.
selected_measure	*sequência*	Especifique a dimensão de medida. Exemplo: setPropertyValue("selected_measure", "Measures") .
connection_type	AdminServer TM1Server	Indica o tipo de conexão. O padrão é AdminServer.

Tabela 254. Propriedades do nó tm1odataexport . (continuação)		
tm1odataexport propriedades do nó	**Tipo de dados**	**Descrição da propriedade**
admin_host	*sequência*	A URL para o nome do host da API REST. Necessário se o connection_type for AdminServer
server_name	*sequência*	O nome do servidor TM1 selecionado a partir do admin_host. Necessário se o connection_type for AdminServer
server_url	*sequência*	A URL para a API de REST do TM1 Server Necessário se o connection_type for TM1Server

tm1export Propriedades do Nó (descontinuado)

O nó de exportação do IBM Cognos TM1 exporta dados em um formato que pode ser lido por bancos de dados do Cognos TM1.

Nota: Esse nó foi descontinuado no Modeler 18.0 O nome do script do nó de substituição é *tm1odataexport*

Tabela 255. Propriedades do nó tm1export		
Propriedades do nó tm1export	**Tipo de dados**	**Descrição da propriedade**
pm_host	*sequência*	**Nota:** Somente para as versões 16.0 e 17.0 O nome do host. Por exemplo: TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')
tm1_connection	*["campo", "campo" ..., "campo"]*	**Nota:** Somente para as versões 16.0 e 17.0 Uma propriedade de lista que contém os detalhes da conexão com o servidor TM1. O formato é: ["TM1_Server_Name", "tm1_username", "tm1_password"] Por exemplo: TM1_export.setPropertyValue("tm1_connection", ['Planning Sample', "admin" "apple"])
selected_cube	*campo*	O nome do cubo para o qual você está exportando dados. Por exemplo: TM1_export.setPropertyValue("selected_cube", "plan_BudgetPlan")

Tabela 255. Propriedades do nó tm1export (continuação)

Propriedades do nó `tm1export`	Tipo de dados	Descrição da propriedade
`spssfield_tm1element_mapping`	*lista*	O elemento tm1 a ser mapeado deve fazer parte da dimensão da coluna para visualização de cubo selecionada. O formato é: `[[[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...], [[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...]]` Há 2 listas para descrever as informações de mapeamento.. O mapeamento de um elemento folha para uma dimensão corresponde ao exemplo 2 abaixo: Exemplo 1: A primeira lista: `([[Field_1, Dimension_1, False], [Element_1, Dimension_2, True], ...])` é usada para as informações do mapa de dimensão TM1 . Cada lista de 3 valores indica informações de mapeamento de dimensão. O terceiro valor booleano é usado para indicar se ele seleciona um elemento de uma dimensão Por exemplo: `"[Field_1, Dimension_1, False]"` significa que `Field_1` é mapeado para `Dimension_1`; `"[Element_1, Dimension_2, True]"` significa que `Element_1` é selecionado para `Dimension_2` Exemplo 2: A segunda lista: `([[Field_2, ExistMeasureElement, False], [Field_3, NewMeasureElement, True], ...])` é usada para as informações do mapa do elemento de dimensão de medida do TM1 . Cada lista de 3 valores indica informações de mapeamento do elemento de medida O terceiro valor booleano é usado para indicar a necessidade de criar um novo elemento `"[Field_2, ExistMeasureElement, False]"` significa que `Field_2` é mapeado para `ExistMeasureElement`; `"[Field_3, NewMeasureElement, True]"` significa que `NewMeasureElement` precisa ser a dimensão de medida escolhida em `selected_measure` e que `Field_3` é mapeado para ele.

Tabela 255. Propriedades do nó tm1export (continuação)		
Propriedades do nó `tm1export`	**Tipo de dados**	**Descrição da propriedade**
`selected_measure`	*sequência*	Especifique a dimensão de medida Exemplo: `setPropertyValue("selected_measure", "Measures")`

Propriedades de xmlexportnode

O nó de exportação XML gera dados para um arquivo no formato XML. Opcionalmente, é possível criar um nó de origem XML para ler os dados exportados de volta no fluxo.

Exemplo

```
stream = modeler.script.stream()
xmlexportnode = stream.createAt("xmlexport", "XML Export", 200, 200)
xmlexportnode.setPropertyValue("full_filename", "c:/export/data.xml")
xmlexportnode.setPropertyValue("map", [["/catalog/book/genre", "genre"], ["/
catalog/book/title", "title"]])
```

Tabela 256. Propriedades de xmlexportnode		
propriedades `xmlexportnode`	**Tipo de dados**	**Descrição da propriedade**
`full_filename`	*sequência*	(necessário) Caminho e o nome do arquivo completos do arquivo de exportação XML.
`use_xml_schema`	*sinalização*	Especifica se deseja usar um esquema XML (arquivo XSD ou DTD) para controlar a estrutura dos dados exportados.
`full_schema_filename`	*sequência*	Caminho e nome de arquivo completos do arquivo XSD ou DTD a ser utilizado. Necessário se `use_xml_schema` for configurado como true.
`generate_import`	*sinalização*	Gera um nó de origem XML que lerá o arquivo de dados exportado de volta no fluxo.
`records`	*sequência*	Expressão XPath que indica o limite do registro.
`map`	*sequência*	Mapeia o nome do campo para estrutura XML.

Capítulo 18. IBM SPSS Statistics Propriedades do Nó

Propriedades de statisticsimportnode

O nó Arquivo de Estatísticas lê dados do formato de arquivo *.sav* ou *.zsav* usado pelo IBM SPSS Statistics, bem como de arquivos de cache salvos em IBM SPSS Modeler que também utilizam o mesmo formato.

Exemplo

```
stream = modeler.script.stream()
statisticsimportnode = stream.createAt("statisticsimport", "SAV Import",
200, 200)
statisticsimportnode.setPropertyValue("full_filename", "C:/data/drug1n.sav")
statisticsimportnode.setPropertyValue("import_names", True)
statisticsimportnode.setPropertyValue("import_data", True)
```

Tabela 257. propriedades de statisticsimportnode

propriedades statisticsimportnode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*sequência*	O nome do arquivo completo, incluindo o caminho.
password	*sequência*	A senha. O parâmetro password deve ser configurado antes do parâmetro file_encrypted.
file_encrypted	*sinalização*	Se o arquivo deve ou não ser protegido por senha.
import_names	NamesAndLabels LabelsAsNames	Método para manipular nomes e rótulos de variáveis.
import_data	DataAndLabels LabelsAsData	Método para manipular valores e rótulos.
use_field_format_for_storage	*Booleano*	Especifica se informações de formato de campo do IBM SPSS Statistics devem ser usadas ao importar.

Propriedades de statisticstransformnode

O nó Transformação de Estatísticas executa uma seleção de comandos de sintaxe do IBM SPSS Statistics com relação às origens de dados no IBM SPSS Modeler. Este nó requer uma cópia licenciada de IBM SPSS Statistics.

Exemplo

```
stream = modeler.script.stream()
statisticstransformnode = stream.createAt("statisticstransform",
```

```
"Transform", 200, 200)
statisticstransformnode.setPropertyValue("syntax", "COMPUTE NewVar = Na +
K.")
statisticstransformnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed
Drugs")
statisticstransformnode.setPropertyValue("check_before_saving", True)
```

Tabela 258. Propriedades de statisticstransformnode

propriedades statisticstransformnode	Tipo de dados	Descrição da propriedade
syntax	*sequência*	
check_before_saving	*sinalização*	Valida a sintaxe inserida antes de salvar as entradas. Exibirá uma mensagem de erro se a sintaxe for inválida.
default_include	*sinalização*	Consulte o tópico "Propriedades de filternode" na página 168 para obter informações adicionais.
include	*sinalização*	Consulte o tópico "Propriedades de filternode" na página 168 para obter informações adicionais.
new_name	*sequência*	Consulte o tópico "Propriedades de filternode" na página 168 para obter informações adicionais.

Propriedades de statisticsmodelnode

O nó Modelo de Estatísticas permite analisar e trabalhar com seus dados executando os procedimentos do IBM SPSS Statistics que produzem o PMML. Este nó requer uma cópia licenciada de IBM SPSS Statistics.

Exemplo

```
stream = modeler.script.stream()
statisticsmodelnode = stream.createAt("statisticsmodel", "Model", 200, 200)
statisticsmodelnode.setPropertyValue("syntax", "COMPUTE NewVar = Na + K.")
statisticsmodelnode.setKeyedPropertyValue("new_name", "NewVar", "Mixed
Drugs")
```

propriedades statisticsmodelnode	Tipo de dados	Descrição da propriedade
syntax	*sequência*	
default_include	*sinalização*	Consulte o tópico "Propriedades de filternode" na página 168 para obter informações adicionais.
include	*sinalização*	Consulte o tópico "Propriedades de filternode" na página 168 para obter informações adicionais.

propriedades statisticsmodelnode	Tipo de dados	Descrição da propriedade
new_name	*sequência*	Consulte o tópico "Propriedades de filternode" na página 168 para obter informações adicionais.

Propriedades de statisticsoutputnode

O nó Saída de Estatísticas permite chamar um procedimento do IBM SPSS Statistics para analisar seus dados do IBM SPSS Modeler. Uma ampla variedade de procedimentos de analítica do IBM SPSS Statistics está disponível. Este nó requer uma cópia licenciada de IBM SPSS Statistics.

Exemplo

```
stream = modeler.script.stream()
statisticsoutputnode = stream.createAt("statisticsoutput", "Output", 200,
200)
statisticsoutputnode.setPropertyValue("syntax", "SORT CASES BY Age(A) Sex(A)
BP(A) Cholesterol(A)")
statisticsoutputnode.setPropertyValue("use_output_name", False)
statisticsoutputnode.setPropertyValue("output_mode", "File")
statisticsoutputnode.setPropertyValue("full_filename", "Cases by Age, Sex
and Medical History")
statisticsoutputnode.setPropertyValue("file_type", "HTML")
```

Tabela 259. Propriedades de statisticsoutputnode

propriedades statisticsoutputnode	Tipo de dados	Descrição da propriedade
mode	Dialog Syntax	Seleciona a opção "Diálogo do IBM SPSS Statistics" ou o Editor de Sintaxe
syntax	*sequência*	
use_output_name	*sinalização*	
output_name	*sequência*	
output_mode	Screen File	
full_filename	*sequência*	
file_type	HTML SPV SPW	

Propriedades de statisticsexportnode

O nó Exportação de Estatísticas gera dados no formato IBM SPSS Statistics *.sav* ou *.zsav*. Os arquivos `.sav` ou `.zsav` podem ser lidos pelo IBM SPSS Statistics Base e por outros produtos. Este também é o formato usado para arquivos de cache em IBM SPSS Modeler.

Exemplo

```
stream = modeler.script.stream()
statisticsexportnode = stream.createAt("statisticsexport", "Export", 200,
200)
statisticsexportnode.setPropertyValue("full_filename", "c:/output/
SPSS_Statistics_out.sav")
statisticsexportnode.setPropertyValue("field_names", "Names")
statisticsexportnode.setPropertyValue("launch_application", True)
statisticsexportnode.setPropertyValue("generate_import", True)
```

Tabela 260. Propriedades de statisticsexportnode

propriedades statisticsexportnode	**Tipo de dados**	**Descrição da propriedade**
`full_filename`	*sequência*	

`file_type`	`sav`
	
	`zsav`
		Arquivo de salvamento no formato *sav* ou *zsav*. Por exemplo:
		
		`statisticsexportnode.setPropertyValue("file_type","sav")`

`encrypt_file`	*sinalização*	Se o arquivo deve ou não ser protegido por senha.
`password`	*sequência*	A senha.
`launch_application`	*sinalização*	

`export_names`	`NamesAndLabels`
	
	`NamesAsLabels`
		Usado para mapear os nomes de campos a partir do IBM SPSS Modeler na exportação para o IBM SPSS Statistics ou nomes de variáveis do SAS.

`generate_import`	*sinalização*	

Capítulo 19. Propriedades do nó Python

propriedades gmm

Um modelo Gaussian Mixture© é um modelo probabilístico que assume que todos os pontos de dados são gerados a partir de uma mistura de um número finito de distribuições gaussianas com parâmetros desconhecidos. Pode-se pensar em modelos de mistura como generalização clusterização de k-médias para incorporar informações sobre a estrutura de covariância dos dados, assim como os centros dos Gaussianos latentes. O Nó da Mistura Gaussiana em SPSS Modeler expõe os principais recursos e parâmetros comumente usados da biblioteca de Mistura Gaussiana. O nó é implementado em Python.

Tabela 261. propriedades gmm

propriedades gmm	**Tipo de dados**	**Descrição da propriedade**
`use_partition`	*Booleano*	Configure como `True` ou `False` para especificar se deve usar dados particionados. O padrão é `False`.
`covariance_type`	*sequência*	Especifique `Full`, `Tied`, `Diag`ou `Spherical` para configurar o tipo de covariância.
`number_component`	*Número inteiro*	Especifique um número inteiro para o número de componentes da mistura. O valor mínimo é 1. O valor padrão é 2.
`component_lable`	*Booleano*	Especifique `True` para configurar o rótulo do cluster para uma sequência ou `False` para configurar o rótulo do cluster para um número. O padrão é `False`.
`label_prefix`	*sequência*	Se estiver usando um rótulo de cluster de sequência, será possível especificar um prefixo.
`enable_random_seed`	*Booleano*	Especifique `True` se você desejar usar um valor inicial aleatório. O patrão é `False`.
`random_seed`	*Número inteiro*	Se estiver usando um valor inicial aleatório, especifique um número inteiro para ser usado para gerar amostras aleatórias.
`tol`	*double*	Especifique o limite de convergência. O padrão é `0.000.1`.
`max_iter`	*Número inteiro*	Especifique o número máximo de iterações a serem executadas. O padrão é `100`.
`init_params`	*sequência*	Configure o parâmetro de inicialização a ser usado. As opções são `Kmeans` ou `Random`.

Tabela 261. propriedades gmm (continuação)		
propriedades gmm	**Tipo de dados**	**Descrição da propriedade**
warm_start	*Booleano*	Especifique True para usar a solução do último ajuste como a inicialização para a próxima chamada de ajuste. O patrão é False.

propriedades hdbscannode

O Hierarchical Density-Based Spatial Clustering (HDBSCAN)© usa o aprendizado não supervisionado para localizar clusters ou regiões densas de um conjunto de dados. O nó HDBSCAN em SPSS Modeler expõe os recursos principais e parâmetros comumente usados da biblioteca HDBSCAN. O nó é implementado no Python e é possível usá-lo para agrupar seu conjunto de dados em grupos distintos quando não sabe o que esses grupos são a princípio.

Tabela 262. propriedades hdbscannode		
propriedades hdbscannode	**Tipo de dados**	**Descrição da propriedade**
inputs	*campo*	Campos de entrada para armazenamento em cluster.
useHPO	*Booleano*	Especifique true ou false para ativar ou desativar o Hyper-Parameter Optimization (HPO) baseado em Rbfopt, que descobre automaticamente a combinação ideal de parâmetros para que o modelo alcance a taxa de erro esperada ou inferior nas amostras. O padrão é false.
min_cluster_size	*Número inteiro*	O tamanho mínimo de clusters. Especifique um número inteiro. O padrão é 5.
min_samples	*Número inteiro*	O número de amostras em uma vizinhança para um ponto a ser considerado um ponto principal. Especifique um número inteiro. Se configurado como 0, o min_cluster_size é usado. O padrão é 0.
algorithm	*sequência*	Especifique qual algoritmo usar: best, generic, prims_kdtree, prims_balltree, boruvka_kdtree ou boruvka_balltree. O padrão é best.
metric	*sequência*	Especifique qual métrica usar ao calcular distância entre instâncias em uma matriz de recursos: euclidean, cityblock, L1, L2, manhattan, braycurtis, canberra, chebyshev, correlation, minkowski ou sqeuclidean. O padrão é euclidean.

Tabela 262. propriedades hdbscannode (continuação)

propriedades hdbscannode	Tipo de dados	Descrição da propriedade
useStringLabel	*Booleano*	Especifique `true` para usar um rótulo de cluster de sequência ou `false` para usar um rótulo de cluster de número. O padrão é `false`.
stringLabelPrefix	*sequência*	Se o parâmetro `useStringLabel` for configurado como `true`, especifique um valor para o prefixo do rótulo de sequência. O prefixo padrão é `cluster`.
approx_min_span_tree	*Booleano*	Especifique `true` para aceitar uma árvore de amplitude mínima aproximada ou `false` se você estiver disposto a sacrificar a velocidade para correção. O padrão é `true`.
cluster_selection_method	*sequência*	Especifique o método a ser usado para selecionar clusters na árvore condensada: `eom` ou `leaf`. O padrão é `eom` (algoritmo Excesso de massa).
allow_single_cluster	*Booleano*	Especifique `true` se você deseja permitir resultados de cluster únicos. O patrão é `false`.
p_value	*double*	Especifique o `p value` para usar se você estiver usando `minkowski` para a métrica. O padrão é `1.5`.
leaf_size	*Número inteiro*	Se usar um algoritmo de árvore espacial (`boruvka_kdtree` ou `boruvka_balltree`), especifique o número de pontos em um nó de folha da árvore. O padrão é `40`.
outputValidity	*Booleano*	Especifique `true` ou `false` para controlar se o gráfico Índice de validade está incluído na saída do modelo.
outputCondensed	*Booleano*	Especifique `true` ou `false` para controlar se o gráfico de Árvore condensada está incluído na saída do modelo.
outputSingleLinkage	*Booleano*	Especifique `true` ou `false` para controlar se o gráfico Árvore de ligação única está incluído na saída do modelo.
outputMinSpan	*Booleano*	Especifique `true` ou `false` para controlar se o gráfico Árvore de abrangência mínima está incluído na saída do modelo.
is_split		Incluído na versão 18.2.1.1..

propriedades kdemodel

Estimativa De Densidade Do Kernel (KDE) © utiliza os algoritmos Ball Tree ou KD Tree para consultas eficientes, e combina conceitos a partir de aprendizado não supervisionado, engenharia de recursos e modelagem de dados. Abordagens baseadas em vizinhos, como o KDE, são algumas das técnicas de estimativa de densidade mais populares e úteis. Os Nós KDE Modelagem e Simulação KDE em SPSS Modeler expõem os principais recursos e parâmetros comumente usados da biblioteca KDE. Os nós são implementados em Python.

Tabela 263. propriedades kdemodel

propriedades kdemodel	**Tipo de dados**	**Descrição da propriedade**
`bandwidth`	*double*	O padrão é `1`.
`kernel`	*sequência*	O kernel a ser usado: `gaussian`, `tophat`, `epanechnikov`, `exponential`, `linear` ou `cosine`. O padrão é `gaussian`.
`algorithm`	*sequência*	O algoritmo da árvore a ser usado: `kd_tree`, `ball_tree` ou `auto`. O padrão é `auto`.
`metric`	*sequência*	A métrica a ser usada ao calcular distância. Para o algoritmo `kd_tree`, escolha entre: `Euclidean`, `Chebyshev`, `Cityblock`, `Minkowski`, `Manhattan`, `Infinity`, `P`, `L2` ou `L1`. Para o algoritmo `ball_tree`, escolha entre: `Euclidian`, `Braycurtis`, `Chebyshev`, `Canberra`, `Cityblock`, `Dice`, `Hamming`, `Infinity`, `Jaccard`, `L1`, `L2`, `Minkowski`, `Matching`, `Manhattan`, `P`, `Rogersanimoto`, `Russellrao`, `Sokalmichener`, `Sokalsneath` ou `Kulsinski`. O padrão é `Euclidean`.
`atol`	*Valor flutuante*	A tolerância absoluta desejada do resultado. Uma tolerância maior geralmente levará a uma execução mais rápida. O padrão é `0.0`.
`rtol`	*Valor flutuante*	A tolerância relativa desejada do resultado. Uma tolerância maior geralmente levará a uma execução mais rápida. O padrão é `1E-8`.
`breadthFirst` renomeado para `breadth_first` começando com a versão 18.2.1.1	*Booleano*	Configure como `True` para usar uma abordagem de amplitude primeiro. Configure como `False` para usar uma abordagem de profundidade primeiro. O padrão é `True`.
`LeafSize` renomeado para `leaf_size` começando com a versão 18.2.1.1	*Número inteiro*	O tamanho de folha da árvore subjacente. O padrão é `40`. A mudança deste valor pode impactar significativamente o desempenho.

Tabela 263. propriedades kdemodel (continuação)		
propriedades kdemodel	**Tipo de dados**	**Descrição da propriedade**
pValue	*double*	Especifique o Valor P para usar se você estiver usando Minkowski para a métrica. O patrão é 1.5.
custom_name		
default_node_name		
use_HPO		

propriedades kdeexport

Estimativa De Densidade Do Kernel (KDE) © utiliza os algoritmos Ball Tree ou KD Tree para consultas eficientes, e combina conceitos a partir de aprendizado não supervisionado, engenharia de recursos e modelagem de dados. Abordagens baseadas em vizinhos, como o KDE, são algumas das técnicas de estimativa de densidade mais populares e úteis. Os Nós KDE Modelagem e Simulação KDE em SPSS Modeler expõem os principais recursos e parâmetros comumente usados da biblioteca KDE. Os nós são implementados em Python.

Tabela 264. propriedades kdeexport		
propriedades kdeexport	**Tipo de dados**	**Descrição da propriedade**
bandwidth	*double*	O padrão é 1.
kernel	*sequência*	O kernel a ser usado: gaussian ou tophat. O padrão é gaussian.
algorithm	*sequência*	O algoritmo da árvore a ser usado: kd_tree, ball_tree ou auto. O padrão é auto.
metric	*sequência*	A métrica a ser usada ao calcular distância. Para o algoritmo kd_tree, escolha entre: Euclidean, Chebyshev, Cityblock, Minkowski, Manhattan, Infinity, P, L2 ou L1. Para o algoritmo ball_tree, escolha entre: Euclidian, Braycurtis, Chebyshev, Canberra, Cityblock, Dice, Hamming, Infinity, Jaccard, L1, L2, Minkowski, Matching, Manhattan, P, Rogersanimoto, Russellrao, Sokalmichener, Sokalsneath ou Kulsinski. O padrão é Euclidean.
atol	*Valor flutuante*	A tolerância absoluta desejada do resultado. Uma tolerância maior geralmente levará a uma execução mais rápida. O padrão é 0.0.
rtol	*Valor flutuante*	A tolerância relativa desejada do resultado. Uma tolerância maior geralmente levará a uma execução mais rápida. O padrão é 1E-8.

Tabela 264. propriedades kdeexport (continuação)		
propriedades kdeexport	**Tipo de dados**	**Descrição da propriedade**
breadthFirst	*Booleano*	Configure como True para usar uma abordagem de amplitude primeiro. Configure como False para usar uma abordagem de profundidade primeiro. O padrão é True.
LeafSize	*Número inteiro*	O tamanho de folha da árvore subjacente. O padrão é 40. A mudança deste valor pode impactar significativamente o desempenho.
pValue	*double*	Especifique o Valor P para usar se você estiver usando Minkowski para a métrica. O patrão é 1.5.

propriedades gmm

Um modelo Gaussian Mixture© é um modelo probabilístico que assume que todos os pontos de dados são gerados a partir de uma mistura de um número finito de distribuições gaussianas com parâmetros desconhecidos. Pode-se pensar em modelos de mistura como generalização clusterização de k-médias para incorporar informações sobre a estrutura de covariância dos dados, assim como os centros dos Gaussianos latentes. O Nó da Mistura Gaussiana em SPSS Modeler expõe os principais recursos e parâmetros comumente usados da biblioteca de Mistura Gaussiana. O nó é implementado em Python.

Tabela 265. propriedades gmm		
propriedades gmm	**Tipo de dados**	**Descrição da propriedade**
use_partition	*Booleano*	Configure como True ou False para especificar se deve usar dados particionados. O padrão é False.
covariance_type	*sequência*	Especifique Full, Tied, Diagou Spherical para configurar o tipo de covariância.
number_component	*Número inteiro*	Especifique um número inteiro para o número de componentes da mistura. O valor mínimo é 1. O valor padrão é 2.
component_lable	*Booleano*	Especifique True para configurar o rótulo do cluster para uma sequência ou False para configurar o rótulo do cluster para um número. O padrão é False.
label_prefix	*sequência*	Se estiver usando um rótulo de cluster de sequência, será possível especificar um prefixo.
enable_random_seed	*Booleano*	Especifique True se você desejar usar um valor inicial aleatório. O patrão é False.

Tabela 265. propriedades gmm (continuação)		
propriedades gmm	**Tipo de dados**	**Descrição da propriedade**
random_seed	*Número inteiro*	Se estiver usando um valor inicial aleatório, especifique um número inteiro para ser usado para gerar amostras aleatórias.
tol	*double*	Especifique o limite de convergência. O padrão é 0.000.1.
max_iter	*Número inteiro*	Especifique o número máximo de iterações a serem executadas. O padrão é 100.
init_params	*sequência*	Configure o parâmetro de inicialização a ser usado. As opções são Kmeans ou Random.
warm_start	*Booleano*	Especifique True para usar a solução do último ajuste como a inicialização para a próxima chamada de ajuste. O patrão é False.

propriedades ocsvmnode

O nó SVM de Uma Classe utiliza um algoritmo de aprendizagem não supervisionado. O nó pode ser usado para detecção de novidades. Ele detectará o limite flexível de um determinado conjunto de amostras, para então classificar novos pontos como pertencentes ou não a esse conjunto. Este nó de modelagem SVM de Classe One em SPSS Modeler é implementado em Python e requer a biblioteca scikit-learn© Python .

Tabela 266. propriedades ocsvmnode		
propriedades ocsvmnode	**Tipo de dados**	**Descrição da propriedade**
role_use Renomeado para custom_fields a partir da versão 18.2.1.1	*sequência*	Especifique predefined para usar funções predefinidas ou custom para usar designações de campo customizadas. O padrão é predefinido.
splits	*campo*	Lista dos nomes de campo para divisão.
use_partition	*Booleano*	Especifique true ou false. O padrão é true. Se configurado como true, somente dados de treinamento serão usados ao construir o modelo.
mode_type	*sequência*	O modo. Os valores possíveis são simple ou expert. Todos os parâmetros na guia Especialista serão desativados se simple for especificado.
stopping_criteria	*sequência*	Uma sequência de notação científica. Os valores possíveis são 1.0E-1, 1.0E-2, 1.0E-3, 1.0E-4, 1.0E-5 ou 1.0E-6. O padrão é 1.0E-3.

Tabela 266. propriedades ocsvmnode (continuação)

propriedades ocsvmnode	Tipo de dados	Descrição da propriedade
precision	*Valor flutuante*	A precisão de regressão (nu). Limite na fração de erros de treinamento e vetores de suporte. Especifique um número maior que 0 e menor ou igual a 1.0. O padrão é 0.1.
kernel	*sequência*	O tipo de kernel a ser usado no algoritmo. Os valores possíveis são linear, poly, rbf, sigmoid ou precomputed. O padrão é rbf.
enable_gamma	*Booleano*	Ativa o parâmetro gamma. Especifique true ou false. O padrão é true.
gamma	*Valor flutuante*	Esse parâmetro é ativado apenas para os kernels rbf, poly e sigmoid. Se o parâmetro enable_gamma for configurado como false, esse parâmetro será configurado como auto. Se configurado como true, o padrão será 0.1.
coef0	*Valor flutuante*	Termo independente na função kernel. Esse parâmetro é ativado apenas para o kernel poly e o kernel sigmoid. O valor padrão é 0.0.
degree	*Número inteiro*	Grau da função de kernel polinomial. Esse parâmetro só é ativado para o kernel poly. Especifique qualquer número inteiro. O patrão é 3.
shrinking	*Booleano*	Especifica se deve usar a opção heurística de redução. Especifique true ou false. O padrão é false.
enable_cache_size	*Booleano*	Ativa o parâmetro cache_size. Especifique true ou false. O padrão é false.
cache_size	*Valor flutuante*	O tamanho do cache do kernel em MB. O padrão é 200.
pc_type	*sequência*	O tipo do gráfico de coordenadas paralelas. As opções possíveis são independent ou general.
lines_amount	*Número inteiro*	Número máximo de linhas para incluir no gráfico. Especifique um número inteiro entre 1 e 1000.

Tabela 266. propriedades ocsvmnode (continuação)		
propriedades ocsvmnode	**Tipo de dados**	**Descrição da propriedade**
`lines_fields_custom`	*Booleano*	Ativa o parâmetro `lines_fields`, que permite especificar campos customizados para mostrar na saída do gráfico. Se configurado como `false`, todos os campos serão mostrados. Se configurado como `true`, apenas os campos especificados com o parâmetro lines_fields serão mostrados. Por motivos de desempenho, um máximo de 20 campos será exibido.
`lines_fields`	*campo*	Lista dos nomes de campo para incluir no gráfico como eixos verticais.
`enable_graphic`	*Booleano*	Especifique `true` ou `false`. Ativa a saída do gráfico (desative essa opção se você quiser economizar tempo e reduzir o tamanho do arquivo de fluxo).
`enable_hpo`	*Booleano*	Especifique `true` ou `false` para ativar ou desativar as opções de HPO. Se configurado como `true`, o Rbfopt será aplicado para descobrir o "melhor" modelo de One-Class SVM automaticamente, que atinge o valor do objetivo de destino definido pelo usuário com o parâmetro `target_objval` a seguir.
`target_objval`	*Valor flutuante*	O valor da função objetivo (taxa de erro do modelo nas amostras) que desejamos atingir (por exemplo, o valor do ideal desconhecido). Configure esse parâmetro para o valor apropriado se o ideal for desconhecido (por exemplo, `0.01`).
`max_iterations`	*Número inteiro*	O número máximo de iterações para tentar o modelo. O padrão é `1000`.
`max_evaluations`	*Número inteiro*	O número máximo de avaliações de função para tentar o modelo, em que o foco é a precisão sobre a velocidade. O padrão é `300`.

propriedades rfnode

O Random Forest nó utiliza uma implementação avançada de um algoritmo de bagging com um modelo de árvore como o modelo base. Este Nó de modelagem da Floresta Aleatória em SPSS Modeler é implementado em Python e requer a biblioteca `scikit-learn`© Python .

Tabela 267. propriedades rfnode		
propriedades rfnode	**Tipo de dados**	**Descrição da propriedade**
role_use	*sequência*	Especifique predefined para usar funções predefinidas ou custom para usar designações de campo customizadas. O padrão é predefinido.
inputs	*campo*	Lista dos nomes de campo para entrada.
splits	*campo*	Lista dos nomes de campo para divisão.
n_estimators	*Número inteiro*	Número de árvores a serem construídas. O padrão é 10.
specify_max_depth	*Booleano*	Especifique a profundidade máxima customizada. Se false, os nós são expandidos até que todas as folhas sejam puras ou até que todas as folhas contenham menos que min_samples_split amostras. O padrão é false.
max_depth	*Número inteiro*	A profundidade máxima da árvore. O padrão é 10.
min_samples_leaf	*Número inteiro*	Tamanho mínimo do nó da folha. O padrão é 1.
max_features	*sequência*	O número de recursos a considerar ao procurar a melhor divisão: • Se auto, max_features=sqrt(n_features) para classificador e max_features = sqrt (n_features) para regressão. • Se sqrt, então max_features=sqrt(n_features). • Se log2, então max_features=log2 (n_features). O padrão é auto.
bootstrap	*Booleano*	Use amostras de autoinicialização ao construir árvores. O padrão é true.
oob_score	*Booleano*	Use amostras out-of-bag para estimar a precisão de generalização. O valor padrão é false.
extreme	*Booleano*	Use árvores extremamente randomizadas. O padrão é false.
use_random_seed	*Booleano*	Especifique isso para obter resultados replicados. O padrão é false.
random_seed	*Número inteiro*	O valor inicial de número aleatório para usar ao construir árvores. Especifique qualquer número inteiro.
cache_size	*Valor flutuante*	O tamanho do cache do kernel em MB. O padrão é 200.

Tabela 267. propriedades rfnode (continuação)		
propriedades `rfnode`	**Tipo de dados**	**Descrição da propriedade**
`enable_random_seed`	*Booleano*	Ativa o parâmetro `random_seed`. Especifique true ou false. O patrão é `false`.
`enable_hpo`	*Booleano*	Especifique `true` ou `false` para ativar ou desativar as opções de HPO. Se configurado como `true`, o Rbfopt será aplicado para determinar o "melhor" modelo de Floresta aleatória automaticamente, que atinge o valor do objetivo do destino definido pelo usuário com o parâmetro `target_objval` a seguir.
`target_objval`	*Valor flutuante*	O valor da função objetiva (taxa de erro do modelo nas amostras) que você deseja atingir (por exemplo, o valor do desconhecido ideal). Configure esse parâmetro para o valor apropriado se o ideal for desconhecido (por exemplo, `0.01`).
`max_iterations`	*Número inteiro*	O número máximo de iterações para tentar o modelo. O padrão é `1000`.
`max_evaluations`	*Número inteiro*	O número máximo de avaliações de função para tentar o modelo, em que o foco é a precisão sobre a velocidade. O padrão é `300`.

propriedades smotenode

O nó Técnica de Sobreamostragem Minoritária Sintética (SMOTE) fornece um algoritmo de sobreamostragem para lidar com conjuntos de dados desbalanceados. Ele fornece um método avançado para balanceamento de dados. O nó do processo SMOTE em SPSS Modeler é implementado em Python e requer a biblioteca `imbalanced-learn`© Python .

Tabela 268. propriedades smotenode		
propriedades `smotenode`	**Tipo de dados**	**Descrição da propriedade**
`target_field` Renomeado para `target` a partir da versão 18.2.1.1	*campo*	O campo de destino.
`sample_ratio`	*sequência*	Ativa um valor de proporção customizado. As duas opções são Automático (`sample_ratio_auto`) ou Proporção configurada (`sample_ratio_manual`).

Tabela 268. propriedades smotenode (continuação)		
propriedades smotenode	**Tipo de dados**	**Descrição da propriedade**
sample_ratio_value	*Valor flutuante*	A razão é o número de amostras na classe minoritária sobre o número de amostras na classe majoritária. Ela deve ser maior que 0 e menor ou igual a 1. O padrão é auto.
enable_random_seed	*Booleano*	Se configurado como true, a propriedade random_seed será ativada.
random_seed	*Número inteiro*	O valor inicial usado pelo gerador de número aleatório.
k_neighbours	*Número inteiro*	O número de vizinhos mais próximos a serem usados para construir amostras sintéticas. O padrão é 5.
m_neighbours	*Número inteiro*	O número de vizinhos mais próximos a serem usados para determinar se uma amostra minoritária está em perigo Esta opção só é ativada com os tipos de algoritmo SMOTE borderline1 e borderline2. O padrão é 10.
algorithm_kind Renomeado para algorithm a partir da versão 18.2.1.1	*sequência*	O tipo de algoritmo SMOTE: regular, borderline1 ou borderline2.
usepartition Renomeado para use_partition a partir da versão 18.2.1.1	*Booleano*	Se configurado como true, somente dados de treinamento serão usados para construção de modelo. O patrão é true.

Propriedades tsnenode

t-Distributed Stochastic Neighbor Embedding (t-SNE) é uma ferramenta para visualização de dados de alta dimensionais. Ele converte afinidades de pontos de dados em probabilidades. Este nó t-SNE em SPSS Modeler é implementado em Python e requer a biblioteca scikit-learn© Python .

Tabela 269. propriedades tsnenode		
propriedades tsnenode	**Tipo de dados**	**Descrição da propriedade**
mode_type	*sequência*	Especifique simple ou modo expert .
n_components	*sequência*	Dimensão do espaço integrado (2D ou 3D). Especifique 2 ou 3. O padrão é 2.
method	*sequência*	Especifique barnes_hut ou exact. O padrão é barnes_hut.
init	*sequência*	Inicialização da integração... Especifique random ou pca. O padrão é random.

Tabela 269. propriedades tsnenode (continuação)

propriedades tsnenode	Tipo de dados	Descrição da propriedade
target_field Renomeado para target a partir da versão 18.2.1.1	*sequência*	Nome do campo de destino.. Será um colormap no gráfico de saída. O gráfico usará uma cor se nenhum campo de destino for especificado..
perplexity	*Valor flutuante*	A perplexidade está relacionada ao número de vizinhos mais próximos usados em outros algoritmos de aprendizagem. Conjuntos de dados maiores geralmente requerem uma maior perplexidade. Considere selecionar um valor entre 5 e 50. O padrão é 30.
early_exaggeration	*Valor flutuante*	Controla o quão apertado os clusters naturais no espaço original estão no espaço integrado e quanto espaço haverá entre eles. O padrão é 12.0.
learning_rate	*Valor flutuante*	O padrão é 200.
n_iter	*Número inteiro*	O número máximo de iterações para a otimização Configure como pelo menos 250 O patrão é 1000.
angle	*Valor flutuante*	O tamanho angular do nó distante, conforme medido a partir de um ponto Especifique um valor no intervalo de 0 a 1. O padrão é 0.5.
enable_random_seed	*Booleano*	Configure como true para ativar o parâmetro random_seed O padrão é false.
random_seed	*Número inteiro*	O valor inicial de número aleatório a ser usado O padrão é None.
n_iter_without_progress	*Número inteiro*	Número máximo de iterações sem progresso O padrão é 300.
min_grad_norm	*sequência*	Se a norma gradiente estiver abaixo desse limite, a otimização será interrompida. O padrão é 1.0E-7. Os valores possíveis são: • 1.0E-1 • 1.0E-2 • 1.0E-3 • 1.0E-4 • 1.0E-5 • 1.0E-6 • 1.0E-7 • 1.0E-8
isGridSearch	*Booleano*	Configure como true para executar o t-SNE com várias perplexidades diferentes O patrão é false.

Tabela 269. propriedades tsnenode (continuação)		
propriedades tsnenode	**Tipo de dados**	**Descrição da propriedade**
output_Rename	*Booleano*	Especifique true se você desejar fornecer um nome customizado ou false para nomear a saída automaticamente O padrão é false.
output_to	*sequência*	Especifique Screen ou Output. O padrão é Screen.
full_filename	*sequência*	Especifique o nome do arquivo de saída.
output_file_type	*sequência*	Formato do arquivo de saída.. Especifique HTML ou Output object. O padrão é HTML.

propriedades xgboostlinearnode

XGBoost Linear© é uma implementação avançada de um algoritmo de boosting de gradiente com um modelo linear como o modelo base. Os algoritmos de boosting aprendem iterativamente classificadores fracos e os incluem em um classificador forte final. O Nó XGBoost Linear em SPSS Modeler é implementado em Python.

Tabela 270. propriedades xgboostlinearnode		
propriedades xgboostlinearnode	**Tipo de dados**	**Descrição da propriedade**
TargetField Renomeado para target a partir da versão 18.2.1.1	*campo*	
InputFields Renomeado para inputs a partir da versão 18.2.1.1	*campo*	
alpha	*double*	O parâmetro de intensificador linear alpha. Especifique qualquer número 0 ou maior. O patrão é 0.
lambda	*double*	O parâmetro de intensificador linear lambda. Especifique qualquer número 0 ou maior. O patrão é 1.
lambdaBias	*double*	O parâmetro de intensificador linear de bias lambda. Especifique qualquer número. O padrão é 0.
numBoostRound Renomeado para num_boost_round a partir da versão 18.2.1.1	*Número inteiro*	O valor redondo de impulso num para construção de modelo. Especifique um valor entre 1 e 1000. O padrão é 10.

Tabela 270. propriedades xgboostlinearnode (continuação)		
propriedades xgboostlinearnode	**Tipo de dados**	**Descrição da propriedade**
objectiveType	*sequência*	O tipo de objetivo para a tarefa de aprendizagem. Os valores possíveis são reg:linear, reg:logistic, reg:gamma, reg:tweedie, count:poisson, rank:pairwise, binary:logistic ou multi. Observe que para destinos de sinalização, apenas binary:logistic ou multi podem ser usados. Se multi for usado, o resultado da pontuação mostrará os tipos de objetivos XGBoost multi:softmax e multi:softprob.
random_seed	*Número inteiro*	O valor inicial de número aleatório. Qualquer número entre 0 e 9999999. O padrão é 0.
useHPO	*Booleano*	Especifique true ou false para ativar ou desativar as opções de HPO. Se configurado como true, o Rbfopt será aplicado para descobrir o "melhor" modelo de One-Class SVM automaticamente, que atinge o valor do objetivo de destino definido pelo usuário com o parâmetro target_objval.

propriedades xgboosttreenode

XGBoost Tree© é uma implementação avançada de um algoritmo de boosting de gradiente com um modelo de árvore como o modelo base. Os algoritmos de boosting aprendem iterativamente classificadores fracos e os incluem em um classificador forte final. O XGBoost Tree é muito flexível e fornece muitos parâmetros que podem ser esmagadores para a maioria dos usuários, assim, o nó do XGBoost Tree em SPSS Modeler expõe os principais recursos e parâmetros comumente usados. O nó é implementado em Python.

Tabela 271. propriedades xgboosttreenode		
propriedades xgboosttreenode	**Tipo de dados**	**Descrição da propriedade**
TargetField Renomeado para target a partir da versão 18.2.1.1	*campo*	Os campos de destino.
InputFields Renomeado para inputs a partir da versão 18.2.1.1	*campo*	Os campos de entrada.

Tabela 271. propriedades xgboosttreenode (continuação)

propriedades xgboosttreenode	Tipo de dados	Descrição da propriedade
treeMethod Renomeado para tree_method a partir da versão 18.2.1.1	*sequência*	O método da árvore para construção de modelo. Os valores possíveis são auto, exact ou approx. O padrão é auto.
numBoostRound Renomeado para num_boost_round a partir da versão 18.2.1.1	*Número inteiro*	O valor redondo de impulso num para construção de modelo. Especifique um valor entre 1 e 1000. O padrão é 10.
maxDepth Renomeado para max_depth a partir da versão 18.2.1.1	*Número inteiro*	A profundidade máxima para o crescimento de árvores. Especifique um valor de 1 ou superior. O patrão é 6.
minChildWeight Renomeado para min_child_weight a partir da versão 18.2.1.1	*double*	O peso infantil mínimo para o crescimento de árvores. Especifique um valor de 0 ou superior. O patrão é 1.
maxDeltaStep Renomeado para max_delta_step a partir da versão 18.2.1.1	*double*	A etapa delta máximo para o crescimento de árvores. Especifique um valor de 0 ou superior. O patrão é 0.
objectiveType Renomeado para objective_type a partir da versão 18.2.1.1	*sequência*	O tipo de objetivo para a tarefa de aprendizagem. Os valores possíveis são reg:linear, reg:logistic, reg:gamma, reg:tweedie, count:poisson, rank:pairwise, binary:logistic ou multi. Observe que para destinos de sinalização, apenas binary:logistic ou multi podem ser usados. Se multi for usado, o resultado da pontuação mostrará os tipos de objetivos XGBoost multi:softmax e multi:softprob.
earlyStopping Renomeado para early_stopping a partir da versão 18.2.1.1	*Booleano*	Seja para usar a função de parada antecipada. O padrão é False.
earlyStoppingRounds Renomeado para early_stopping_rounds a partir da versão 18.2.1.1	*Número inteiro*	O erro de validação precisa diminuir pelo menos a cada início(s) de parada antecipada(s) para continuar treinando. O padrão é 10.

Tabela 271. propriedades xgboosttreenode (continuação)

propriedades xgboosttreenode	Tipo de dados	Descrição da propriedade
evaluationDataRatio Renomeado para evaluation_data_ratio a partir da versão 18.2.1.1	*double*	Razão de dados de entrada usados para erros de validação. O padrão é 0.3.
random_seed	*Número inteiro*	O valor inicial de número aleatório. Qualquer número entre 0 e 9999999. O padrão é 0.
sampleSize Renomeado para sample_size a partir da versão 18.2.1.1	*double*	A subamostra para o superajuste de controle. Especifique um valor entre 0.1 e 1.0. O padrão é 0.1.
eta	*double*	A eta para o superajuste de controle. Especifique um valor entre 0 e 1. O padrão é 0.3.
gamma	*double*	A gamma para o superajuste de controle. Especifique qualquer número 0 ou maior. O patrão é 6.
colsSampleRatio Renomeado para col_sample_ratio a partir da versão 18.2.1.1	*double*	A colsample por árvore para o superajuste de controle. Especifique um valor entre 0.01 e 1. O padrão é 1.
colsSampleLevel Renomeado para col_sample_level a partir da versão 18.2.1.1	*double*	A colsample por nível para o superajuste de controle. Especifique um valor entre 0.01 e 1. O padrão é 1.
lambda	*double*	A lambda para o superajuste de controle. Especifique qualquer número 0 ou maior. O patrão é 1.
alpha	*double*	O alpha para o superajuste de controle. Especifique qualquer número 0 ou maior. O patrão é 0.
scalePosWeight Renomeado para scale_pos_weight a partir da versão 18.2.1.1	*double*	O peso pos escala para manipulação de conjuntos de dados desbalanceados. O padrão é 1.
use_HPO Incluído na versão 18.2.1.1		

Capítulo 20. Propriedades do nó do Spark

propriedades isotonicasnode

A Regressão Isotônica pertence à família de algoritmos de regressão. O nó Isotonic-AS em SPSS Modeler é implementado em Spark. Para obter detalhes sobre algoritmos de Regressão isotônica, consulte https://spark.apache.org/docs/2.2.0/mllib-isotonic-regression.html.

Tabela 272. propriedades isotonicasnode

propriedades isotonicasnode	Tipo de dados	Descrição da propriedade
label	*sequência*	Esta propriedade é uma variável dependente para a qual a regressão isotônica é calculada.
features	*sequência*	Esta propriedade é uma variável independente.
weightCol	*sequência*	O peso representa uma série de medidas. O padrão é 1.
isotonic	*Booleano*	Esta propriedade indica se o tipo é isotonic ou antitonic.
featureIndex	*Número inteiro*	Esta propriedade é para o índice do recurso se featuresCol é uma coluna vetorial. O patrão é 0.

propriedades kmeansasnode

K-Médias é um dos algoritmos de armazenamento em cluster mais comumente usados. Ele armazena em cluster pontos de dados em um número predefinido de clusters. O nó K-Means-AS no SPSS Modeler é implementado no Spark. Para obter detalhes sobre algoritmos K-Means, consulte https://spark.apache.org/docs/2.2.0/ml-clustering.html. Observe que o nó K-Means-AS executa a codificação one-hot automaticamente para variáveis categóricas.

Tabela 273. propriedades kmeansasnode

Propriedades kmeansasnode	Valores	Descrição da propriedade
roleUse	*sequência*	Especifique predefined para usar funções predefinidas ou custom para usar designações de campo customizadas. O padrão é predefined.
autoModel	*Booleano*	Especifique true para usar o nome padrão ($S-prediction) para o novo campo de pontuação gerado ou false para usar um nome customizado. O padrão é true.

Tabela 273. propriedades kmeansasnode (continuação)		
Propriedades kmeansasnode	**Valores**	**Descrição da propriedade**
features	*campo*	Lista dos nomes de campo para entrada quando a propriedade roleUse é configurada como custom.
name	*sequência*	O nome do novo campo de pontuação do gerado quando a propriedade autoModel é configurada como false.
clustersNum	*Número inteiro*	O número de clusters a serem criados. O padrão é 5.
initMode	*sequência*	O algoritmo de inicialização. Os valores possíveis são k-means|| ou random. O padrão é k-means||.
initSteps	*Número inteiro*	O número de etapas de inicialização quando initMode é configurado como k-means||. O padrão é 2.
advancedSettings	*Booleano*	Especifique true para disponibilizar as quatro propriedades a seguir. O patrão é false.
maxIteration	*Número inteiro*	Número máximo de iterações para clusterização. O padrão é 20.
tolerance	*sequência*	A tolerância para parar as iterações. As configurações possíveis são 1.0E-1, 1.0E-2, ..., 1.0E-6. O padrão é 1.0E-4.
setSeed	*Booleano*	Especifique true para usar um valor inicial aleatório customizado. O patrão é false.
randomSeed	*Número inteiro*	O valor inicial aleatório customizado quando a propriedade setSeed é true.

propriedades multilayerperceptronnode

Perceptron multicamadas é um classificador baseado na rede neural artificial feedforward e consiste em múltiplas camadas. Cada camada está totalmente conectada à próxima camada na rede. O nó do Perceptron-AS MultiLayerno SPSS Modeler é implementado no Spark Para obter detalhes sobre o multilayer perceptron classifier (MLPC), consulte https://spark.apache.org/docs/latest/ml-classification-regression.html#multilayer-perceptron-classifier.

Tabela 274. propriedades multilayerperceptronnode		
propriedades multilayerperceptronnode	**Tipo de dados**	**Descrição da propriedade**
features	*campo*	Um ou mais campos a serem usados como entradas para a predição

Tabela 274. propriedades multilayerperceptronnode (continuação)		
propriedades multilayerperceptronnode	**Tipo de dados**	**Descrição da propriedade**
label	*campo*	O campo a ser usado como o destino para a predição
layers[0]	*Número inteiro*	O número de camadas de perceptron a serem incluídas O padrão é 1.
layers[1…<latest-1>]	*Número inteiro*	O número de camadas ocultas.. O padrão é 1.
layers[<latest>]	*Número inteiro*	O número de camadas de saída. O padrão é 1.
seed	*Número inteiro*	O valor inicial aleatório customizado..
maxiter	*Número inteiro*	O número máximo de iterações a serem executadas O padrão é 10.

propriedades xgboostasnode

XGBoost é uma implementação avançada de um algoritmo de boosting de gradiente. Os algoritmos de boosting aprendem iterativamente classificadores fracos e os incluem em um classificador forte final. O XGBoost é muito flexível e fornece muitos parâmetros que podem ser opressores para a maioria dos usuários, portanto, o nó XGBoost-AS no SPSS Modeler expõe os recursos principais e os parâmetros comumente usados. O nó XGBoost-AS é implementado no Spark.

Tabela 275. propriedades xgboostasnode		
propriedades xgboostasnode	**Tipo de dados**	**Descrição da propriedade**
target_field	*campo*	Lista dos nomes de campo para destino.
input_fields	*campo*	Lista dos nomes de campo para entradas.
nWorkers	*Número inteiro*	O número de trabalhadores usados para treinar o modelo XGBoost. O padrão é 1.
numThreadPerTask	*Número inteiro*	O número de encadeamentos usados por trabalhador. O padrão é 1.
useExternalMemory	*Booleano*	Indica se deve usar a memória externa como cache. O padrão é **false**
boosterType	*sequência*	O tipo de intensificador a ser usado. As opções disponíveis são gbtree, gblinear ou dart. O padrão é gbtree.
numBoostRound	*Número inteiro*	O número de rodadas para boosting. Especifique um valor de 0 ou superior. O patrão é 10.
scalePosWeight	*double*	Controle o balanceamento de ponderações positivas e negativas. O padrão é 1.
randomseed	*Número inteiro*	O valor inicial usado pelo gerador de número aleatório. O padrão é **0**.

Tabela 275. propriedades xgboostasnode (continuação)

propriedades xgboostasnode	Tipo de dados	Descrição da propriedade
objectiveType	*sequência*	O objetivo de aprendizagem. Os valores possíveis sãoreg:linear, reg:logistic, reg:gamma, reg:tweedie,rank:pairwise, binary:logistic ou multi. Observe que para destinos de sinalização, apenas binary:logistic ou multi podem ser usados. Se multi for usado, o resultado da pontuação mostrará os tipos de objetivos XGBoost multi:softmax e multi:softprob. O padrão é reg:linear.
evalMetric	*sequência*	As métricas de avaliação para dados de validação. Uma métrica padrão será atribuída de acordo com o objetivo. Os valores possíveis são rmse, mae, logloss, error, merror, mlogloss, auc, ndcg, map ou gamma-deviance. O padrão é rmse.
lambda	*double*	Termo de regularização L2 em ponderações. Aumentar esse valor tornará o modelo mais conservador. Especifique qualquer número 0 ou maior. O patrão é 1.
alpha	*double*	Termo de regularização L1 sobre pesos. Aumentar esse valor tornará o modelo mais conservador. Especifique qualquer número 0 ou maior. O patrão é 0.
lambdaBias	*double*	Termo de regularização L2 no viés. Se o tipo de intensificador gblinear for usado, esse parâmetro de intensificador linear de bias lambda estará disponível. Especifique qualquer número 0 ou maior. O padrão é 0.
treeMethod	*sequência*	Se o tipo de intensificador gbtree ou dart for usado, esse parâmetro de método de árvore para crescimento de árvore (e os outros parâmetros de árvore que se seguem) estará disponível. Ele especifica o algoritmo de construção de árvore XGBoost a ser usado. As opções disponíveis são auto, exact ou approx. O padrão é auto.
maxDepth	*Número inteiro*	A profundidade máxima para as árvores. Especifique um valor de 2 ou superior. O patrão é 6.
minChildWeight	*double*	A soma mínima de peso da instância (hessiana) necessária em uma criança. Especifique um valor de 0 ou superior. O patrão é 1.

Tabela 275. propriedades xgboostasnode (continuação)

propriedades xgboostasnode	Tipo de dados	Descrição da propriedade
maxDeltaStep	*double*	A etapa de delta máxima para permitir a estimativa de peso de cada árvore. Especifique um valor de 0 ou superior. O patrão é 0.
sampleSize	*double*	A subamostra é a razão da instância de treinamento. Especifique um valor entre 0.1 e 1.0. O padrão é 1.0.
eta	*double*	A redução do tamanho da etapa usada durante a etapa de atualização para evitar o superajuste. Especifique um valor entre 0 e 1. O padrão é 0.3.
gamma	*double*	A redução de perda mínima necessária para fazer uma partição adicional em um nó folha da árvore. Especifique qualquer número 0 ou maior. O patrão é 6.
colsSampleRatio	*double*	A razão de subamostra de colunas ao construir cada árvore. Especifique um valor entre 0.01 e 1. O padrão é 1.
colsSampleLevel	*double*	A proporção de subamostra de colunas para cada divisão, em cada nível. Especifique um valor entre 0.01 e 1. O padrão é 1.
normalizeType	*sequência*	Se o tipo de intensificador DART for usado, esse parâmetro dart e os três parâmetros de dart a seguir estarão disponíveis. Esse parâmetro configura o algoritmo de normalização. Especifique tree ou forest. O padrão é tree.
sampleType	*sequência*	O tipo de algoritmo de amostragem. Especifique uniform ou weighted. O padrão é uniform.
rateDrop	*double*	O parâmetro do intensificador DART de taxa de dropout. Especifique um valor entre 0.0 e 1.0. O padrão é 0.0.
skipDrop	*double*	O parâmetro de intensificador DART para a probabilidade de ignorar dropout. Especifique um valor entre 0.0 e 1.0. O padrão é 0.0.

Capítulo 21. Propriedades do Supernó

As propriedades que são específicas para SuperNodes estão descritas nas tabelas a seguir. Observe que as propriedades do nó comum também se aplicam a SuperNodes.

Tabela 276. Propriedades de supernó de terminal

Nome da propriedade	Tipo de Propriedade /Lista de valores	Descrição da propriedade

`execute_method`	`Script`
	
	`Normal`	

`script`	*sequência*	

Parâmetros de SuperNode

É possível utilizar scripts para criar ou configurar parâmetros de SuperNode usando o formato geral:

```
mySuperNode.setParameterValue("minvalue", 30)
```

É possível recuperar o valor do parâmetro com:

```
value mySuperNode.getParameterValue("minvalue")
```

Localizando SuperNodes Existentes

É possível localizar SuperNodes em fluxos usando a função findByType():

```
source_supernode = modeler.script.stream().findByType("source_super", None)
process_supernode = modeler.script.stream().findByType("process_super", None)
terminal_supernode = modeler.script.stream().findByType("terminal_super",
None)
```

Configurando Propriedades para Nós Encapsulados

É possível configurar propriedades para nós específicos encapsulados em um SuperNode ao acessar o diagrama filho dentro do SuperNode. Por exemplo, suponha que você tenha um SuperNode de origem com um nó Arquivo Variável encapsulado para leitura nos dados. É possível transmitir o nome do arquivo a ser lido (especificado utilizando a propriedade `full_filename`) ao acessar o diagrama filho e localizar o nó relevante, conforme a seguir:

```
childDiagram = source_supernode.getChildDiagram()
varfilenode = childDiagram.findByType("variablefile", None)
varfilenode.setPropertyValue("full_filename", "c:/mydata.txt")
```

Criando Supernós

Se desejar criar um SuperNode e seu conteúdo desde o início, será possível fazer isso de forma semelhante ao criar o SuperNode, acessar o diagrama filho e criar os nós que desejar. Assegure-se

também de que os nós no diagrama de SuperNode também estejam vinculados aos nós do conector de entrada e/ou de saída. Por exemplo, se desejar criar um SuperNode de processo:

```
process_supernode = modeler.script.stream().createAt("process_super", "My
SuperNode", 200, 200)
childDiagram = process_supernode.getChildDiagram()
filternode = childDiagram.createAt("filter", "My Filter", 100, 100)
childDiagram.linkFromInputConnector(filternode)
childDiagram.linkToOutputConnector(filternode)
```

Apêndice A. Referência de nomes de nós

Esta seção fornece uma referência para os nomes de script dos nós no IBM SPSS Modeler.

Nomes do Nugget do Modelo

Os nuggets do modelo (também conhecidos como modelos gerados) podem ser referidos por tipo, assim como os objetos de nó e de saída. As tabelas a seguir listam os nomes de referência do objeto modelo.

Observe que esses nomes são utilizados especificamente para referenciar nuggets do modelo na paleta Modelos (no canto superior direito da janela do IBM SPSS Modeler). Para nós de modelo de referência que foram incluídos em um fluxo para fins de pontuação, um conjunto diferente de nomes prefixados com `apply...` é utilizado.

Nota: Sob circunstâncias normais, referenciar modelos por nome *e* também por tipo é recomendado para evitar confusão.

Tabela 277. Nomes do Nugget do Modelo (Paleta de Modelagem)

Nome do modelo	**Modelar**
anomalydetection	Anomalia
a priori	A priori
autoclassifier	Classificador Automático
autocluster	Cluster automático
autonumeric	Numeração Automática
bayesnet	rede bayesiana
c50	C5.0
carma	Carma
carrinho	Árvore C&R
chaid	CHAID
coxreg	regressão de Cox
decisionlist	Lista de Decisão
discriminante	Discriminante
fator	PCA/Fator
featureselection	Seleção de Variáveis
genlin	Regressão linear generalizada
glmm	GLMM
kmeans	K-Médias
knn	vizinho *k* mais próximo
kohonen	Kohonen
linear	Linear
logreg	Regressão logística

Tabela 277. Nomes do Nugget do Modelo (Paleta de Modelagem) (continuação)	
Nome do modelo	**Modelar**
neuralnetwork	Rede neural
quest	QUEST
regressão	Regressão linear
sequência	Sequência
slrm	Modelo de resposta de autoaprendizado
statisticsmodel	Modelo do IBM SPSS Statistics
svm	Support Vector Machine
série temporal	Séries temporais
twostep	TwoStep

Tabela 278. Nomes do Nugget do Modelo (Paleta de Modelagem de Banco de Dados)	
Nome do modelo	**Modelar**
db2imcluster	Armazenamento em Cluster do IBM ISW
db2imlog	Regressão Logística do IBM ISW
db2imnb	IBM ISW Naive Bayes
db2imreg	Regressão do IBM ISW
db2imtree	Árvore de Decisão do IBM ISW
msassoc	Regras de associação da MS
msbayes	Naive Bayes da MS
mscluster	Cluster MS
mslogistic	Regressão logística da MS
msneuralnetwork	Rede Neural da MS
msregression	Regressão linear da MS
mssequencecluster	Cluster de Sequências da MS
mstimeseries	Séries temporais da MS
mstree	Árvore de decisão MS
netezzabayes	Rede bayesiana Netezza
netezzadectree	Árvore de decisão Netezza
netezzadivcluster	Cluster de divisão Netezza
netezzaglm	Modelo linear generalizado Netezza
netezzakmeans	K-Médias Netezza
netezzaknn	KNN Netezza
netezzalineregression	Regressão linear Netezza
netezzanaivebayes	Naive Bayes Netezza
netezzapca	PCA Netezza

Tabela 278. Nomes do Nugget do Modelo (Paleta de Modelagem de Banco de Dados) (continuação)	
Nome do modelo	**Modelar**
netezzaregtree	Árvore de regressão Netezza
netezzatimeseries	Séries temporais Netezza
oraabn	Oracle Adaptive Bayes
oraai	AI da Oracle
oradecisiontree	Árvore de decisão da Oracle
oraglm	GLM da Oracle
orakmeans	*k*-Médias da Oracle
oranb	Oracle Naive Bayes
oranmf	NMF da Oracle
oraocluster	O-Cluster Oracle
orasvm	Oracle SVM

Evitando Nomes de Modelos Duplicados

Quando utilizar scripts para manipular modelos gerados, lembre-se de que permitir nomes de modelos duplicados pode resultar em referências ambíguas. Para evitar isso, recomenda-se requerer nomes exclusivos para modelos gerados ao executar script.

Para configurar opções para nomes de modelo duplicados:

1. No menu, escolha:

   Ferramentas > **Opções do usuário**
2. Clique na tarefa **Notificações**.
3. Selecione **Substituir modelo anterior** para restringir nomenclatura duplicada para modelos gerados.

O comportamento da execução do script pode variar entre o SPSS Modeler e o IBM SPSS Collaboration and Deployment Services quando houver referências de modelo ambíguas. O cliente do SPSS Modeler inclui a opção "Substituir modelo anterior" que substitui automaticamente os modelos que tiverem o mesmo nome (por exemplo, onde um script iterar através de um loop para produzir um modelo diferente todas as vezes). No entanto, essa opção não está disponível quando o mesmo script for executado no IBM SPSS Collaboration and Deployment Services. É possível evitar esta situação renomeando o modelo gerado em cada iteração para evitar referências ambíguas aos modelos ou limpando o modelo atual (por exemplo, incluir uma instrução `clear generated palette`) antes do término do loop.

Nomes do Tipo de Saída

A tabela a seguir lista todos os tipos de objetos de saída e os nós que criam esses objetos.

Tabela 279. Tipos de objetos de saída e os nós que criam esses objetos.	
Tipo de objeto de saída	**Nó**
analysisoutput	Análise
collectionoutput	Coleção
dataauditoutput	Auditoria de dados
distributionoutput	Distribuição

Tabela 279. Tipos de objetos de saída e os nós que criam esses objetos. (continuação)	
Tipo de objeto de saída	**Nó**
evaluationoutput	Avaliação
histogramoutput	Histograma
matrixoutput	Matriz
meansoutput	Médias
multiplotoutput	Multigráficos
plotoutput	Gráfico
qualityoutput	Qualidade
reportdocumentoutput	Este tipo de objeto não é de um nó, é a saída criada por um relatório do projeto
reportoutput	Veja o relatório
statisticsprocedureoutput	Estatísticassaída
statisticsoutput	Estatísticas
tableoutput	Tabela
timeplotoutput	Gráfico de tempo
weboutput	Web

Apêndice B. Migrando do script legado para o script Pythong

Visão geral de migração de script de legado

Esta seção fornece um resumo das diferenças entre script Python e script legado no IBM SPSS Modeler e fornece informações sobre como migrar seus scripts legados para scripts Python. Nesta seção você encontrará uma lista de comandos legados padrão do SPSS Modeler e os comandos Python equivalentes.

Diferenças gerais

O script legado deve muito de seu design aos scripts de comando do S.O. O script de legado é orientado por linha e, embora haja algumas estruturas de bloco, por exemplo, `if...then...else...endif` e `for...endfor`, a indentação geralmente não é significativa.

No script Python, a indentação é significativa e as linhas pertencentes ao mesmo bloco lógico devem ser indentadas pelo mesmo nível.

Nota: É necessário ter atenção ao copiar e colar o código Python. Uma linha que é indentada utilizando guias pode parecer igual no editor a uma linha que é indentada utilizando espaços. No entanto, o script Python gerará um erro porque as linhas não são consideradas como igualmente indentadas.

O contexto de script

O contexto de script define o ambiente no qual o script está sendo executado, por exemplo, o fluxo ou SuperNode que executa o script. No script legado, o contexto é implícito, o que significa, por exemplo, que todas as referências de nó em um script de fluxo são assumidas como estando dentro do fluxo que executa o script.

No script Python, o contexto de script é fornecido explicitamente por meio do módulo `modeler.script`. Por exemplo, um script de fluxo Python pode acessar o fluxo que executa o script com o código a seguir:

```
s = modeler.script.stream()
```

Em seguida, as funções relacionadas ao fluxo podem ser chamadas por meio do objeto retornado.

Comandos e funções

O script legado é orientado a comando. Isso significa que cada linha de script geralmente inicia com o comando a ser executado seguido pelos parâmetros, por exemplo:

```
connect 'Type':typenode to :filternode
rename :derivenode as "Compute Total"
```

O Python utiliza funções que normalmente são chamadas por meio de um objeto (um módulo, classe ou objeto) que define a função, por exemplo:

```
stream = modeler.script.stream()
typenode = stream.findByType("type", "Type)
filternode = stream.findByType("filter", None)
stream.link(typenode, filternode)
derive.setLabel("Compute Total")
```

Literais e comentários

Alguns comandos literais e de comentário que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 280. Mapeamento de script legado para script Python para literais e comentários

Script anterior	**Script Python**
Número Inteiro, por exemplo, 4	Mesmo
Flutuação, por exemplo, `0.003`	Mesmo
Sequências entre aspas simples, por exemplo, `'Hello'`	Mesmo **Nota:** Os literais de sequência que contiverem caracteres não ASCII devem ser prefixados por um u para assegurar que eles sejam representados como Unicode.
Sequências de aspas duplas, por exemplo, `"Hello again"`	Mesmo **Nota:** Os literais de sequência que contiverem caracteres não ASCII devem ser prefixados por um u para assegurar que eles sejam representados como Unicode.
Sequências longas, por exemplo, `"""This is a string that spans multiple lines"""`	Mesmo
Listas, por exemplo, `[1 2 3]`	`[1, 2, 3]`
Referência de variável, por exemplo, `set x = 3`	`x = 3`
Continuação de linha (\), por exemplo, `set x = [1 2 \ 3 4]`	`x = [ 1, 2,\ 3, 4]`
Comentário de bloco, por exemplo, `/* This is a long comment over a line. */`	`""" This is a long comment over a line. """`
Comentário de linha, por exemplo `set x = 3 # make x 3`	`x = 3 # make x 3`
`undef`	`None`
`true`	`True`
`false`	`False`

Operadores

Alguns comandos do operador que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 281. Mapeamento de script legado para script Python para operadores

Script anterior	Script Python

`NUM1 + NUM2`
`LIST + ITEM`
`LIST1 + LIST2`
	`NUM1 + NUM2`
	`LIST.append(ITEM)`
	`LIST1.extend(LIST2)`

`NUM1 - NUM2` `LIST - ITEM`	`NUM1 - NUM2` `LIST.remove(ITEM)`
`NUM1 * NUM2`	`NUM1 * NUM2`
`NUM1 / NUM2`	`NUM1 / NUM2`
`=` `==`	`==`
`/=` `/==`	`!=`
`X ** Y`	`X ** Y`

`X < Y`
`X <= Y`
`X > Y`
`X >= Y`
	`X < Y`
	`X <= Y`
	`X > Y`
	`X >= Y`

`X div Y`
`X rem Y`
`X mod Y`
	`X // Y`
	`X % Y`
	`X % Y`

`and`
`or`
`not(EXPR)`
	`and`
	`or`
	`not EXPR`

Condicionais e Loop

Alguns comandos condicionais e de loop que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 282. Mapeamento de script legado para script Python para condicionais e loop

Script anterior	Script Python

`for VAR from INT1 to INT2`
`  …`
`endfor`
	`for VAR in range(INT1, INT2):`
	`  …`
	
	ou
	
	`VAR = INT1`
	`while VAR <= INT2:`
	`  ...`
	`  VAR += 1`

`for VAR in LIST`
`  …`
`endfor`
	`for VAR in LIST:`
	`  …`

Tabela 282. Mapeamento de script legado para script Python para condicionais e loop (continuação)

Script anterior	Script Python

`for VAR in_fields_to NODE`
`  …`
`endfor`
	`for VAR in NODE.getInputDataModel():`
	`  ...`

`for VAR in_fields_at NODE`
`  …`
`endfor`
	`for VAR in NODE.getOutputDataModel():`
	`  ...`

`if…then`
`  …`
`elseif…then`
`  …`
`else`
`  …`
`endif`
	`if …:`
	`  …`
	`elif …:`
	`  …`
	`else:`
	`  …`

`with TYPE OBJECT`
`  …`
`endwith`
	Sem equivalente

`var VAR1`	A declaração de variável não é necessária

Variáveis

No script legado, as variáveis são declaradas antes de serem referenciadas, por exemplo:

```
var mynode
set mynode = create typenode at 96 96
```

No Python script, as variáveis são criadas quando forem referenciadas pela primeira vez, por exemplo:

```
mynode = stream.createAt("type", "Type", 96, 96)
```

No script legado, as referências a variáveis devem ser explicitamente removidas utilizando o operador ^, por exemplo:

```
var mynode
set mynode = create typenode at 96 96
set ^mynode.direction."Age" = Input
```

Assim como acontece a maioria das linguagens de script, isto não é necessário no script Python, por exemplo:

```
mynode = stream.createAt("type", "Type", 96, 96)
mynode.setKeyedPropertyValue("direction","Age","Input")
```

Tipos de nó, de saída e de modelo

No script legado, os tipos de objeto diferentes (nó, saída e modelo) normalmente têm o tipo anexado ao tipo de objeto. Por exemplo, o nó Derivar tem o tipo de `derivenode`:

```
set feature_name_node = create derivenode at 96 96
```

Como a API do IBM SPSS Modeler em Python não inclui o sufixo node, o nó Derivar possui o tipo `derive`, por exemplo:

```
feature_name_node = stream.createAt("derive", "Feature", 96, 96)
```

A única diferença nos nomes de tipo no script legado e Python é a ausência do sufixo do tipo.

Nomes de propriedades

Os nomes de propriedade são os mesmos nos scripts legado e Pynthon. Por exemplo, no nó Arquivo Variável, a propriedade que define o local do arquivo é `full_filename` em ambos os ambientes de script.

Referências do Nó

Muitos scripts anteriores utilizam uma procura implícita para localizar e acessar o nó a ser modificado. Por exemplo, os comandos a seguir procuram o fluxo atual para um nó Tipo com o rótulo "Type" e, em seguida, configuram a direção (ou função de modelagem) do campo "Age" para Input e o campo "Drug" para ser o Target, que é o valor a ser previsto:

```
set 'Type':typenode.direction."Age" = Input
set 'Type':typenode.direction."Drug" = Target
```

No script Python, os objetos de nó devem ser localizados explicitamente antes de chamar a função para configurar o valor da propriedade, por exemplo:

```
typenode = stream.findByType("type", "Type")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
typenode.setKeyedPropertyValue("direction", "Drug", "Target")
```

Nota: Neste caso, "Target" deve estar entre aspas da sequência.

Os scripts Python podem utilizar como alternativa a enumeração `ModelingRole` no pacote `modeler.api`.

Embora a versão de script Python seja mais detalhada, ela proporciona um melhor desempenho de tempo de execução porque a procura para o nó geralmente é feita apenas uma vez. No exemplo de script legado, a procura do nó é feita para cada comando.

Localizar nós por ID também é suportado (o ID do nó é visível na guia Anotações do diálogo de nó). Por exemplo, em scripts anteriores:

```
# id65EMPB9VL87 is the ID of a Type node
set @id65EMPB9VL87.direction."Age" = Input
```

O script a seguir mostra o mesmo exemplo no script Python:

```
typenode = stream.findByID("id65EMPB9VL87")
typenode.setKeyedPropertyValue("direction", "Age", "Input")
```

Obtendo e configurando propriedades

O script legado utiliza o comando `set` para designar um valor. O termo após o comando `set` pode ser uma definição de propriedade. O script a seguir mostra dois formatos de script possíveis para configurar uma propriedade:

```
set <node reference>.<property> = <value>
set <node reference>.<keyed-property>.<key> = <value>
```

No script Python, o mesmo resultado é obtido utilizando as funções `setPropertyValue()` e `setKeyedPropertyValue()`, por exemplo:

```
object.setPropertyValue(property, value)
object.setKeyedPropertyValue(keyed-property, key, value)
```

No script legado, o acesso aos valores da propriedade pode ser obtido usando o comando `get`, por exemplo:

```
var n v
set n = get node :filternode
set v = ^n.name
```

No script Python, o mesmo resultado é obtido utilizando a função `getPropertyValue()`, por exemplo:

```
n = stream.findByType("filter", None)
v = n.getPropertyValue("name")
```

Editando fluxos

No script legado, o comando `create` é utilizado para criar um novo nó, por exemplo:

```
var agg select
set agg = create aggregatenode at 96 96
set select = create selectnode at 164 96
```

No script Python, os fluxos possuem vários métodos para a criação de nós, por exemplo:

```
stream = modeler.script.stream()
agg = stream.createAt("aggregate", "Aggregate", 96, 96)
select = stream.createAt("select", "Select", 164, 96)
```

No script legado, o comando `connect` é utilizado para criar links entre os nós, por exemplo:

```
connect ^agg to ^select
```

No script Python, o método `link` é utilizado para criar links entre os nós, por exemplo:

```
stream.link(agg, select)
```

No script legado, o comando `disconnect` é utilizado para remover links entre os nós, por exemplo:

```
disconnect ^agg from ^select
```

No script Python, o método `unlink` é utilizado para remover links entre os nós, por exemplo:

```
stream.unlink(agg, select)
```

No script legado, o comando `position` é utilizado para posicionar os nós na tela do fluxo ou entre outros nós, por exemplo:

```
position ^agg at 256 256
position ^agg between ^myselect and ^mydistinct
```

No script Python, o mesmo resultado é obtido utilizando dois métodos separados: `setXYPosition` e `setPositionBetween`. Por exemplo:

```
agg.setXYPosition(256, 256)
agg.setPositionBetween(myselect, mydistinct)
```

Operações do nó

Alguns comandos de operação do nó que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 283. Mapeamento de script legado para script Python para operações do nó

Script anterior	Script Python

`create nodespec at x y`	`stream.create(type, name)`
	`stream.createAt(type, name, x, y)`
	`stream.createBetween(type, name, preNode, postNode)`
	`stream.createModelApplier(model, name)`

`connect fromNode to toNode`	`stream.link(fromNode, toNode)`
`delete node`	`stream.delete(node)`
`disable node`	`stream.setEnabled(node, False)`
`enable node`	`stream.setEnabled(node, True)`
`disconnect fromNode from toNode`	`stream.unlink(fromNode, toNode)` `stream.disconnect(node)`
`duplicate node`	`node.duplicate()`
`execute node`	`stream.runSelected(nodes, results)` `stream.runAll(results)`
`flush node`	`node.flushCache()`
`position node at x y`	`node.setXYPosition(x, y)`
`position node between node1 and node2`	`node.setPositionBetween(node1, node2)`
`rename node as name`	`node.setLabel(name)`

Looping

No script legado, há duas opções de loop principais que são suportadas:

- Loops *Contados*, em que uma variável de índice se move entre dois limites de número inteiro.
- Loops de *Sequência* que executam loop através de uma sequência de valores, ligando o valor atual à variável de loop.

O script a seguir é um exemplo de um loop contado no script legado:

```
for i from 1 to 10
    println ^i
endfor
```

O script a seguir é um exemplo de um loop de sequência no script legado:

```
var items
set items = [a b c d]

for i in items
    println ^i
endfor
```

Também há outros tipos de loops que podem ser usados:

- Iterando através dos modelos na paleta de modelos ou através das saídas na paleta de saídas.
- Iterando através dos campos que entram ou que saem de um nó.

O script Python também suporta diferentes tipos de loops: O script a seguir é um exemplo de um loop contado no script Python:

```
i = 1
while i <= 10:
    print i
    i += 1
```

O script a seguir é um exemplo de um loop de sequência no script Python:

```
items = ["a", "b", "c", "d"]
for i in items:
    print i
```

O loop de sequência é muito flexível e, quando combinado com os métodos da API do IBM SPSS Modeler, pode suportar a maioria dos casos de uso de script legado. O exemplo a seguir mostra como usar um loop de sequência no script Python para iterar nos campos que saem de um nó:

```
node = modeler.script.stream().findByType("filter", None)
for column in node.getOutputDataModel().columnIterator():
    print column.getColumnName()
```

Executando fluxos

Durante a execução de fluxo, os objetos de modelo ou de saída que são gerados são incluídos em um dos gerenciadores de objeto. No script legado, o script deve localizar os objetos construídos a partir do gerenciador de objeto ou acessar a saída gerada mais recentemente do nó que gerou a saída.

A execução de fluxo no Python é diferente, em que quaisquer objetos de modelo ou de saída que são gerados a partir da execução são retornados em uma lista que é transmitida para a função de execução. Isso facilita o acesso aos resultados da execução de fluxo.

O script legado suporta três comandos de execução de fluxo:

- `execute_all` executa todos os nós terminais executáveis no fluxo.
- `execute_script` executa o script de fluxo, independentemente da configuração da execução do script.
- `execute` *`node`* executa o nó especificado.

O script Python suporta um conjunto semelhante de funções:

- *`stream`*`.runAll(`*`results-list`*`)` executa todos os nós terminais executáveis no fluxo.
- *`stream`*`.runScript(`*`results-list`*`)` executa o script de fluxo, independentemente da configuração da execução do script.
- *`stream`*`.runSelected(`*`node-array, results-list`*`)` executa o conjunto de nós especificado na ordem em que eles são fornecidos.
- *`node`*`.run(`*`results-list`*`)` executa o nó especificado.

No script legado, uma execução de fluxo pode ser encerrada utilizando o comando `exit` com um código de número inteiro opcional, por exemplo:

```
exit 1
```

No script Python, o mesmo resultado pode ser obtido com o script a seguir:

```
modeler.script.exit(1)
```

Acessando objetos por meio do sistema de arquivos e do repositório

No script legado, é possível abrir um fluxo, modelo ou objeto de saída existente utilizando o comando open, por exemplo:

```
var s
set s = open stream "c:/my streams/modeling.str"
```

No script Python, existe a classe TaskRunner que é acessível a partir da sessão e pode ser utilizada para executar tarefas semelhantes, por exemplo:

```
taskrunner = modeler.script.session().getTaskRunner()
s = taskrunner.openStreamFromFile("c:/my streams/modeling.str", True)
```

Para salvar um objeto no script anterior, é possível usar o comando save , por exemplo:

```
save stream s as "c:/my streams/new_modeling.str"
```

A abordagem script Python equivalente seria utilizar a classe TaskRunner, por exemplo:

```
taskrunner.saveStreamToFile(s, "c:/my streams/new_modeling.str")
```

As operações baseadas em legado do IBM SPSS Collaboration and Deployment Services Repository são suportadas no script legado por meio dos comandos retrieve e store, por exemplo:

```
var s
set s = retrieve stream "/my repository folder/my_stream.str"
store stream ^s as "/my repository folder/my_stream_copy.str"
```

No script Python, a funcionalidade equivalente seria acessada por meio do objeto Repositório que está associado à sessão, por exemplo:

```
session = modeler.script.session()
repo = session.getRepository()
s = repo.retrieveStream("/my repository folder/my_stream.str", None, None, True)
repo.storeStream(s, "/my repository folder/my_stream_copy.str", None)
```

Nota: O acesso do Repositório requer que a sessão tenha sido configurada com uma conexão do repositório válida.

Operações de fluxo

Alguns comandos de operação de fluxo que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 284. Mapeamento de script legado para script Python para operações de fluxo

Script anterior	**Script Python**
`create stream` *DEFAULT_FILENAME*	*taskrunner*`.createStream(`*name*`,` *autoConnect*`,` *autoManage*`)`
`close stream`	*stream*`.close()`
`clear stream`	*stream*`.clear()`
`get stream` *stream*	Sem equivalente
`load stream` *path*	Sem equivalente
`open stream` *path*	*taskrunner*`.openStreamFromFile(`*path*`,` *autoManage*`)`

Tabela 284. Mapeamento de script legado para script Python para operações de fluxo (continuação)	
Script anterior	**Script Python**
`save stream as path`	`taskrunner.saveStreamToFile(stream, path)`
`retreive stream path`	`repository.retreiveStream(path, version, label, autoManage)`
`store stream as path`	`repository.storeStream(stream, path, label)`

Operações de modelo

Alguns comandos de operação de modelo que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 285. Mapeamento de script legado para script Python para operações de modelo	
Script anterior	**Script Python**
`open model path`	`taskrunner.openModelFromFile(path, autoManage)`
`save model as path`	`taskrunner.saveModelToFile(model, path)`
`retrieve model path`	`repository.retrieveModel(path, version, label, autoManage)`
`store model as path`	`repository.storeModel(model, path, label)`

Operações de saída do documento

Alguns comandos de operação de saída de documento que são normalmente utilizados no IBM SPSS Modeler possuem comandos equivalentes no script Python. Isso pode ajudar a converter seus scripts SPSS Modeler Legacy em scripts Python para uso em IBM SPSS Modeler 17.

Tabela 286. Mapeamento de script legado para script Python para operações de saída de documento	
Script anterior	**Script Python**
`open output path`	`taskrunner.openDocumentFromFile(path, autoManage)`
`save output as path`	`taskrunner.saveDocumentToFile(output, path)`
`retrieve output path`	`repository.retrieveDocument(path, version, label, autoManage)`
`store output as path`	`repository.storeDocument(output, path, label)`

Outras diferenças entre script legado e script Python

Os scripts legados fornecem suporte para manipular projetos do IBM SPSS Modeler. No entanto, o script Python não suporta isso no momento.

O script legado fornece algum suporte para carregar objetos de *estado* (combinações de fluxos e de modelos). Os objetos de estado foram descontinuados desde o IBM SPSS Modeler 8.0. O script Python não suporta objetos de estado.

O script Python oferece os recursos adicionais a seguir que não estão disponíveis no script legado:

- Definições de classe e de função
- Manipulação de erros
- Suporte para entrada/saída mais sofisticado
- Módulos externos e de terceiros

Avisos

Estas informações foram desenvolvidas para os produtos e serviços oferecidos nos EUA. Este material pode estar disponível pela IBM em outros idioma. No entanto, pode ser necessário possuir uma cópia do produto ou da versão do produto no mesmo idioma para acessá-lo.

É possível que a IBM não ofereça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte seu representante IBM local para obter informações sobre os produtos e serviços disponíveis atualmente em sua área. Qualquer referência a produtos, programas ou serviços IBM não significa que apenas produtos, programas ou serviços IBM possam ser utilizados. Qualquer produto, programa ou serviço funcionalmente equivalente que não infrinja nenhum direito de propriedade intelectual da IBM pode ser usado em substituição. Entretanto, a avaliação e verificação da operação de qualquer produto, programa ou serviço não IBM são de responsabilidade do Cliente.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados nesta publicação. O fornecimento desta publicação não lhe garante direito algum sobre tais patentes. Pedidos de licença devem ser enviados, por escrito, para:

Gerência de Relações Comerciais e Industriais da IBM Brasil
IBM Corporation
Botafogo
Rio de Janeiro, RJ
Brasil

Para pedidos de licença relacionados a informações de Conjunto de Caracteres de Byte Duplo (DBCS), entre em contato com o Departamento de Propriedade Intelectual da IBM em seu país ou envie pedidos de licença, por escrito, para:

Intellectual Property Licensing
IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO "NO ESTADO EM QUE SE ENCONTRA", SEM GARANTIA DE NENHUM TIPO, SEJA EXPRESSA OU IMPLÍCITA, INCLUINDO, MAS NÃO SE LIMITANDO ÀS GARANTIAS IMPLÍCITAS DE MERCADO OU DE ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em certas transações; portanto, essa disposição pode não se aplicar ao Cliente.

Essas informações podem conter imprecisões técnicas ou erros tipográficos. São feitas alterações periódicas nas informações aqui contidas; tais alterações serão incorporadas em futuras edições desta publicação. A IBM pode, a qualquer momento, aperfeiçoar e/ou alterar os produtos e/ou programas descritos nesta publicação, sem aviso prévio.

Referências nestas informações a Web sites não IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a esses websites. Os materiais contidos nesses websites não fazem parte dos materiais desse produto IBM e a utilização desses websites é de inteira responsabilidade do Cliente.

A IBM pode utilizar ou distribuir as informações fornecidas da forma que julgar apropriada sem incorrer em qualquer obrigação para com o Cliente.

Licenciados deste programa que desejam obter informações sobre este assunto com objetivo de permitir: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este) e (ii) a utilização mútua das informações trocadas, devem entrar em contato com:

Gerência de Relações Comerciais e Industriais da IBM Brasil
IBM Corporation

Botafogo
Rio de Janeiro, RJ
Brasil

Tais informações podem estar disponíveis, sujeitas a termos e condições apropriadas, incluindo em alguns casos o pagamento de uma taxa.

O programa licenciado descrito nesta publicação e todo o material licenciado disponível são fornecidos pela IBM sob os termos do Contrato com o Cliente IBM, do Contrato Internacional de Licença do Programa IBM ou de qualquer outro contrato equivalente.

Os exemplos de clientes e dados de desempenho citados são apresentados com propósitos meramente ilustrativos. Os resultados reais de desempenho podem variar, dependendo das configurações e condições operacionais específicas.

As informações relativas a produtos não IBM foram obtidas junto aos fornecedores dos respectivos produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não IBM. Dúvidas sobre os recursos de produtos não IBM devem ser encaminhadas diretamente a seus fornecedores.

As declarações relacionadas aos objetivos e intenções futuras da IBM estão sujeitas a alterações ou cancelamento sem aviso prévio e representam apenas metas e objetivos.

Estas informações contêm exemplos de dados e relatórios utilizados nas operações diárias de negócios. Para ilustrá-los da forma mais completa possível, os exemplos podem incluir nomes de indivíduos, empresas, marcas e produtos. Todos estes nomes são fictícios e qualquer semelhança com nomes e endereços utilizados por uma empresa real é mera coincidência.

Marcas comerciais

IBM, o logotipo IBM e ibm.com são marcas comerciais ou marcas registradas da International Business Machines Corp., registradas em várias jurisdições no mundo todo. Outros nomes de empresas, produtos e serviços podem ser marcas comerciais da IBM ou de outras empresas. Uma lista atual de marcas registradas da IBM está disponível na web em "Copyright and trademark information" em www.ibm.com/legal/copytrade.shtml.

Adobe, o logotipo Adobe, PostScript e o logotipo PostScript são marcas ou marcas registradas do Adobe Systems Incorporated nos Estados Unidos e/ou em outros países.

Intel, o logotipo Intel, Intel Inside, o logotipo Intel Inside, Intel Centrino, o logotipo do Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium e Pentium são marcas comerciais ou marcas registradas da Intel Corporation ou suas subsidiárias nos Estados Unidos e em outros países.

Linux é uma marca registrada da Linus Torvalds nos Estados Unidos e/ou em outros países.

Microsoft, Windows, Windows NT e o logotipo Windows são marcas comerciais da Microsoft Corporation nos Estados Unidos e/ou em outros países.

UNIX é uma marca registrada do The Open Group nos Estados Unidos e/ou em outros países.

Java e todas as marcas comerciais e logotipos baseados em Java são marcas comerciais ou marcas registradas da Oracle e/ou de suas afiliadas.

Termos e condições da documentação do produto

As permissões para a utilização destas publicações são concedidas sujeitas aos termos e condições a seguir.

Aplicação

Estes termos e condições estão em adição a quaisquer termos de uso para o website IBM.

Uso pessoal

É possível reproduzir estas publicações para seu uso pessoal não comercial, desde que todos os avisos do proprietário sejam preservados. O Cliente não pode distribuir, exibir ou fazer trabalho derivado destas publicações, ou de qualquer parte delas, sem o consentimento expresso da IBM.

Uso Comercial

O Cliente pode reproduzir, distribuir e exibir estas publicações unicamente dentro de sua empresa, contanto que todos os avisos do proprietário sejam preservados. O Cliente não pode fazer trabalhos derivados destas publicações, ou reproduzir, distribuir ou exibir estas publicações ou qualquer parte delas fora da empresa, sem o consentimento expresso da IBM.

Direitas

Exceto quando expressamente concedido nesta permissão, nenhuma outra permissão, licença ou direito é concedido, seja de maneira expressa ou implícita, para as publicações ou quaisquer informações, dados, software ou outras propriedades intelectuais aqui contidas.

A IBM reserva-se o direito de retirar as permissões concedidas aqui sempre que, a seu critério, o uso das publicações seja prejudicial a seus interesses ou, conforme determinado pela IBM, as instruções acima não estejam sendo seguidas corretamente.

O Cliente não pode fazer download, exportar ou re-exportar estas informações, exceto se estiver em conformidade total com todas as leis e regulamentos aplicáveis, incluindo todas as leis e regulamentos de exportação dos Estados Unidos.

A IBM NÃO FAZ QUALQUER TIPO DE GARANTIA QUANTO AO CONTEÚDO DESTAS PUBLICAÇÕES. AS PUBLICAÇÕES SÃO FORNECIDAS "COMO ESTÃO" E SEM GARANTIA DE QUALQUER TIPO, EXPRESSAS OU IMPLÍCITAS, INCLUINDO MAS NÃO SE LIMITANDO A GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO, NÃO INFRAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO.

Índice remissivo

Caracteres Especiais

A

B

C

D

E

F

H

I

J

L

M

N

O

P

R

S

T

V