Guia de Script e Automação Python do IBM SPSS Modeler 17

Observação

Antes de usar estas informações e o produto por elas suportadas, leia as informações em "Avisos" na página 319.

Informações do produto

Esta edição se aplica à versão 17, release 0, modificação 0 do IBM(r) SPSS(r) Modeler e a todas as liberações e modificações subsequentes, até que seja 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 os scripts criados no IBM SPSS Statistics no 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 > **Execução**
2. Clique na guia **Execução** para trabalhar com scripts para o fluxo atual.

Os ícones da barra de ferramentas na parte superior da caixa de diálogo de script de fluxo permitem executar as operações a seguir:

- Importar o conteúdo de um script independente pré-existente na 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).
- Verificar a sintaxe do script e, se quaisquer erros forem localizados, exibi-los para revisão no painel inferior da caixa de diálogo.

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

Além disso, é possível especificar se esse script deve ou não ser executado quando o fluxo for executado. É possível selecionar **Executar este script** para executar o script toda vez em que o fluxo for executado, respeitando a ordem de execução do 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 utilizados pelo marcador de sintaxe podem ser customizados utilizando as preferências de exibição do IBM SPSS Modeler. É possível acessar as preferências de exibição escolhendo **Ferramentas** > **Opções** > **Opções do Usuário** e clicando na 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. Pressione Esc para sair do modo de sugestão automática sem modificar o texto existente.

A guia **Depuração** exibe mensagens de depuração e pode ser utilizada para avaliar o estado do script quando o script tiver sido executado. A guia **Depuração** consiste em uma área de texto somente leitura e em 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 texto de entrada sempre contém o prompt de comandos (`-->` para script legado).

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

- Um script é executado utilizando o botão "Executar este script" ou o botão "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 do painel de script não modificará o contexto de script do painel de script. Os valores de quaisquer variáveis criadas como parte dessa execução não estarão visíveis dentro do 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 de demo *druglearn.str* (disponível na pasta */Demos/streams/* em sua instalação do IBM SPSS Modeler) pode 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. Para aprender mais sobre a linguagem de script em geral, consulte Visão Geral de Linguagem de Script .

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 mais informações.

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/16/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 .

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`. Consulte o tópico “Propriedades de featureselectionnode” na página 187 para obter mais informações.

Scripts SuperNode

É possível criar e salvar os scripts em quaisquer SuperNodes de terminal utilizando a linguagem de script do 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 19, “Propriedades do SuperNode”, na página 303 para obter mais informações. Este método funciona para SuperNodes de origem e de processo, bem como para SuperNodes de terminal.

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

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'
```

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 execução de loop e colá-lo no editor de script clicando em **Colar...** no canto inferior direito da subguia Executar Loop; a exibição da guia Execução principal é alterada 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 condicional 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 o botão **Chave de Iteração ...** no canto inferior esquerdo da subguia Loop ou clique com o botão direito em qualquer nó no fluxo e selecione **Execução de Loop/Condicional** > **Definir Chave de Iteração (Campos)** ou **Execução de Loop/Condicional** > **Definir Chave de Iteração (Valores)**. 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 "Selecionando campos para iterações" na página 10 para obter mais informações.

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 o botão **Incluir Variável...** no canto inferior esquerdo da subguia Loop ou clique com o botão direito em qualquer nó no fluxo e selecione: **Execução de Loop/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 da 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 "Criando uma chave de iteração para loop em fluxos" na página 8 para obter mais informações.
  - **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 condicional e colá-lo no editor de script clicando em **Colar...** no canto inferior direito da subguia Condicional; a exibição da guia Execução principal é alterada 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. Consulte o tópico "Utilizando Argumentos de Linha de Comandos" na página 61 para obter mais informações.

Nota: Um script SuperNode é executado quando o SuperNode é executado, desde que **Executar este script** tenha sido selecionado 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 o texto do script ou da expressão, incluindo o editor de script, o construtor de expressões do CLEM ou ao definir um modelo no nó Relatório. Ao editar o texto em qualquer destas áreas, pressione `Ctrl+F` para acessar a caixa de diálogo, assegurando-se de que o cursor tenha o foco em uma área de texto. Se estiver trabalhando em um nó de Preenchimento, por exemplo, será 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 Tudo** para atualizar todas ou somente as instâncias selecionadas.
6. A caixa de diálogo é fechada após cada operação. Pressione F3 a partir de qualquer área de texto para repetir a última operação de procura ou pressione Ctrl+F para acessar a caixa de diálogo novamente.

Opção de Procura

Respeitar maiúsculas e minúsculas. Especifica se a operação de procura faz distinção entre maiúsculas e minúsculas, por exemplo, se *myvar* corresponde a *myVar*. O texto de substituição é sempre inserido exatamente conforme digitado, independentemente dessa configuração.

Somente palavras inteiras. Especifica se a operação de procura corresponde ao texto integrado nas palavras. Se selecionado, por exemplo, uma procura por *spider* não corresponderá a *spiderman* ou *spider-man*.

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

Somente o texto selecionado. Controla o escopo da procura quando utilizar a opção **Substituir Tudo**.

Sintaxe de Expressão Regular

As expressões regulares permitem procurar por caracteres especiais, como tabulações ou caracteres de nova linha, classes ou intervalos de caracteres, como *a* a *d*, qualquer dígito ou não dígito e limites, como o início ou o término de uma linha. Os tipos de expressões a seguir são suportados.

Tabela 1. Correspondências de caracteres.

Caracteres	**Correspondências**
x	O caractere x
\\	O caractere barra invertida
\0n	O caractere com o valor octal 0n (0 <= n <= 7)
\0nn	O caractere com o valor octal 0nn (0 <= n <= 7)
\0mnn	O caractere com o valor octal 0mn (0 <= m <= 3, 0 <= n <= 7)
\xhh	O caractere com o valor hexadecimal 0xhh
\uhhhh	O caractere com o 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 até z ou A até Z, inclusive (intervalo)
[a-d[m-p]]	a até d ou m até p (união). Como alternativa, isso pode ser especificado como [a-dm-p]
[a-z&&[def]]	a até z, e d, e ou f (interseção)
[a-z&&[^bc]]	a até z, exceto para b e c (subtração). Como alternativa, isso pode ser especificado como [ad-z]
[a-z&&[^m-p]]	a até z e não m até p (subtração). Como alternativa, isso pode ser especificado como [a-lq-z]

Tabela 3. Classes de caracteres predefinidas.

Classes de caracteres predefinidas	**Correspondências**
.	Qualquer caractere (pode ou não corresponder a 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 não espaço 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.

Comparadores de limites	**Correspondências**
^	O início de uma linha
$	O término de uma linha
\b	Um limite de palavra
\B	Um limite de não palavra
\A	O início da entrada
\Z	O término da entrada, mas para o terminador final, se houver algum
\z	O término 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. Este 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 utilizando um sinal de igual (=). Por exemplo, para designar o valor "3" para uma variável chamada "x", você utiliza 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`	O x é menor que y?
`x > y`	O x é maior que y?
`x <= y`	O x é menor ou igual a y?
`x >= y`	O x é maior ou igual a y?
`x == y`	O x é igual a y?
`x != y`	O x não é igual a y?
`x <> y`	O x não é igual a y?
`x + y`	Soma y ao x
`x - y`	Subtrai y de x
`x * y`	Multiplica x por y
`x / y`	Divide x por y
`x ** y`	Eleva o x à potência de 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 colchetes ([]) é conhecido como um *índice* e faz referência a um elemento específico 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 de caracteres

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 aspas pode conter o outro caractere aspas sem escape ou um caractere aspas com escape, que é precedido por um caractere 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()`	Altera a letra inicial `s` em maiúscula
`s.count(ss {,start {,end}})`	Conta as ocorrências de `ss` em `s[start:end]`

Tabela 6. Métodos de sequência (continuação)

Método	Uso
`s.startswith(str {, start {, end}})` `s.endswith(str {, start {, end}})`	Testa se o `s` inicia com `str` Testa se o `s` termina com `str`
`s.expandtabs({size})`	Substitui tabulações por espaços, o `size` padrão 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 para a esquerda.
`s.index(str {, start {, end}})` `s.rindex(str {, start {, end}})`	Localiza primeiro índice de `str` em `s`; se não localizado: aumenta o `ValueError`. `rindex` procura da direita para a 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)`	Junta as sequências em `seq` com `s` como o separador
`s.splitlines({keep})`	Divide `s` em linhas, se keep for `true`, mantém as novas linhas
`s.split({sep {, max}})`	Divide `s` em "palavras" usando `sep` (o `sep` padrão é um espaço em branco) para até `max` vezes

`s.ljust(width)`
`s.rjust(width)`
`s.center(width)`
`s.zfill(width)`
	Justifica a sequência à esquerda em um campo de largura `width`
	Justifica a sequência à direita em um campo de largura `width`
	Centraliza a sequência em um campo de largura `width`
	Preenche com 0.

`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})`	Converte `s` utilizando a tabela, depois de remover quaisquer caracteres em `delc`. `str` deve ser uma sequência de comprimento `== 256`.
`s.replace(old, new {, max})`	Substitui todas as `max` ocorrências da sequência `old` pela sequência `new`

Observações

As observações são comentários que são introduzidos pelo sinal de sustenido ou hash (#). Todo o texto após 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 para 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 e devem iniciar com um caractere alfabético maiúsculo ou minúsculo ou com 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 parâmetro:** `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. Os 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 após ele. 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:

```
Isso demonstra o uso de uma vírgula no término de uma instrução print.
```

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 são 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 agente iterativo `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

No 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)`	Retorna o limite de `x` como um valor flutuante, que é o menor número inteiro maior ou igual a `x`
`math.copysign(x, y)`	Retorna `x` com sinal de `y`. `copysign(1, -0.0)` retorna `-1`
`math.fabs(x)`	Retorna o valor absoluto de `x`
`math.factorial(x)`	Retorna `x` fatorial. Se `x` for negativo ou não for um número inteiro, um `ValueError` será emitido.
`math.floor(x)`	Retorna o piso de `x` como um valor flutuante, que é o maior número inteiro menor ou igual a `x`
`math.frexp(x)`	Retorna 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, de forma que `x == m * 2**e` exatamente. Se `x` for zero, retornará `(0,0, 0)`, caso contrário, `0.5 <= abs(m) < 1`.
`math.fsum(iterable)`	Retorna uma soma de valores de ponto flutuante de precisão `iterable`
`math.isinf(x)`	Verifique se o valor flutuante `x` é infinitivo positivo ou negativo
`math.isnan(x)`	Verifique se o valor flutuante `x` é `NaN` (não um número)
`math.ldexp(x, i)`	Retorna `x * (2**i)`. Isso é essencialmente o inverso da função `frexpf`.
`math.modf(x)`	Retornar as partes fracionárias e inteiras de `x`. Ambos os resultados transportam o sinal de `x` e são valores flutuantes.
`math.trunc(x)`	Retorna o valor `Real` de `x`, que foi truncado para um `Integral`.
`math.exp(x)`	Retorna `e**x`
`math.log(x[, base])`	Retorna o logaritmo de `x` para o valor especificado de `base`. Se `base` não for especificado, o logaritmo natural de `x` será retornado.
`math.log1p(x)`	Retorna o logaritmo natural de `1+x (base e)`
`math.log10(x)`	Retorna o logaritmo de base 10 de `x`

Tabela 7. Métodos matemáticos (continuação)

Método	Uso
`math.pow(x, y)`	Retorna `x` elevado à potência `y`. `pow(1.0, x)` e `pow(x, 0.0)` sempre retorna 1, mesmo quando `x` for zero ou NaN.
`math.sqrt(x)`	Retorna 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)`	Retorna o arco cosseno de `x` em radianos
`math.asin(x)`	Retorna o arco seno de `x` em radianos
`math.atan(x)`	Retorna o arco tangente de `x` em radianos
`math.atan2(y, x)`	Retorna `atan(y / x)` em radianos.
`math.cos(x)`	Retorna o cosseno de `x` em radianos.
`math.hypot(x, y)`	Retorna a norma euclidiana `sqrt(x*x + y*y)`. Este é o comprimento do vetor da origem até o ponto `(x, y)`.
`math.sin(x)`	Retorna o seno de `x` em radianos
`math.tan(x)`	Retorna a tangente de `x` em radianos
`math.degrees(x)`	Converte o ângulo `x` de radianos em graus
`math.radians(x)`	Converte o ângulo `x` de graus em radianos
`math.acosh(x)`	Retorna o cosseno hiperbólico inverso de `x`
`math.asinh(x)`	Retorna o seno hiperbólico inverso de `x`
`math.atanh(x)`	Retorna a tangente hiperbólica inversa de `x`
`math.cosh(x)`	Retorna o cosseno hiperbólico de `x`
`math.sinh(x)`	Retorna o cosseno hiperbólico de `x`
`math.tanh(x)`	Retorna a tangente hiperbólica de `x`

Há também duas constantes matemáticas. O valor de `math.pi` é o pi constante matemático. O valor de `math.e` é a constante matemática 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 literais de sequência Unicode sejam especificados incluindo um prefixo de caractere `u` antes do literal de sequência:

```
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 este propósito.
- **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.

No Python, as classes são definidas utilizando a instrução `class`. A instrução `class` tem o seguinte formato:

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

ou

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

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 utilizada porque uma instrução é necessária para concluir 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):
        passar

    def method2(self, p1, p2):
        passar

    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 37 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 utilizada com o script de SuperNode. Esta função retorna o diagrama no SuperNode. Para outros tipos de script, esta função retorna a mesma função `stream()`.
- A função `supernode()` pode ser utilizada com scripts de SuperNode. 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 das funções de `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`.	O mesmo para `stream()`	Não aplicável
Fluxos	Retorna uma sessão	Retorna um fluxo	O mesmo 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 forma de terminar 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 que é definido para um fluxo é `runAll(Lista)`. 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 for concluída, quaisquer objetos que forem gerados pela execução podem ser acessados a partir da 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 utilizado.
`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 esse tipo não 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 nó pode ser `None`, caso em que o outro parâmetro é utilizado. Se diversos nós resultarem em uma correspondência, uma correspondência arbitrária será escolhida e retornada. Se nenhum nó resultar em uma correspondência, então o valor de retorno será `None`.
`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.

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 essa propriedade não existir.
`p.setPropertyValue(propertyName, value)`	Não aplicável	Configura o valor da propriedade nomeada.
`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 essa propriedade ou chave não existir.
`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.
`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.

Tabela 13. Métodos para vincular e desvincular nós (continuação)

Método	Tipo de retorno	Descrição
`s.disconnect(node)`	Não aplicável	Remove quaisquer links entre o nó fornecido e quaisquer outros nós no fluxo especificado.
`s.isValidLink(source, target)`	*boolean*	Retorna `true` se for válido criar um link entre os nós de origem e de 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. O sinalizador `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 dos limites.
`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 dos limites.

Tabela 15. Métodos para identificar os nós de envio e de recebimento de dados (continuação)

Método	Tipo de retorno	Descrição
`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()`	*string*	Retorna o rótulo de exibição do nó especificado. O rótulo será o valor da propriedade `custom_name` somente se essa propriedade for uma sequência não vazia e a propriedade `use_custom_name` não estiver configurada, caso contrário, o rótulo será o valor de `getName()`.

Tabela 16. Métodos para obter o ID, o nome e o rótulo de um nó (continuação)

Método	Tipo de retorno	Descrição
`n.setLabel(label)`	Não aplicável	Configura o rótulo de exibição do nó especificado. Se o novo rótulo for uma sequência não vazia, ele será designado à propriedade `custom_name` e `False` será designado à propriedade `use_custom_name`, de modo que o rótulo especificado terá prioridade, caso contrário, uma sequência vazia será designada à propriedade `custom_name` e `True` será designado à propriedade `use_custom_name`.
`n.getName()`	*string*	Retorna o nome do nó especificado.
`n.getID()`	*string*	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()`	*string*	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.
`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.

Tabela 17. Métodos para obter informações sobre um nó (continuação)

Método	Tipo de retorno	Descrição
`n.isCacheEnabled()`	*Booleano*	Retornará `True` se o cache estiver ativado, caso contrário, retornará `False`.
`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*	Retornará `True` se o cache estiver cheio, caso contrário, retornará `False`.
`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 *Guia de Referência da API de Script Python do IBM SPSS Modeler 17*.

Exemplo: procurando por nós utilizando 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. Criação de uma nova classe que estende `NodeFilter` e que implemente 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 atenderem 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)
```

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 utiliza o fluxo `druglearn.str` padrão do IBM SPSS Modeler e, para cada campo, constrói um modelo com um dos campos de entrada eliminados. 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 de modo que a linguagem de script de fluxo é configurada para Legacy).

```
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*	Retornará `True` se uma coluna com o nome fornecido existir neste DataModel, caso contrário, `False`.
`d.getColumn(name)`	Coluna	Retorna a coluna com o nome especificado.
`d.getColumnGroup(name)`	ColumnGroup	Retorna o grupo de coluna nomeado ou `None` se esse grupo de coluna não existir.

Tabela 18. Métodos do objeto DataModel para acessar informações sobre campos ou colunas (continuação)

Método	Tipo de retorno	Descrição
`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()`	Column[]	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()`	*string*	Retorna o nome da coluna.
`c.getColumnLabel()`	*string*	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.
`c.isMeasureDiscrete()`	*Boolean*	Retornará `True` se a coluna for discreta. As colunas que forem um conjunto ou um sinalizador são consideradas discretas.
`c.isModelOutputColumn()`	*Boolean*	Retornará `True` se a coluna uma coluna de saída de modelo.
`c.isStorageDatetime()`	*Booleano*	Retornará `True` se o armazenamento da coluna for um valor de hora, data ou registro de data e hora.
`c.isStorageNumeric()`	*Booleano*	Retornará `True` se o armazenamento da coluna for um número inteiro ou um número real.
`c.isValidValue(value)`	*Booleano*	Retornará `True` se o valor especificado for válido para esse armazenamento e `valid` quando os valores de coluna válidos forem conhecidos.
`c.getModelingRole()`	ModelingRole	Retorna o tipo de modelagem para a coluna.
`c.getSetValues()`	Object[]	Retorna uma matriz de valores válidos para a coluna ou `None` se um dos valores não for conhecido ou se a coluna não for um conjunto.

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

Método	Tipo de retorno	Descrição
`c.getValueLabel(value)`	*string*	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 de indicador "false" para a coluna ou `None` se o valor não for conhecido ou se a coluna não for um sinalizador.
`c.getTrueFlag()`	Objeto	Retorna o valor de indicador "true" para a coluna ou `None` se o valor não for conhecido ou se a coluna não for um sinalizador.
`c.getLowerBound()`	Objeto	Retorna o valor de limite inferior para os valores na coluna ou `None` se o valor não for conhecido ou se a coluna não for contínua.
`c.getUpperBound()`	Objeto	Retorna o valor de limite superior para os valores na coluna ou `None` se o valor não for conhecido ou se a coluna não for 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 executa loop nos resultados e quaisquer saídas de modelo resultantes da execução são salvas como um arquivo (`.gm`) de modelo do IBM SPSS Modeler e o modelo é exportado pelo 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)`	Fluxos	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.
`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)`	Fluxos	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 poderá fazer com que uma `ModelerException` seja emitida (todas as exceções que são geradas pelo 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

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 esse parâmetro não 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)`	*string*	Retorna o rótulo do parâmetro nomeado ou `None` se esse parâmetro não existir.

Tabela 21. Funções definidas pelo objeto ParameterProvider (continuação)

Método	Tipo de retorno	Descrição
`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 esse parâmetro não existir.
`p.setParameterStorage( parameterName, storage)`	Não aplicável	Configura o armazenamento do parâmetro nomeado.
`p.getParameterType(parameterName)`	ParameterType	Retorna o tipo do parâmetro nomeado ou `None` se esse parâmetro não existir.
`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 esse parâmetro não existir.
`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.

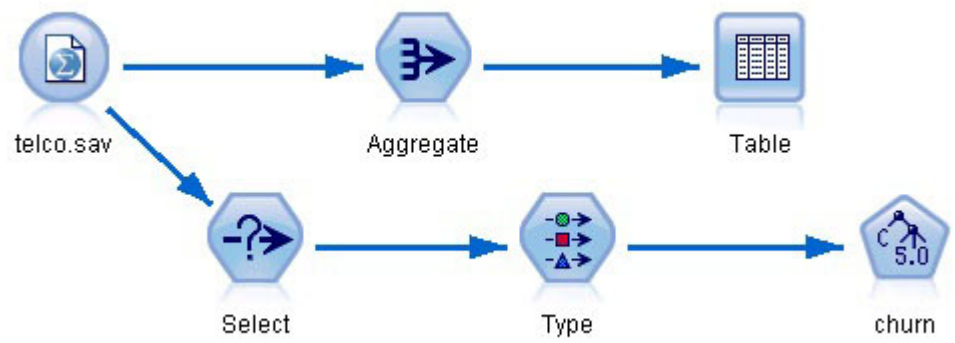


Figura 5. Fluxo resultante do script de exemplo

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 nome do campo especificados, 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 contendo as entradas conhecidas para o nome do campo especificado, ou `None` se não houver entradas existentes para o campo.

`GlobalValues.Type` define o tipo de estatísticas de resumo 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/17/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()
```

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 Collaboration and Deployment Services Repository.

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 isso ocorrer, o script executará loop em cada campo no nó e utilizará a função `field.upper()` ou `field.getColumnName().upper()` para alterar o nome para letras maiúsculas.

Acessando Objetos no IBM SPSS Collaboration and Deployment Services Repository

Se você tiver licenciado o IBM SPSS Collaboration and Deployment Services Repository, será possível armazenar, recuperar, bloquear e desbloquear objetos a partir do repositório usando comandos de script. O repositório permite gerenciar o ciclo de vida de modelos de mineração de dados e objetos preditivos relacionados no contexto de aplicativos, ferramentas e soluções corporativos.

Conectando-se ao IBM SPSS Collaboration and Deployment Services Repository

Para acessar o repositório, deve-se primeiro configurar uma conexão válida com ele, por meio do menu Ferramentas da interface com o usuário do IBM SPSS Modeler ou por meio da linha de comandos. (Consulte o tópico "Argumentos de Conexão do IBM SPSS Collaboration and Deployment Services Repository" na página 65 para obter mais informações).)

Armazenando e Recuperando Objetos

Em um script, os comandos `retrieve` e `store` permitem acessar vários objetos, incluindo fluxos, modelos, saída, nós e projetos. A sintaxe é a seguinte:

```
store object as REPOSITORY_PATH {label LABEL}
store object as URI [#l.label]

retrieve object REPOSITORY_PATH {label LABEL | version VERSION}
retrieve object URI [(#m.marker | #l.label)]
```

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.

```
store stream as "/folder_1/folder_2/mystream.str"
store model Drug as "/myfolder/drugmodel"
store model Drug as "/myfolder/drugmodel.gm" label "final"
store node DRUG1n as "/samples/drug1ntypenode"
store project as "/CRISPDM/DrugExample.cpj"
store output "Data Audit of [6 fields]" as "/my folder/My Audit"
```

Opcionalmente, uma extensão como *.str* ou *.gm* pode ser incluída no nome do objeto, mas isso não é necessário desde que o nome esteja consistente. Por exemplo, se um modelo estiver armazenado sem uma extensão, ele deverá ser recuperado pelo mesmo nome:

```
store model "/myfolder/drugmodel"
retrieve model "/myfolder/drugmodel"
```

versus:

```
store model "/myfolder/drugmodel.gm"
retrieve model "/myfolder/drugmodel.gm" version "0:2005-10-12 14:15:41.281"
```

Observe que ao recuperar objetos, a versão mais recente do objeto é sempre retornada, a menos que você especifique uma versão ou rótulo. Ao recuperar um objeto de nó, o nó será inserido automaticamente no fluxo atual. Ao recuperar um objeto de fluxo, deve-se utilizar um script independente. Não é possível recuperar um objeto de fluxo de dentro de um script de fluxo.

Bloqueio e Desbloqueio de 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:

```
lock REPOSITORY_PATH
lock URI

unlock REPOSITORY_PATH
unlock 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.

```
lock "/myfolder/Stream1.str"

unlock "/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. Seguem alguns exemplos:

```
lock "spsscr:///myfolder/Stream1.str"

unlock "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)

Por meio 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. Apenas especifique e execute um fluxo independente na linha de comandos quando 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 irá acontecer ou não dependerá das configurações das opções **Incluir modelo no 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 do 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 utilizados por scripts para orientar a execução subsequente. 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]

# Now that we have the C5.0 model output object, access the
# relevant content model
cm = modeloutput.getContentModel("PMML")

# 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")
```

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 em formato XML
- O **Modelo de conteúdo JSON** fornece acesso ao conteúdo armazenado em formato JSON
- O **Modelo de conteúdo de estatísticas de coluna** fornece acesso às estatísticas de resumo sobre um campo específico
- O **Modelo de conteúdo de estatísticas de coluna de pares** fornece acesso às estatísticas de resumo entre dois campos ou valores entre dois campos separados

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

Tabela 23. API

Retornar	Método	Descrição
`int`	`getRowCount()`	Retorna o número de linhas nesta tabela.
`int`	`getColumnCount()`	Retorna o número de colunas nesta tabela.
`Sequência de caracteres`	`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.
`Objeto`	`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 24. 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

Tabela 25. API

Retornar	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 assegurar que as mudanças sejam selecionadas por chamadas subsequentes.
`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 26. 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

Tabela 27. API

Retornar	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).
`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).

Tabela 27. API (continuação)

Retornar	Método	Descrição
`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).

Script de exemplo

Se houver um nó construtor de saída que cria a saída com base no formato JSON, o seguinte poderá ser utilizado para acessar informações sobre um conjunto de manuais:

```
results = []
outputbuilder.run(results)
output = results[0]
cm = output.getContentModel("jsonContent")

bookTitle = cm.getObjectAt(["books", "ISIN123456", "title"], None)

# Alternatively, get the book object and use it as the root
# for subsequent entries
book = cm.getObjectAt(["books", "ISIN123456"], None)
bookTitle = cm.getObjectAt(["title"], book)

# Get all child values for aspecific book
bookInfo = cm.getChildValuesAt(["books", "ISIN123456"], None)

# Get the third book entry. Assumes the top-level "books" value
# contains a JSON array which can be indexed
bookInfo = cm.getObjectAt(["books", 2], None)

# Get a list of all child entries
allBooks = cm.getChildrenAt(["books"], None)
```

Modelo de Conteúdo de Estatísticas de Coluna e Modelo de Conteúdo de Estatísticas de Pares

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 28. API de ColumnStatsContentModel.

Retornar	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 29. API PairwiseStatsContentModel.

Retornar	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 30. Nós e saídas.

Nome do nó	Nome de saída	ID do Contêiner	Notas
`"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 utilizar a linha de comandos de seu sistema operacional para ativar o IBM SPSS Modeler, conforme a seguir:

1. Em um computador no qual o IBM SPSS Modeler está instalado, abra uma janela de DOS ou de prompt de comandos.
2. Para ativar a interface do IBM SPSS Modeler no modo interativo, digite o comando `modelerclient` seguido pelos argumentos necessários, por exemplo:

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

Os argumentos disponíveis (sinalizadores) permitem conectar-se a um servidor, carregar fluxos, executar scripts ou especificar outros parâmetros conforme necessário.

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 31. Tipos de argumentos de linha de comandos.

Tipo de argumento	Onde descrito
Argumentos do sistema	Consulte o tópico "Argumentos do sistema" na página 62 para obter mais informações.
Argumentos de parâmetro	Consulte o tópico "Argumentos de Parâmetros" na página 63 para obter mais informações.
Argumentos de conexão do servidor	Consulte o tópico "Argumentos de Conexão do Servidor" na página 64 para obter mais informações.
Argumentos de conexão do IBM SPSS Collaboration and Deployment Services Repository	Consulte o tópico "Argumentos de Conexão do IBM SPSS Collaboration and Deployment Services Repository" na página 65 para obter mais informações.
Argumentos de conexão do IBM SPSS Analytic Server	Consulte o tópico "Argumentos de Conexão do IBM SPSS Analytic Server" na página 65 para obter mais informações.

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
```

Também é possível executar os estados e scripts do IBM SPSS Modeler dessa maneira utilizando os 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 32. 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. Consulte o tópico "Combinando Diversos Argumentos" na página 66 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:\\`
`-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.
`-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` <project>	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 com o 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 a partir do IBM SPSS Collaboration and Deployment Services Repository

Como é possível carregar determinados objetos a partir de um arquivo ou a partir do IBM SPSS Collaboration and Deployment Services Repository (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 Collaboration and Deployment Services Repository 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 Collaboration and Deployment Services Repository, consulte o tópico "Acessando Objetos no IBM SPSS Collaboration and Deployment Services Repository" na página 50.

Argumentos de Parâmetros

Os parâmetros podem ser utilizados como sinalizadores durante a execução da linha de comandos do IBM SPSS Modeler. Nos argumentos da linha de comandos, o sinalizador `-P` é utilizado para denotar um parâmetro no formato `-P` <*name*>=<*value*>.

Os parâmetros podem ser qualquer um dos seguintes:

- **Parâmetros simples** (ou parâmetros utilizados diretamente em expressões do CLEM).
- **Parâmetros de slot**, também referidos como **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 69 para obter informações adicionais.
- **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,
 true}"
```

O formato é o mesmo que o parâmetro `datasource` da propriedade do nó `databasenode`. Para obter mais informações, consulte: "Propriedades de databasenode" na página 81.

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%\"}
```

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
```

Observe que conectar-se a um cluster de servidores requer o Coordenador de Processos por meio do IBM SPSS Collaboration and Deployment Services, portanto, o argumento `-cluster` deverá ser usado em combinação com as opções de conexão do repositório (`spsscr_*`). Consulte o tópico "Argumentos de Conexão do IBM SPSS Collaboration and Deployment Services Repository" na página 65 para obter mais informações.

Tabela 33. 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 <number>`	O número da porta do servidor especificado. Disponível apenas no modo de servidor.
`-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 Collaboration and Deployment Services Repository. 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 Collaboration and Deployment Services Repository" na página 65 para obter mais informações.
`-username <name>`	O nome do usuário com o qual efetuar logon no servidor. Disponível apenas no modo de servidor.

Tabela 33. Argumentos de conexão do servidor (continuação).

Argumento	Comportamento/Descrição
`-password` <password>	A senha com a qual efetuar logon no servidor. Disponível apenas no modo de servidor. *Nota*: Se o argumento `-password` não for utilizado, será solicitado a fornecer 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 Collaboration and Deployment Services Repository

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 Collaboration and Deployment Services Repository. 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 34. Argumentos de conexão do IBM SPSS Collaboration and Deployment Services Repository

Argumento	Comportamento/Descrição
`-spsscr_hostname` <hostname or IP address>	O nome do host ou o endereço IP do servidor no qual o IBM SPSS Collaboration and Deployment Services Repository está instalado.
`-spsscr_port` <number>	O número da porta na qual o IBM SPSS Collaboration and Deployment Services Repository 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.
`-spsscr_username` <name>	O nome de usuário com o qual efetuar logon no IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_password` <password>	A senha com a qual efetuar logon no IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_epassword` <encoded password>	A senha codificada com a qual efetuar logon no IBM SPSS Collaboration and Deployment Services Repository.
`-spsscr_domain` <name>	O domínio utilizado para efetuar logon no IBM SPSS Collaboration and Deployment Services Repository. Esse sinalizador é opcional - não o utilize, a não ser que você efetue logon utilizando o LDAP ou o Active Directory.

Argumentos de Conexão do IBM SPSS Analytic Server

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 do Analytic Server é obtido a partir do SPSS Modeler Server e não pode ser alterado no cliente.

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

Tabela 35. Argumentos de conexão do IBM SPSS Analytic Server

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 diversas propriedades diferentes para nós, fluxos, SuperNodes e projetos. 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".
- As propriedades do nó podem ser utilizadas em parâmetros de SuperNode.
- As propriedades do nó também podem ser utilizadas como parte de uma opção de linha de comandos (utilizando o sinalizador `-P`) ao iniciar o 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.

Para obter mais informações sobre a linguagem de script, consulte Linguagem de Script.

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). Esta propriedade pode ter (ou ter recebido) o valor `True` ou `False`. Suponha que um nó Filtro denominado `mynode` possua (em seu lado de envio de dados) um campo chamado `Age`. Para desativar isso, configure a propriedade `include`, com a chave `Age`, para o valor `False`, conforme 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 36. Abreviações padrão utilizadas em toda a sintaxe

Abreviação	Significado
abs	Valor absoluto
len	Comprimento
mín.	Mínimo
máx.	Máximo
correl	Correlação
covar	Covariância
num	Número ou numérico
pct	Percentual ou porcentagem
transp	Transparência
xval	Validação cruzada
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 37. 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 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 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 slot podem ser estruturados por si só, nos casos em que os atributos de um nó forem mais complexos do que apenas 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 de propriedade pode especificar o intervalo de valores legais, como `Discard`, `PairAndDiscard` e `IncludeAsText`, caso em que um erro será gerado se qualquer outro valor for utilizado. As propriedades do sinalizador devem ser lidas ou configuradas utilizando valores `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. Utilizar `true` e `false` de modo consistente 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 38. Propriedades comuns do nó.

Nome da propriedade	Tipo de dados	Descrição da propriedade
`use_custom_name`	*sinalizador*	
`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*	

Tabela 38. Propriedades comuns do nó (continuação).

Nome da propriedade	Tipo de dados	Descrição da propriedade
`annotation`	*sequência*	
`keywords`	*sequência*	Slot estruturado que especifica uma lista de palavras-chave associadas ao objeto (por exemplo, `["Keyword1" "Keyword2"]`).
`cache_enabled`	*sinalizador*	

`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, ao invés de referenciar 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 19, “Propriedades do SuperNode”, na página 303 para obter mais informações.

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 39. Propriedades do Fluxo.

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

Tabela 39. 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" MES AAAA t T AAAA ss SM AAAA	
date_baseline	*número*	
date_2digit_baseline	*número*	
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	*flag*	
import_datetime_as_string	*flag*	
decimal_places	*número*	
decimal_symbol	Default Período Comma	
angles_in_radians	*flag*	
use_max_set_size	*flag*	
max_set_size	*número*	
ruleset_evaluation	Voting FirstHit	

Tabela 39. Propriedades do Fluxo (continuação).

Nome da propriedade	Tipo de dados	Descrição da propriedade
refresh_source_nodes	*flag*	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_execução	*booleano*	
stream_rewriting_optimise_syntax_execução	*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*	
record_count_feedback_interval	*número inteiro*	
use_stream_auto_create_node_configurações	*boolean*	Se true, as configurações específicas do fluxo serão utilizadas, caso contrário, as preferências do usuário são utilizadas.
create_model_applier_for_new_modelos	*boolean*	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.

Tabela 39. Propriedades do Fluxo (continuação).

Nome da propriedade	Tipo de dados	Descrição da propriedade

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

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 40. Propriedades comuns do nó de origem.

Nome da propriedade	Tipo de dados	Descrição da propriedade

`direction`	`Input`
	`Target`
	`Both`
	`None`
	`Partição`
	`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.

Tabela 40. Propriedades comuns do nó de origem (continuação).

Nome da propriedade	Tipo de dados	Descrição da propriedade
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 Data Timestamp	Propriedade definida como chave somente leitura para tipo de armazenamento de campo. Formato de uso: NODE.storage.FIELDNAME
check	None Anular Impor Discard Avisar 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	Leitura Passagem 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 40. Propriedades comuns do nó de origem (continuação).

Nome da propriedade	Tipo de dados	Descrição da propriedade
`extend_values`	*flag*	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`	*string*	Utilizado para especificar um rótulo de valor. Observe que os valores devem ser especificados primeiro.
`enable_missing`	*flag*	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`	*flag*	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`	*string*	Quando `range_missing` for true, especifica o limite inferior do intervalo de valores omissos. Formato de uso: `NODE.missing_lower.FIELDNAME`
`missing_upper`	*string*	Quando `range_missing` for true, especifica o limite superior do intervalo de valores omissos. Formato de uso: `NODE.missing_upper.FIELDNAME`
`null_missing`	*flag*	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`	*flag*	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`	*string*	Utilizado para especificar um rótulo ou descrição de campo.

Tabela 40. Propriedades comuns do nó de origem (continuação).

Nome da propriedade	Tipo de dados	Descrição da propriedade
default_include	*flag*	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	*flag*	Propriedade definida como chave utilizada para determinar se campos individuais são incluídos ou filtrados: NODE.include.FIELDNAME.
new_name	*string*	
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 definida como chave é semelhante a type por ela poder ser utilizada para definir a medida associada ao campo. A diferença é que, no script Python, a função setter também pode transmitir um dos valores de MeasureType, ao passo que a função 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 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 Polígono 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	*boolean*	Para campos geoespaciais, essa propriedade define se esse campo tem um sistema de coordenadas
coordinate_system	*string*	Para campos geoespaciais, esta propriedade definida como chave define o sistema de coordenadas para este campo.

Tabela 40. Propriedades comuns do nó de origem (continuação).

Nome da propriedade	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`	Essa propriedade definida como chave é semelhante a `custom_storage` por ela poder ser utilizada para definir o armazenamento de substituição para o campo. A diferença é que, no script Python, a função setter também pode transmitir um dos valores de `StorageType`, ao passo que a função getter 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`	*integer*	Para campos de lista, esta propriedade definida como chave especifica a profundidade do campo

Propriedades de asimport

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

exemplo

```
node = stream.create("asimport", "My node")
node.setPropertyValue("data_source", "Drug1n")
```

Tabela 41. Propriedades de asimport.

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

Propriedades do Nó cognosimport

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

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 42. Propriedades do nó cognosimport.

Propriedades do nó cognosimport	**Tipo de dados**	**Descrição da propriedade**
`mode`	`Data` `Relatórios`	Especifica se devem ser importados dados (padrão) ou relatórios do Cognos BI.
`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.
`cognos_package_name`	*string*	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`	*["field","field", ... ,"field"]*	O nome de um ou mais objetos de dados a serem importados. O formato de *field* é [*namespace*].[*query_subject*].[*query_item*]
`cognos_filters`	*campo*	O nome de um ou mais filtros para aplicar antes de importar os dados.

Tabela 42. Propriedades do nó cognosimport (continuação).

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

O nó Banco de Dados pode ser utilizado para importar dados de uma variedade de outros pacotes utilizando ODBC (Open Database Connectivity), incluindo Microsoft SQL Server, o DB2, o Oracle e outros.

exemplo

```
import modeler.api
stream = modeler.script.stream()
nnode = 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 43. Propriedades de databasenode.

Propriedades de **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).

Tabela 43. Propriedades de databasenode (continuação).

Propriedades de databasenode	Tipo de dados	Descrição da propriedade
username	*sequência*	Detalhes de conexão com o banco de dados (consulte também a nota a seguir).
password	*sequência*	
credential	*string*	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 51 para obter mais informações. Esta propriedade é somente leitura durante a execução.
tablename	*sequência*	Nome da tabela que deseja acessar.
strip_spaces	Nenhum 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 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 os valores da sequência datasource entre aspas duplas e chaves, como no exemplo a seguir: "{\"SQL Server\",spssuser,abcd1234,false}".

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 44. Propriedades de databasenode - específica da origem de dados.

Propriedades de 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 Importação de Dados do IBM SPSS Data Collection importa dados de pesquisa de opinião com base no Modelo de Dados do IBM SPSS Data Collection usado pelos produtos de pesquisa de mercado do IBM Corp.. O IBM SPSS Data Collection Data Library deve ser instalado para usar este nó.

Figura 7. nó Importação de Dados de Dimensões

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 45. Propriedades de datacollectionimportnode.

Propriedades de `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 IBM SPSS Data Collection 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 45. Propriedades de datacollectionimportnode (continuação).

Propriedades de `datacollectionimportnode`	**Tipo de dados**	**Descrição da propriedade**

`casedata_name`	*sequência*	O nome do CDSC. Os 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.
`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 IBM SPSS Data Collection, 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`
	`Mais recente`
	`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.

Tabela 45. Propriedades de datacollectionimportnode (continuação).

Propriedades de datacollectionimportnode	Tipo de dados	Descrição da propriedade
use_context	*sequência*	Define se um contexto específico deve ser importado. Os contextos são utilizados para variar a descrição associada às respostas.
contexto	*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*	
import_system_variables	Common Nenhum All	Especifica quais variáveis do sistema são importadas.
import_codes_variables	*sinalizador*	
import_sourcefile_variables	*sinalizador*	
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 46. Propriedades de excelimportnode.

Propriedades de 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.
named_range	*sequência*	
worksheet_mode	Index Nome	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).

Propriedades de evimportnode

O nó Visualização Corporativa cria uma conexão com um IBM SPSS Collaboration and Deployment Services Repository, permitindo ler dados da Visualização Corporativa em um fluxo e empacotar um modelo em um cenário que possa ser acessado a partir do repositório por outros usuários.
Nota: O nó Visualização Corporativa foi substituído no SPSS Modeler 16.0 pelo nó Visualização de Dados. Para fluxos salvos em liberações anteriores, o nó Visualização Corporativa ainda é suportado. No entanto, ao atualizar ou criar novos fluxos, recomenda-se utilizar o nó Visualização de Dados.

exemplo

```
node = stream.create("evimport", "My node")
node.setPropertyValue("connection", ["Training data","/Application views/Marketing","LATEST",
"Analytic", "/Data Providers/Marketing"])
node.setPropertyValue("tablename", "cust1")
```

Tabela 47. Propriedades de evimportnode.

Propriedades de evimportnode	Tipo de dados	Descrição da propriedade
connection	*lista*	Propriedade estruturada -- lista de parâmetros que compõem uma conexão de Visualização Corporativa. Formato de uso: `evimportnode.connection = [description,app_view_path, app_view_version_label, environment,DPD_path]`
tablename	*sequência*	O nome de uma tabela na Visualização do Aplicativo.

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 48. Propriedades de fixedfilenode.

Propriedades de fixedfilenode	Tipo de dados	Descrição da propriedade
record_len	*número*	Especifica o número de caracteres em cada registro.
line_oriented	*flag*	Ignora o caractere de nova linha no término de cada registro.
decimal_symbol	Default Vírgula Period	O tipo de separador decimal utilizado em sua origem de dados.
skip_header	*número*	Especifica o número de linhas a serem ignoradas no início do primeiro registro. Útil para ignorar cabeçalhos da coluna.
auto_recognize_datetime	*flag*	Especifica se datas ou horas são identificadas automaticamente nos dados de origem.
lines_to_scan	*número*	
fields	*lista*	Propriedade estruturada.
full_filename	*string*	Nome completo do arquivo a ser lido, incluindo o diretório.

Tabela 48. Propriedades de fixedfilenode (continuação).

Propriedades de **fixedfilenode**	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*	
use_custom_values	*flag*	
custom_storage	Unknown String Integer Real Time Date Timestamp	
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" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MES AAAA t T AAAA ww WK YYYY	Esta propriedade se aplicará apenas se um armazenamento customizado tiver sido especificado.

Tabela 48. Propriedades de fixedfilenode (continuação).

Propriedades de fixedfilenode	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"	Esta propriedade se aplicará apenas se um armazenamento customizado tiver sido especificado.
custom_decimal_symbol	*field*	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 49. Propriedades do nó gsdata_import

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

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 50. Propriedades de sasimportnode.

Propriedades de 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	*sinalizador*	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 quiser avaliar o resultado de um modelo preditivo quando houver incerteza nas entradas do modelo.

Tabela 51. Propriedades de simgennode.

Propriedades de simgennode	Tipo de dados	Descrição da propriedade
campos	Propriedade estruturada	Veja o exemplo
correlations	Propriedade estruturada	Veja o exemplo
keep_min_max_setting	*boolean*	
refit_correlations	*boolean*	
max_cases	*integer*	O valor mínimo é 1000 e o valor máximo é 2.147.483.647
create_iteration_field	*boolean*	
iteration_field_name	*string*	
replicate_results	*boolean*	
random_seed	*integer*	
parameter_xml	*string*	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]
])
```

O `distribution` é uma declaração do nome de distribuição seguida por uma lista contendo pares de nomes e valores de atributos. 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 utiliza 2 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 `NegativeBinomialFailures` e também em `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 299.

Propriedades do Nó tm1import

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

Tabela 52. Propriedades do nó tm1import.

Propriedades do nó tm1import	Tipo de dados	Descrição da propriedade
pm_host	*string*	O nome do host. Por exemplo: `TM1_import.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
tm1_connection	*["field","field", ... ,"field"]*	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'])`

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 53. Propriedades de userinputnode.

Propriedades de 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, os registros serão gerados 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 obtido de cada coluna para cada registro para 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 54. Propriedades de variablefilenode.

Propriedades de variablefilenode	Tipo de dados	Descrição da propriedade
skip_header	*número*	Especifica o número de caracteres a serem ignorados no início do primeiro registro.
num_fields_auto	*flag*	Determina o número de campos em cada registro automaticamente. Os registros devem ser finalizados com um caractere de nova linha.
num_fields	*número*	Especifica manualmente o número de campos em cada registro.
delimit_space	*flag*	Especifica o caractere utilizado para delimitar limites de campo no arquivo.
delimit_tab	*flag*	
delimit_new_line	*flag*	
delimit_non_printing	*flag*	
delimit_comma	*flag*	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	*flag*	Permite especificar um delimitador customizado utilizando a propriedade other.
other	*string*	Especifica o delimitador utilizado quando delimit_other é *true*.
decimal_symbol	Default Vírgula Period	Especifica o separador decimal utilizado na origem de dados.
multi_blank	*flag*	Trata diversos caracteres delimitadores em branco adjacentes como um delimitador único.
read_field_names	*flag*	Trata a primeira linha no arquivo de dados como rótulos para a coluna.
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*	

Tabela 54. Propriedades de variablefilenode (continuação).

Propriedades de variablefilenode	Tipo de dados	Descrição da propriedade
break_case_by_newline	*flag*	Especifica que o delimitador de linha é o caractere de nova linha.
lines_to_scan	*número*	Especifica quantas linhas devem ser varridas para tipos de dados especificados.
auto_recognize_datetime	*flag*	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	*flag*	
custom_storage	Unknown Sequência de caracteres Número inteiro Real Time Data Timestamp	
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" "DD/MM/YY" "DD/MM/YYYY" "MM/DD/YY" "MM/DD/YYYY" "DD/MON/YY" "DD/MON/YYYY" MES AAAA t T AAAA ww WK YYYY	Aplicável apenas se um armazenamento customizado tiver sido especificado.

Tabela 54. Propriedades de variablefilenode (continuação).

Propriedades de 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	*field*	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 55. Propriedades de xmlimportnode.

Propriedades de 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	*sinalizador*	Especifica se deve também ler arquivos XML a partir de todos os subdiretórios do diretório especificado.
full_filename	*sequência*	(necessário) Caminho e nome do arquivo completos do arquivo XML a ser importado (se read = single).
directory_name	*sequência*	(necessário) Caminho e nome do diretório a partir 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.

Tabela 55. Propriedades de xmlimportnode (continuação).

Propriedades de xmlimportnode	Tipo de dados	Descrição da propriedade
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.

Propriedades de dataviewimport

O nó Visualização de Dados importa dados da Visualização de Dados no IBM SPSS Modeler.

exemplo

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

dvnode = stream.createAt("dataviewimport", "Data View", 96, 96)
dvnode.setPropertyValue("analytic_data_source",
["","/folder/adv", "LATEST"])
dvnode.setPropertyValue("table_name", ["","com.ibm.spss.Table"])
dvnode.setPropertyValue("data_access_plan",
["","DataAccessPlan"])
dvnode.setPropertyValue("optional_attributes",
[["","NewDerivedAttribute"]])
dvnode.setPropertyValue("include_xml", True)
dvnode.setPropertyValue("include_xml_field", "xml_data")
```

Tabela 56. Propriedades de dataviewimport

Propriedades de dataviewimport	Tipo de dados	Descrição da propriedade
analytic_data_source	*string*	O objeto de Visualização de Dados Analíticos armazenado no IBM SPSS Collaboration and Deployment Services. O nome do caminho e o rótulo de versão para a versão a ser utilizada. ["Object ID","Full path", "Version"]
table_name	*string*	A tabela de visualização de dados utilizada na Visualização de Dados Analíticos. O nome da tabela deve ser qualificado como pacote. É possível obter o pacote ao exportar o BOM a partir do cliente IBM SPSS Collaboration and Deployment Services Deployment Manager e procurar no arquivo default.bom no archive zip exportado. O nome do pacote deve ser sempre o mesmo, a menos que o BOM tenha sido importado do IBM Operational Decision Management (iLOG). ["Object ID", "Name"]

Tabela 56. Propriedades de dataviewimport (continuação)

Propriedades de **dataviewimport**	Tipo de dados	Descrição da propriedade
`data_access_plan`	*string*	O plano de acesso a dados utilizado para fornecer os dados para a Visualização de Dados Analíticos. `["Object ID","Name"]`
`optional_attributes`	*string*	Uma lista de atributos derivados a ser incluída. `[["ID1","Name1"], ["ID2", "Name2"]]`
`include_xml`	*boolean*	`True` se um campo com dados de instância XOM tiver que ser incluído. A menos que os nós do IBM Analytical Decision Management iLOG sejam utilizados, a configuração recomendada é `false`. Ativar isso pode incluir muito processamento extra.
`include_xml_field`	*string*	O nome do campo para incluir quando `include_xml` for configurado como `true`.

Capítulo 10. Propriedades do Nó de Operações de Registro

Propriedades de 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 57. Propriedades de appendnode.

Propriedades de appendnode	Tipo de dados	Descrição da propriedade
match_by	Posição Nome	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	*sinalizador*	Ativa a sensibilidade de maiúsculas e minúsculas ao corresponder nomes de campo.
include_fields_from	Main All	
create_tag_field	*sinalizador*	
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 58. Propriedades de aggregatenode.

Propriedades de aggregatenode	Tipo de dados	Descrição da propriedade
keys	*list*	Lista campos que podem ser utilizados como chaves para agregação. Por exemplo, se Sex e Region forem os campos-chave, cada combinação exclusiva de M e F com regiões N e S (quatro combinações exclusivas) terá um registro agregado.
contiguous	*flag*	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 de 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	*flag*	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 pedido quando a agregação é executada no Analytic Server
bin_count	*integer*	Especifica o número de categorias a serem utilizadas na aproximação

Propriedades de 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 onde uma condição for true 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 59. Propriedades de balancenode.

Propriedades de 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	*sinalizador*	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 (utilizando aspas duplas) forem 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 maior clareza.

Propriedades de 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 60. Propriedades do nó Space-Time-Boxes

Propriedades de derive_stbnode	Tipo de dados	Descrição da propriedade
mode	IndividualRecords Hangouts	
latitude_field	*campo*	
longitude_field	*campo*	
timestamp_field	*campo*	

Tabela 60. Propriedades do nó Space-Time-Boxes (continuação)

Propriedades de derive_stbnode	Tipo de dados	Descrição da propriedade
hangout_density	*density*	Uma única densidade. Consulte densities para obter 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	*integer*	O valor de número inteiro válido mínimo é 2.
qualifying_pct	*integer*	Deve estar no intervalo de 1 a 100.
add_extension_as	Prefixo Sufixo	
name_extension	*string*	

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 61. Propriedades de distinctnode.

Propriedades de 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	*sinalizador*	Cria um campo extra que especifica quantos registros de entrada foram agregados para formar cada registro agregado.
count_field	*string*	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	*sinalizador*	
low_distinct_key_count	*sinalizador*	Especifica que você tem apenas um pequeno número de registros e/ou um pequeno número de valores exclusivos de um ou mais campos-chave.
keys_pre_sorted	*sinalizador*	Especifica que todos os registros com os mesmos valores da chave são agrupados na entrada.
disable_sql_generation	*sinalizador*	

Exemplo para propriedade composite_value

A propriedade composite_value tem a seguinte forma geral:

```
node.setKeyedPropertyValue("composite_value", FIELD, FILLOPTION)
```

FILLOPTION has the form [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 a propriedade `composite_values`

A propriedade `composite_values` tem a seguinte forma geral:

```
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 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 do cliente internos e dados demográficos adquiridos.

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 62. Propriedades de mergenode.

Propriedades de mergenode	Tipo de dados	Descrição da propriedade
method	Order Chaves Condição 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 para Condition, especifica a condição para incluir ou descartar registros.
key_fields	*lista*	
common_keys	*flag*	
join	Inner FullOuter PartialOuter Anti	
outer_join_tag.n	*flag*	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	*flag*	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 é um pouco diferente da propriedade outer_join_tag (sinalizador versus sequência) porque somente um conjunto de dados de entrada pode ser especificado.
use_existing_sort_keys	*flag*	Especifica se as entradas já estão classificadas por um ou mais campos-chave.
existing_sort_keys	[[*'string'*, 'Ascending'] \ [*'string''*, 'Descending']]	Especifica os campos que já estiverem classificados e a direção na qual eles são classificados.
primary_dataset	*string*	Se method for Rankedcondition, selecione o conjunto de dados primário na mesclagem. Isso pode ser considerado como o lado esquerdo de uma mesclagem de junção externa.

Tabela 62. Propriedades de mergenode (continuação).

Propriedades de mergenode	Tipo de dados	Descrição da propriedade
add_tag_duplicate	*Boolean*	Se method for Rankedcondition, e este for configurado para Y, se o conjunto de dados mesclado resultante contiver diversos campos com o mesmo nome a partir de diferentes origens de dados, as respectivas tags das origens de dados serão incluídas no início dos cabeçalhos de coluna do campo.
merge_condition	*string*	
ranking_expression	*string*	
Num_matches	*integer*	O número de correspondências a serem retornadas, com base no merge_condition e no ranking_expression. Mínimo 1, máximo 100.

Propriedades de rfmaggregatenode

O nó Recency, Frequency, Monetary (RFM) Aggregate permite selecionar os dados transacionais históricos dos clientes, eliminar quaisquer dados não utilizados e combinar todos os dados da transação restantes dos clientes em uma única linha que lista quando eles interagiram pela última vez com você, quantas transações eles realizaram 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 63. Propriedades de rfmaggregatenode.

Propriedades de rfmaggregatenode	Tipo de dados	Descrição da propriedade
relative_to	Fixo Today	Especifique a data a partir da qual a recência das transações será calculada.
reference_date	*date*	Disponível apenas se Fixed for escolhido em relative_to.
contiguous	*flag*	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	*field*	Especifique o campo a ser utilizado para identificar o cliente e suas transações.
date_field	*field*	Especifique o campo de data a ser utilizado para calcular a recência.
value_field	*field*	Especifique o campo a ser utilizado para calcular o valor monetário.
extension	*sequência*	Especifique um prefixo ou sufixo para duplicar campos agregados.

Tabela 63. Propriedades de rfmaggregatenode (continuação).

Propriedades de rfmaggregatenode	Tipo de dados	Descrição da propriedade
add_as	Suffix Prefix	Especifique se extension deve ser incluído como um sufixo ou um prefixo.
discard_low_value_records	*flag*	Ativa 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 relacionadas ao campo value selecionado.
only_recent_transactions	*sinalizador*	Ativa a utilização das configurações de specify_transaction_date ou transaction_within_last.
specify_transaction_date	*sinalizador*	
transaction_date_after	*date*	Disponível apenas 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*	Disponível apenas 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.
transaction_scale	Days Weeks Months Years	Disponível apenas 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	*flag*	Exibe a data da segunda transação mais recente para cada cliente.
save_r3	*sinalizador*	Disponível apenas se save_r2 for selecionado. Exibe a data da terceira transação mais recente para cada cliente.

Propriedades de Rprocessnode

O nó Processamento R permite selecionar dados a partir de um fluxo do IBM(r) SPSS(r) Modeler e modificar os dados utilizando 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 64. Propriedades de Rprocessnode.

Propriedades de Rprocessnode	Tipo de dados	Descrição da propriedade
syntax	*string*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*sinalizador*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*sinalizador*	

Propriedades de samplenode

O nó Amostra seleciona um subconjunto de registros. Diversos tipos de amostra são suportados, 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 65. Propriedades de samplenode.

Propriedades de 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	*número*	Inclui ou descarta a cada *n*º registro.

Tabela 65. Propriedades de samplenode (continuação).

Propriedades de samplenode	Tipo de dados	Descrição da propriedade
rand_pct	*número*	Especifique a porcentagem de registros a incluir ou descartar.
use_max_size	*sinalizador*	Ativar o uso da configuração de maximum_size.
maximum_size	*número inteiro*	Especifica a maior amostra a ser incluída ou descartada do fluxo de dados. Esta opção é redundante e, portanto, desativada quando First e Include forem especificados.
set_random_seed	*sinalizador*	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	Proporções Contagens	
sample_size_proportions	Fixo Custom Variable	
sample_size_counts	Fixo Custom Variable	
fixed_proportions	*número*	
fixed_counts	*número inteiro*	
variable_proportions	*campo*	
variable_counts	*campo*	
use_min_stratum_size	*sinalizador*	
minimum_stratum_size	*número inteiro*	Essa opção se aplica apenas quando uma amostra Complexa é obtida com Sample units=Proportions.
use_max_stratum_size	*sinalizador*	
maximum_stratum_size	*número inteiro*	Essa opção se aplica apenas quando uma amostra Complexa é obtida com Sample units=Proportions.
clusters	*campo*	
stratify_by	*[field1 ... fieldN]*	
specify_input_weight	*sinalizador*	
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 campo de estratificação.
default_proportion	*número*	

Tabela 65. Propriedades de samplenode (continuação).

Propriedades de samplenode	Tipo de dados	Descrição da propriedade
sizes_counts	[[string *string value*][string *string value*]...]	Especifica um valor para cada combinação possível de valores de campo de estratificação. Uso é semelhante a sizes_proportions, mas especificando um número inteiro ao invés de uma proporção.
default_count	*número*	

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 66. Propriedades de selectnode.

Propriedades de 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 67. Propriedades de sortnode.

Propriedades de 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	*sinalizador*	Especifica a ordem de classificação padrão.
use_existing_keys	*sinalizador*	Especifica se a classificação é otimizada utilizando a ordem de classificação anterior para campos que já estiverem classificados.

Tabela 67. Propriedades de sortnode (continuação).

Propriedades de sortnode	Tipo de dados	Descrição da propriedade
existing_keys		Especifica os campos que já estiverem classificados e a direção na qual eles são classificados. Utiliza o mesmo formato que a propriedade keys.

Propriedades de streamingts

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 68. Propriedades de streamingts.

Propriedades de streamingts	Tipo de dados	Descrição da propriedade
custom_fields	*flag*	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*]	
method	ExpertModeler Exsmooth Arima	
calculate_conf	*flag*	
conf_limit_pct	*real*	
use_time_intervals_node	*flag*	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*	

Tabela 68. Propriedades de streamingts (continuação).

Propriedades de **streamingts**	Tipo de dados	Descrição da propriedade
interval_type	Periods Years Quarters Months WeeksNonPeriodic DaysNonPeriodic HoursNonPeriodic MinutesNonPeriodic SecondsNonPeriodic	
events	*campos*	
expert_modeler_method	AllModels Exsmooth Arima	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	
exsmooth_transformation_type	Nenhum SquareRoot NaturalLog	
arima_p	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_d	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_q	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_sp	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_sd	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_sq	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais
arima_transformation_type	Nenhum SquareRoot NaturalLog	Mesma propriedade do nó de modelagem de Séries Temporais
arima_include_constant	*flag*	Mesma propriedade do nó de modelagem de Séries Temporais

Tabela 68. Propriedades de streamingts (continuação).

Propriedades de streamingts	Tipo de dados	Descrição da propriedade
tf_arima_p.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_d.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_q.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_sp.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_sd.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_sq.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_delay.*fieldname*	*integer*	Mesma propriedade do nó de modelagem de Séries Temporais. Para funções de transferência.
tf_arima_transformation_type.*fieldname*	Nenhum SquareRoot NaturalLog	
arima_detect_outlier_mode	Nenhum Automático	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
deployment_force_rebuild	*flag*	
deployment_rebuild_mode	Contagem Porcentagem	
deployment_rebuild_count	*number*	
deployment_rebuild_pct	*number*	
deployment_rebuild_field	*<campo>*	

Capítulo 11. Propriedades do Nó de Operações de Campo

Propriedades de 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 69. Propriedades de anonymizenode

Propriedades de anonymizenode	Tipo de dados	Descrição da propriedade
enable_anonymize	*flag*	Quando configurado para True, ativa a anonimização de valores de campo (equivalente a selecionar em **Sim** para esse campo na coluna Valores Anonimizados).
use_prefix	*flag*	Quando configurado para True, um prefixo customizado será utilizado 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	*string*	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	*flag*	Quando configurado para True, o valor inicial especificado é utilizado (se transformation também for configurado para Random).
random_seed	*integer*	Quando set_random_seed é configurado para True, esse é o valor inicial para o número aleatório.
escala	*number*	Quando transformation é configurado para Fixed, esse valor é usado para "scale by". O valor máximo da escala normalmente é 10, mas poderá ser reduzido para evitar estouro.

Tabela 69. Propriedades de anonymizenode (continuação)

Propriedades de anonymizenode	Tipo de dados	Descrição da propriedade
traduzir	*number*	Quando transformation é configurado para Fixed, esse valor é usado para "translate". O valor máximo da conversão normalmente é 1000, mas poderá ser reduzido para evitar estouro.

Propriedades de 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 70. Propriedades de autodataprepnode

Propriedades de autodataprepnode	Tipo de dados	Descrição da propriedade
objective	Balanced Speed Accuracy Custom	
custom_fields	*flag*	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.
use_frequency	*flag*	
frequency_field	*campo*	
use_weight	*flag*	
weight_field	*field*	
excluded_fields	Filter None	
if_fields_do_not_match	StopExecution ClearAnalysis	
prepare_dates_and_times	*flag*	Controla o acesso a todos os campos de data e hora
compute_time_until_date	*flag*	

Tabela 70. Propriedades de autodataprepnode (continuação)

Propriedades de autodataprepnode	Tipo de dados	Descrição da propriedade
reference_date	Today Fixed	
fixed_date	*date*	
units_for_date_durations	Automatic Fixed	
fixed_date_units	Years Months Days	
compute_time_until_time	*flag*	
reference_time	CurrentTime Fixed	
fixed_time	*time*	
units_for_time_durations	Automatic Fixed	
fixed_date_units	Hours Minutes Seconds	
extract_year_from_date	*flag*	
extract_month_from_date	*flag*	
extract_day_from_date	*flag*	
extract_hour_from_time	*flag*	
extract_minute_from_time	*flag*	
extract_second_from_time	*flag*	
exclude_low_quality_inputs	*flag*	
exclude_too_many_missing	*flag*	
maximum_percentage_missing	*number*	
exclude_too_many_categories	*flag*	
maximum_number_categories	*number*	
exclude_if_large_category	*flag*	
maximum_percentage_category	*number*	
prepare_inputs_and_target	*flag*	
adjust_type_inputs	*flag*	
adjust_type_target	*flag*	
reorder_nominal_inputs	*flag*	
reorder_nominal_target	*flag*	
replace_outliers_inputs	*flag*	
replace_outliers_target	*flag*	
replace_missing_continuous_inputs	*flag*	
replace_missing_continuous_target	*flag*	
replace_missing_nominal_inputs	*flag*	
replace_missing_nominal_target	*flag*	
replace_missing_ordinal_inputs	*flag*	

Tabela 70. Propriedades de autodataprepnode (continuação)

Propriedades de **autodataprepnode**	Tipo de dados	Descrição da propriedade
replace_missing_ordinal_target	*flag*	
maximum_values_for_ordinal	*number*	
minimum_values_for_continuous	*number*	
outlier_cutoff_value	*number*	
outlier_method	Replace Delete	
rescale_continuous_inputs	*flag*	
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	*flag*	
target_final_mean	*number*	
target_final_sd	*number*	
transform_select_input_fields	*flag*	
maximize_association_with_target	*flag*	
p_value_for_merging	*number*	
merge_ordinal_features	*flag*	
merge_nominal_features	*flag*	
minimum_cases_in_category	*number*	
bin_continuous_fields	*flag*	
p_value_for_binning	*number*	
perform_feature_selection	*flag*	
p_value_for_selection	*number*	
perform_feature_construction	*flag*	
transformed_target_name_extension	*string*	
transformed_inputs_name_extension	*string*	
constructed_features_root_name	*string*	
years_duration_ name_extension	*string*	
months_duration_ name_extension	*string*	
days_duration_ name_extension	*string*	
hours_duration_ name_extension	*string*	
minutes_duration_ name_extension	*string*	
seconds_duration_ name_extension	*string*	
year_cyclical_name_extension	*string*	
month_cyclical_name_extension	*string*	
day_cyclical_name_extension	*string*	
hour_cyclical_name_extension	*string*	

Tabela 70. Propriedades de autodataprepnode (continuação)

Propriedades de autodataprepnode	Tipo de dados	Descrição da propriedade
minute_cyclical_name_extension	*string*	
second_cyclical_name_extension	*string*	

Propriedades de astimeintervalsnode

O nó Intervalos de Tempo original não é compatível com o Analytic Server (AS). O nó Intervalos de Tempo do AS (novo no SPSS Modeler release 17.0) contém um subconjunto das funções do nó Intervalos de Tempo existente que pode ser usado com o Analytic Server.

Utilize o nó Intervalos de Tempo do AS para especificar intervalos e derivar um novo campo de tempo para estimativa ou previsão. Uma variedade completa de intervalos de tempo é suportada, desde segundos até anos.

Tabela 71. Propriedades de astimeintervalsnode

Propriedades de 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.
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 72. Propriedades de binningnode

Propriedades de 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 Ótimo	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.
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	*integer*	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	*string*	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	*sinalizador*	Gera quatro categorias de quantil, cada uma contendo 25% dos casos.
tile5	*sinalizador*	Gera cinco categorias de quintil.
tile10	*sinalizador*	Gera 10 categorias de decil.
tile20	*sinalizador*	Gera 20 categorias de vingtile.
tile100	*sinalizador*	Gera 100 categorias de percentil.
use_custom_tile	*sinalizador*	
custom_tile_name_extension	*string*	A extensão padrão é *_TILEN*.
custom_tile_add_as	Sufixo Prefixo	
custom_tile	*integer*	
equal_count_method	RecordCount ValueSum	O método RecordCount visa designar um número igual de registros para cada categoria, ao passo que ValueSum designa registros para que a soma dos valores em cada categoria seja igual.

Tabela 72. Propriedades de binningnode (continuação)

Propriedades de 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	Essa propriedade inclui Ascending (o valor mais baixo é marcado como 1) ou Descending (o valor mais alto é marcado como 1).
rank_add_as	Suffix Prefix	Essa opção se aplica à classificação, à classificação fracional e à classificação de porcentagem.
rank	*sinalizador*	
rank_name_extension	*string*	A extensão padrão é *_RANK*.
rank_fractional	*sinalizador*	Classifica casos em que o valor do novo campo é igual à classificação dividido pela soma dos pesos dos casos não desconhecidos. As classificações fracionárias entram no intervalo de 0 a 1.
rank_fractional_name_ extension	*string*	A extensão padrão é *_F_RANK*.
rank_pct	*sinalizador*	Cada classificação é dividida pelo número de registros com valores válidos e multiplicada por 100. A porcentagem das classificações fracionárias entra no intervalo de 1 a 100.
rank_pct_name_extension	*string*	A extensão padrão é *_P_RANK*.
sdev_name_extension	*string*	
sdev_add_as	Sufixo Prefixo	
sdev_count	Um Dois Árvore	
optimal_name_extension	*string*	A extensão padrão é *_OPTIMAL*.
optimal_add_as	Sufixo Prefixo	
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	*sinalizador*	Especifica que quaisquer categorias com contagens de caso pequenas serão incluídas em uma categoria vizinha maior.
optimal_small_bin_threshold	*integer*	
optimal_pre_bin	*sinalizador*	Indica que a pré-categorização do conjunto de dados deve ocorrer.
optimal_max_bins	*integer*	Especifica um limite superior para evitar a criação de um grande número de categorias excessivamente.
optimal_lower_end_point	Inclusive Exclusive	

Tabela 72. Propriedades de binningnode (continuação)

Propriedades de binningnode	Tipo de dados	Descrição da propriedade
optimal_first_bin	Unbounded Bounded	
optimal_last_bin	Unbounded Bounded	

Propriedades de derivenode

O nó Derivar modifica valores de dados ou cria novos campos a partir de um ou mais campos existentes. Ele cria campos do tipo fórmula, sinalizador, 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 supõe que há duas colunas numéricas chamadas `XPos` e `YPos` que representam as coordenadas X e Y de um ponto (por exemplo, onde um evento ocorreu). 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 73. Propriedades de derivenode

Propriedades de derivenode	Tipo de dados	Descrição da propriedade
new_name	*string*	Nome do novo campo.
mode	Single Multiple	Especifica campos únicos ou múltiplos.

Tabela 73. Propriedades de derivenode (continuação)

Propriedades de derivenode	Tipo de dados	Descrição da propriedade
campos	*lista*	Usado no modo Multiple apenas para selecionar diversos campos.
name_extension	*string*	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	*string*	A expressão para calcular um novo valor de campo em um nó Derivar.
flag_expr	*string*	
flag_true	*string*	
flag_false	*string*	
set_default	*string*	
set_value_cond	*string*	Estruturada para fornecer a condição associada a um determinado valor.
state_on_val	*string*	Especifica o valor para o novo campo quando a condição On for atendida.
state_off_val	*string*	Especifica o valor para o novo campo quando a condição Off for atendida.
state_on_expression	*string*	
state_off_expression	*string*	
state_initial	On Desligado	Designa 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	*string*	
count_inc_condition	*string*	
count_inc_expression	*string*	
count_reset_condition	*string*	
cond_if_cond	*string*	
cond_then_expr	*string*	
cond_else_expr	*string*	
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. A função setter pode ser transmitida para uma sequência ou para um dos valores MeasureType. A função getter sempre retornará nos valores MeasureType.

Tabela 73. Propriedades de derivenode (continuação)

Propriedades de derivenode	Tipo de dados	Descrição da propriedade
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 Polígono 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	*boolean*	Para campos geoespaciais, essa propriedade define se esse campo tem um sistema de coordenadas
coordinate_system	*string*	Para campos geoespaciais, essa propriedade define o sistema de coordenadas para este campo

Propriedades de ensemblenode

O nó Combinação combina dois ou mais nuggets do modelo para obter predições mais precisas do que pode ser obtido a partir de qualquer modelo individual.

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 74. Propriedades de ensemblenode.

Propriedades de 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	*sinalizador*	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 74. Propriedades de ensemblenode (continuação).

Propriedades de 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	*sinalizador*	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 espaços em branco ou valores nulos por um valor específico. Um nó de Preenchimento é geralmente utilizado em conjunto com um nó Tipo para substituir valores omissos.

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 75. Propriedadades de fillernode

Propriedades de fillernode	Tipo de dados	Descrição da propriedade
campos	*lista*	Campos do conjunto de dados cujos valores serão examinados e substituídos.
replace_mode	Always Condicional Blank Null BlankAndNull	É possível substituir todos os valores, valores em branco ou valores nulos ou substituir com base em uma condição especificada.
condition	*string*	
replace_with	*string*	

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 transmita os campos *Age* e *Sex* e descarte todos os outros. Agora suponha que você execute o mesmo script novamente, mas nomeando dois campos diferentes:

```
node = modeler.script.stream().create("filter", "Filter")
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, transmitindo um total de quatro campos (*Age*, *Sex*, *BP* e *Na*). Em outras palavras, reconfigurar o valor de `default_include` para `False` não reconfigura automaticamente todos os campos.

Como alternativa, se agora você alterar `default_include` para `True`, seja utilizando um script ou na caixa de diálogo do nó Filtro, isso inverterá o comportamento fazendo que os quatro campos listados acima sejam descartados ao invés de incluídos. Quando estiver em dúvida, experimentar os controles na caixa de diálogo do nó Filtro poderá ser útil para entender essa interação.

Tabela 76. Propriedades de filternode

Propriedades de `filternode`	Tipo de dados	Descrição da propriedade
`default_include`	*sinalizador*	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.

Tabela 76. Propriedades de filternode (continuação)

Propriedades de filternode	Tipo de dados	Descrição da propriedade
include	*sinalizador*	Propriedade definida como chave para inclusão e remoção de campo.
new_name	*string*	

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 utilizar um nó Histórico, você pode querer classificar os dados utilizando um nó Classificar.

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 77. Propriedades de historynode

Propriedades de historynode	Tipo de dados	Descrição da propriedade
campos	*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.
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 78. Propriedades de partitionnode

Propriedades de partitionnode	Tipo de dados	Descrição da propriedade
new_name	*string*	Nome do campo de partição gerado pelo nó.
create_validation	*sinalizador*	Especifica se uma partição de validação deve ser criada.
training_size	*integer*	Porcentagem de registros (0 a 100) a ser alocada para a partição de treinamento.
testing_size	*integer*	Porcentagem de registros (0 a 100) a ser alocada para a partição de teste.
validation_size	*integer*	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	*string*	Rótulo para a partição de treinamento.
testing_label	*string*	Rótulo para a partição de teste.
validation_label	*string*	Rótulo para a partição de validação. Ignorado se uma partição de validação não for criada.
value_mode	System SystemAndLabel Rótulo	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, pelo rótulo Training ou por uma combinação dos dois, 1_Training.
set_random_seed	*Booleano*	Especifica se um valor inicial aleatório especificado pelo usuário deve ser utilizado.
random_seed	*integer*	Um valor inicial aleatório especificados pelo usuário. Para este valor a ser utilizado, set_random_seed deve ser configurado para 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 este valor seja utilizado, enable_sql_generation deverá ser configurado como True.

Propriedades de 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.

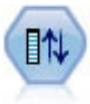
exemplo

```
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 79. Propriedades de reclassifynode

Propriedades de **reclassifynode**	Tipo de dados	Descrição da propriedade
mode	Single Multiple	Simple reclassifica as categorias para um campo. Multiple ativa opções permitindo transformação de mais de um campo por vez.
replace_field	*sinalizador*	
campo	*string*	Usado apenas no modo Single.
new_name	*string*	Usado apenas no modo Single.
campos	*[field1 field2 ... fieldn]*	Usado apenas no modo Multiple.
name_extension	*string*	Usado apenas no modo Multiple.
add_as	Sufixo Prefixo	Usado apenas no modo Multiple.
reclassify	*string*	Propriedade estruturada para valores de campo.
use_default	*sinalizador*	Utiliza o valor padrão.
default	*string*	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 grandes 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 80. Propriedades de reordernode

Propriedades de reordernode	Tipo de dados	Descrição da propriedade
mode	Custom Automático	É possível classificar valores automaticamente ou especificar uma ordem customizada.
sort_by	Nome Tipo Armazenamento	
ascending	*flag*	
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 81. Propriedades de reprojectnode

Propriedades de reprojectnode	Tipo de dados	Descrição da propriedade
reproject_fields	*[field1 field2 ... fieldn]*	Lista todos os campos que devem ser reprojetados.
reproject_type	Streamdefault Especificar	Escolha como deseja reprojetar os campos.
coordinate_system	*string*	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 chamado *tipo de pagamento* com valores de *crédito*, *dinheiro* e *débito*, três novos campos seriam criados (*crédito*, *dinheiro* e *débito*), cada um dos quais podendo 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 82. Propriedades de restructurenode

Propriedades de restructurenode	Tipo de dados	Descrição da propriedade
fields_from	[*category category category*] all	
include_field_name	*sinalizador*	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 deseja utilizar (veja abaixo). Com Flags, os valores são sinalizadores numéricos.
value_fields	*lista*	Necessário se value_mode for OtherFields. Especifica quais campos deseja utilizar como campos de valor.

Propriedades de rfmanalysisnode

O nó Recency, Frequency, Monetary (RFM) Analysis permite determinar de modo quantitativo quais podem ser seus melhores clientes ao examinar quando foi a última vez que eles compraram de você (recência), com que frequência eles compraram (frequência) e quanto eles gastaram em todas as transações (monetário).

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 83. Propriedades de rfmanalysisnode

Propriedades de 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	*integer*	Especifica o número de categorias de recência a serem geradas.
recency_weight	*número*	Especifica o peso a ser aplicado aos dados de recência. O padrão é 100.
frequency_bins	*integer*	Especifica o número de categorias de frequência a serem geradas.
frequency_weight	*número*	Especifica o peso a ser aplicado aos dados de frequência. O padrão é 10.
monetary_bins	*integer*	Especifica o número de categorias monetárias a serem geradas.
monetary_weight	*número*	Especifica o peso a ser aplicado aos dados monetários. O padrão é 1.

Tabela 83. Propriedades de rfmanalysisnode (continuação)

Propriedades de rfmanalysisnode	Tipo de dados	Descrição da propriedade
tied_values_method	Next Current	Especifica em qual categoria os dados de valor vinculados devem ser colocados.
recalculate_bins	Always IfNecessary	
add_outliers	*sinalizador*	Disponível apenas se recalculate_bins for configurado para 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	Recência Frequency Monetário	
recency_thresholds	*valor valor*	Disponível apenas se recalculate_bins for configurado para 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 para Always.
monetary_thresholds	*valor valor*	Disponível apenas se recalculate_bins for configurado para 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 84. Propriedades de settoflagnode

Propriedades de settoflagnode	Tipo de dados	Descrição da propriedade
fields_from	[*category category category*] all	
true_value	*string*	Especifica o valor true utilizado pelo nó ao configurar um sinalizador. O padrão é T.
false_value	*string*	Especifica o valor false utilizado pelo nó ao configurar um sinalizador. O padrão é F.
use_extension	*flag*	Utiliza uma extensão como um sufixo ou prefixo para o novo campo de sinalização.
extension	*string*	
add_as	Suffix Prefix	Especifica se a extensão é incluída como um prefixo ou sufixo.
aggregate	*flag*	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	*list*	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. Esse nó requer uma cópia licenciada do IBM SPSS Statistics.

As propriedades desse nó são descritas em "Propriedades de statisticstransformnode" na página 299.

Propriedades de timeintervalsnode

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 85. Propriedades de timeintervalsnode.

Propriedades de **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	*field*	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	*integer*	Para tipos de intervalo onde aplicável, o ano em que cai o primeiro intervalo.
quarter_start	*integer*	Para tipos de intervalo onde aplicável, o trimestre em que cai o primeiro intervalo.
month_start	Janeiro Fevereiro Março Abril Maio Junho Julho Agosto Setembro Outubro Novembro Dezembro	
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.

Tabela 85. Propriedades de timeintervalsnode (continuação).

Propriedades de timeintervalsnode	Tipo de dados	Descrição da propriedade
fiscal_year_begins	Janeiro Fevereiro Março Abril Maio Junho Julho Agosto Setembro Outubro Novembro Dezembro	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.
day_begins_hour	*integer*	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*	
field_name_extension_as_prefix	*flag*	

Tabela 85. Propriedades de timeintervalsnode (continuação).

Propriedades de **timeintervalsnode**	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" MES AAAA t T AAAA ss SM AAAA	
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"	
aggregate	Média Sum Modo 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.

Tabela 85. Propriedades de timeintervalsnode (continuação).

Propriedades de timeintervalsnode	Tipo de dados	Descrição da propriedade
agg_range_default	Média Sum Modo Min Max	Especifica a função padrão a ser utilizada ao agregar os campos contínuos.
agg_set_default	Modo First Last	Especifica a função padrão a ser utilizada ao agregar os campos nominais.
agg_flag_default	TrueIfAnyTrue Modo 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	*flag*	
estimation_to_end	*flag*	
estimation_start_offset	*número inteiro*	
estimation_num_holdouts	*número inteiro*	
create_future_records	*flag*	
num_future_records	*número inteiro*	
create_future_field	*flag*	
future_field_name	*string*	

Propriedades de transposenode

O nó Transpor troca os dados em linhas e colunas para que os registros se tornem campos e os 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 86. Propriedades de transposenode

Propriedades de transposenode	Tipo de dados	Descrição da propriedade
transposed_names	Prefix Leitura	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	*string*	
num_new_fields	*integer*	Ao utilizar um prefixo, especifica o número máximo de novos campos a serem criados.
read_from_field	*campo*	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	*integer*	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	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*	Especifica os campos a serem transpostos quando a opção Custom for utilizada.
id_field_name	*campo*	

Propriedades de typenode

O nó Tipo especifica metadados e propriedades de campo. Por exemplo, é possível especificar um nível de medição (contínua, nominal, ordinal ou sinalizador) para cada campo, configurar opções para manipular valores omissos e nulos do sistema, configurar a função de um campo para fins de modelagem, especificar rótulos de campo e de 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 87. Propriedades de typenode.

Propriedades de typenode	Tipo de dados	Descrição da propriedade
direction	Input Target Ambos Nenhum Partição 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 Sinalizador Set Sem tipo Discreto OrderedSet Default	Nível de medição do campo (anteriormente chamado de "tipo" de campo). Configurar o Type para Default limpará qualquer configuração do parâmetro values e, se value_mode possuir o valor Specify, ele será reconfigurado para Read. Se value_mode for configurado para Pass ou Read, configurar o 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 - > OrderedSet Nominal-> Discreto Ordinal- > Categórico
storage	Unknown String Integer Real Time Date Registro de data e hora	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*. Configurar esta propriedade configura automaticamente a propriedade value_mode para Specify.
value_mode	Leitura Pass Read+ Current Specify	Determina como os valores são configurados. Observe que não é possível configurar essa propriedade para Specify diretamente; para utilizar valores específicos, configure a propriedade values.

Tabela 87. Propriedades de typenode (continuação).

Propriedades de typenode	Tipo de dados	Descrição da propriedade
extend_values	*flag*	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.
enable_missing	*flag*	Quando configurado para T, ativa o rastreamento de valores omissos para o campo.
missing_values	[*value value ...*]	Especifica valores de dados que denotam dados ausentes.
range_missing	*flag*	Especifica se um intervalo de valores omissos (em branco) é definido para um campo.
missing_lower	*string*	Quando range_missing for true, especifica o limite inferior do intervalo de valores omissos.
missing_upper	*string*	Quando range_missing for true, especifica o limite superior do intervalo de valores omissos.
null_missing	*flag*	Quando configurado para T, *nulos* (valores indefinidos que são exibidos como $null$ no software) são considerados valores omissos.
whitespace_ missing	*flag*	Quando configurado para T, os valores contendo apenas espaços em branco (espaços, tabulações e novas linhas) são considerados valores omissos.
description	*string*	Especifica a descrição de um campo.
value_labels	*[[Value LabelString] [Value LabelString] ...]*	Utilizado para especificar rótulos para pares de valores.
display_places	*integer*	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	*integer*	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.
decimal_separator	DEFAULT PERIOD COMMA	Configura o separador decimal para o campo (aplica-se apenas aos campos com armazenamento REAL).

Tabela 87. Propriedades de typenode (continuação).

Propriedades de typenode	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" MES AAAA t T AAAA ss SM AAAA	Configura o formato de data para o campo (aplica-se apenas aos campos com 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.
standard_places	*integer*	Configura o número de casas decimais para o campo quando exibido em formato padrão. Um valor de -1 utilizará o padrão de fluxo. Observe que o slot display_places existente também altera isso, mas agora foi descontinuado.
scientific_places	*integer*	Configura o número de casas decimais para o campo quando exibido no formato científico. Um valor de -1 utilizará o padrão de fluxo.

Tabela 87. Propriedades de typenode (continuação).

Propriedades de typenode	Tipo de dados	Descrição da propriedade
currency_places	*integer*	Configura o número de casas decimais para o campo quando exibido no formato de moeda. Um valor de -1 utilizará o padrão de fluxo.
grouping_symbol	DEFAULT NENHUM LOCALE PERIOD COMMA SPACE	Configura o símbolo de agrupamento para o campo.
column_width	*integer*	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.
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 definida como chave é semelhante a type por ela poder ser utilizada para definir a medida associada ao campo. A diferença é que, no script Python, a função setter também pode transmitir um dos valores de MeasureType, ao passo que a função 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 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 Polígono 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	*boolean*	Para campos geoespaciais, essa propriedade define se esse campo tem um sistema de coordenadas
coordinate_system	*string*	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	Essa propriedade definida como chave é semelhante a custom_storage por ela poder ser utilizada para definir o armazenamento de substituição para o campo. A diferença é que, no script Python, a função setter também pode transmitir um dos valores de StorageType, ao passo que a função getter sempre retornará nos valores StorageType.

Tabela 87. Propriedades de typenode (continuação).

Propriedades de typenode	**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`	*integer*	Para campos de lista, esta propriedade definida como chave especifica a profundidade do campo

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 88. Propriedades comuns do nó de gráfico

Propriedades comuns do nó de gráfico	**Tipo de dados**	**Descrição da propriedade**
`title`	*string*	Especifica o título. Exemplo: "Este é um título".
`caption`	*string*	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`	*string*	Especifica o caminho de destino e o nome de arquivo para a saída gerada a partir do nó de gráfico.
`use_graph_size`	*flag*	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 um gráfico 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 89. Propriedades de collectionnode

Propriedades de collectionnode	Tipo de dados	Descrição da propriedade
over_field	*campo*	
over_label_auto	*flag*	
over_label	*string*	
collect_field	*campo*	
collect_label_auto	*flag*	
collect_label	*string*	
three_D	*flag*	
by_field	*campo*	
by_label_auto	*flag*	
by_label	*string*	
operation	Sum Mean Min Max SDev	
color_field	*string*	
panel_field	*string*	
animation_field	*string*	

Tabela 89. Propriedades de collectionnode (continuação)

Propriedades de collectionnode	Tipo de dados	Descrição da propriedade
range_mode	Automático UserDefined	
range_min	*número*	
range_max	*número*	
bins	ByNumber ByWidth	
num_bins	*número*	
bin_width	*número*	
use_grid	*flag*	
graph_background	*color*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*color*	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 90. Propriedades de distributionnode

Propriedades de distributionnode	Tipo de dados	Descrição da propriedade
plotar	SelectedFields Flags	
x_field	*campo*	
color_field	*campo*	Campo de sobreposição.
normalize	*flag*	
sort_mode	ByOccurence Alfabético	
use_proportional_scale	*flag*	

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 na representação.

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 91. Propriedades de evaluationnode.

Propriedades de evaluationnode	Tipo de dados	Descrição da propriedade
chart_type	Gains Response Lift Profit ROI ROC	
inc_baseline	*sinalizador*	
field_detection_method	Metadata Nome	
use_fixed_cost	*sinalizador*	
cost_value	*número*	
cost_field	*sequência*	
use_fixed_revenue	*sinalizador*	
revenue_value	*número*	
revenue_field	*sequência*	
use_fixed_weight	*sinalizador*	
weight_value	*número*	
weight_field	*field*	

Tabela 91. Propriedades de evaluationnode (continuação).

Propriedades de evaluationnode	Tipo de dados	Descrição da propriedade
n_tile	Quartis Quintles Decis Vintis Percentis 1000-tiles	
cumulative	*sinalizador*	
style	Line Point	
point_type	Retângulo Ponto Triângulo Hexágono Mais Pentágono Estrela BowTie HorizontalDash VerticalDash IronCross Factory Casa Catedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Ventilador	
export_data	*sinalizador*	
data_filename	*sequência*	
delimiter	*sequência*	
new_line	*sinalizador*	
inc_field_names	*sinalizador*	
inc_best_line	*sinalizador*	
inc_business_rule	*sinalizador*	
business_rule_condition	*sequência*	
plot_score_fields	*sinalizador*	
score_fields	*[field1 ... fieldN]*	
target_field	*field*	
use_hit_condition	*sinalizador*	
hit_condition	*sequência*	
use_score_expression	*sinalizador*	
score_expression	*sequência*	
caption_auto	*sinalizador*	

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 92. Propriedades de graphboardnode

Propriedades de 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 ChoroplethCounts CoordinateMap CoordinateChoroplethMeans CoordinateChoroplethMedians CoordinateChoroplethSums CoordinateChoroplethValues CoordinateChoroplethCounts Dotplot Heatmap HexBinScatter Histogram Line LineChartMap LineOverlayMap Parallel Path Pie PieCountMap PieCounts PieMap PointOverlayMap PolygonOverlayMap Ribbon Scatterplot SPLOM Surface	Identifica o tipo de gráfico.
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*	

Tabela 92. Propriedades de graphboardnode (continuação)

Propriedades de graphboard	Tipo de dados	Descrição da propriedade
values_field	*campo*	
rows_field	*campo*	
columns_field	*campo*	
campos	*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	*string*	
panelcol_field	*string*	
animation_field	*string*	
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 93. Propriedades de histogramnode

Propriedades de histogramnode	Tipo de dados	Descrição da propriedade
campo	*campo*	
color_field	*campo*	
panel_field	*campo*	
animation_field	*campo*	

Tabela 93. Propriedades de histogramnode (continuação)

Propriedades de histogramnode	Tipo de dados	Descrição da propriedade
range_mode	Automático UserDefined	
range_min	*number*	
range_max	*number*	
bins	ByNumber ByWidth	
num_bins	*number*	
bin_width	*number*	
normalize	*flag*	
separate_bands	*flag*	
x_label_auto	*flag*	
x_label	*string*	
y_label_auto	*flag*	
y_label	*string*	
use_grid	*flag*	
graph_background	*color*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*color*	As cores do gráfico padrão são descritas no início desta seção.
normal_curve	*flag*	Indica se a curva de distribuição normal deve ser mostrada na saída.

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 94. Propriedades de multiplotnode

Propriedades de multiplotnode	Tipo de dados	Descrição da propriedade
x_field	*campo*	
y_fields	*lista*	
panel_field	*campo*	
animation_field	*campo*	
normalize	*flag*	
use_overlay_expr	*flag*	
overlay_expression	*string*	
records_limit	*number*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*flag*	
x_label	*string*	
y_label_auto	*flag*	
y_label	*string*	
use_grid	*flag*	
graph_background	*color*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*color*	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 95. Propriedades de plotnode.

Propriedades de plotnode	Tipo de dados	Descrição da propriedade
x_field	*field*	Especifica um rótulo customizado para o eixo *x*. Disponível apenas para rótulos.
y_field	*field*	Especifica um rótulo customizado para o eixo *y*. Disponível apenas para rótulos.
three_D	*sinalizador*	Especifica um rótulo customizado para o eixo *y*. Disponível apenas para os rótulos nos gráficos 3D.
z_field	*field*	
color_field	*field*	Campo de sobreposição.
size_field	*field*	
shape_field	*field*	
panel_field	*field*	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	*field*	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	*field*	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	Nenhum Alisador Function	Especifica se uma função de sobreposição ou um suavizador LOESS é exibido.
overlay_expression	*sequência*	Especifica a expressão usada quando overlay_type for configurado para Function.
style	Point Line	
point_type	Retângulo Ponto Triângulo Hexágono Mais Pentágono Estrela BowTie HorizontalDash VerticalDash IronCross Factory Casa Catedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Ventilador	
x_mode	Classificar Sobreposição AsRead	

Tabela 95. Propriedades de plotnode (continuação).

Propriedades de plotnode	Tipo de dados	Descrição da propriedade
x_range_mode	Automático UserDefined	
x_range_min	*número*	
x_range_max	*número*	
y_range_mode	Automático UserDefined	
y_range_min	*número*	
y_range_max	*número*	
z_range_mode	Automático UserDefined	
z_range_min	*número*	
z_range_max	*número*	
jitter	*sinalizador*	
records_limit	*número*	
if_over_limit	PlotBins PlotSample PlotAll	
x_label_auto	*sinalizador*	
x_label	*sequência*	
y_label_auto	*sinalizador*	
y_label	*sequência*	
z_label_auto	*sinalizador*	
z_label	*sequência*	
use_grid	*sinalizador*	
graph_background	*color*	As cores do gráfico padrão são descritas no início desta seção.
page_background	*color*	As cores do gráfico padrão são descritas no início desta seção.
use_overlay_expr	*sinalizador*	Descontinuado a favor de overlay_type.

Propriedades de timeplotnode

O nó Gráfico de Tempo exibe um ou mais conjuntos de dados de séries temporais. Geralmente, um nó Intervalos de Tempo deverá ser usado primeiro para criar um campo *TimeLabel* que será utilizado 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 96. Propriedades de timeplotnode.

Propriedades de **timeplotnode**	Tipo de dados	Descrição da propriedade
plot_series	Série Models	
use_custom_x_field	*sinalizador*	
x_field	*campo*	
y_fields	*lista*	
panel	*sinalizador*	
normalize	*sinalizador*	
line	*sinalizador*	
points	*sinalizador*	
point_type	Retângulo Ponto Triângulo Hexágono Mais Pentágono Estrela BowTie HorizontalDash VerticalDash IronCross Factory Casa Catedral OnionDome ConcaveTriangle OblateGlobe CatEye FourSidedPillow RoundRectangle Ventilador	
smoother	*sinalizador*	Será possível incluir suavizadores no gráfico somente se você configurar panel para True.
use_records_limit	*sinalizador*	
records_limit	*número inteiro*	
symbol_size	*número*	Especifica um tamanho do símbolo.
panel_layout	Horizontal Vertical	

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. O nó Web pode ser usado, por exemplo, para explorar o relacionamento 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 97. Propriedades de webnode

Propriedades de **webnode**	Tipo de dados	Descrição da propriedade
use_directed_web	*sinalizador*	
campos	*lista*	
to_field	*campo*	
from_fields	*lista*	
true_flags_only	*sinalizador*	
line_values	Absoluto OverallPct PctLarger PctSmaller	
strong_links_heavier	*sinalizador*	
num_links	ShowMaximum ShowLinksAbove ShowAll	
max_num_links	*número*	
links_above	*número*	
discard_links_min	*sinalizador*	
links_min_records	*número*	
discard_links_max	*sinalizador*	
links_max_records	*número*	
weak_below	*número*	
strong_above	*número*	
link_size_continuous	*sinalizador*	
web_display	Circular Rede Directed Grid	
graph_background	*color*	As cores do gráfico padrão são descritas no início desta seção.

Tabela 97. Propriedades de webnode (continuação)

Propriedades de webnode	**Tipo de dados**	**Descrição da propriedade**
`symbol_size`	*número*	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 98. Propriedades comuns do nó de modelagem

Propriedade	Valores	Descrição da propriedade
`custom_fields`	*flag*	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`
	field
	
	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`	*flag*	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`	*flag*	
`splits`	*[field1 ... fieldN]*	Especifica o campo ou campos a serem utilizados para modelagem de divisão. Efetivo apenas se `use_split_data` estiver configurado para `True`.
`use_frequency`	*flag*	Os campos de peso e de frequência são usados por modelos específicos, conforme observado para cada tipo de modelo.
`frequency_field`	*campo*	
`use_weight`	*flag*	
`weight_field`	*campo*	
`use_model_name`	*flag*	
`model_name`	*string*	Nome customizado para o novo modelo.
`mode`	`Simple` `Expert`	

Propriedades de anomalydetectionnode

O nó Detecção de Anomalia identifica casos incomuns, ou valores discrepantes, que não estiverem em conformidade com os padrões de dados "normais". Com este nó, é possível identificar valores discrepantes, mesmo se eles não se ajustarem a nenhum dos padrões anteriormente conhecidos e mesmo se você não souber exatamente o que procura.

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 99. Propriedades de anomalydetectionnode

Propriedades de **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 161 para obter mais informações.
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.
num_records	*number*	Configura o limite para sinalizar registros com base no número de registros nos dados de treinamento.
num_fields	*integer*	O número de campos para relatar cada registro anômalo.
impute_missing_values	*flag*	
adjustment_coeff	*number*	Valor utilizado para balancear o peso relativo fornecido para campos contínuos e categóricos no cálculo da distância.
peer_group_num_auto	*flag*	Calcula automaticamente o número de grupos de peers.
min_num_peer_groups	*integer*	Especifica o número mínimo de grupos de peers utilizados quando peer_group_num_auto for configurado para True.
max_num_per_groups	*integer*	Especifica o número máximo de grupos de peers.
num_peer_groups	*integer*	Especifica o número de grupos de peers utilizados quando peer_group_num_auto for configurado para False.
noise_level	*number*	Determina como os valores discrepantes são tratados durante o armazenamento em cluster. Especifique um valor entre 0 e 0,5.

Tabela 99. Propriedades de anomalydetectionnode (continuação)

Propriedades de anomalydetectionnode	Valores	Descrição da propriedade
noise_ratio	*number*	Especifica a parte da memória alocada para o componente que deve ser utilizado 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 a partir dos dados, retirando as regras com o conteúdo mais alto de informações. O a priori oferece cinco métodos diferentes de selecionar regras e utiliza um esquema de indexação sofisticado para processar grandes conjuntos de dados de maneira eficiente. Para grandes problemas, o a priori geralmente é mais rápido para treinar; ele não possui um limite arbitrário quanto ao número de regras que podem ser retidas e pode manipular regras com até 32 pré-condições. O a priori requer que todos os campos de entrada e de saída sejam categóricos e oferece melhor desempenho porque é otimizado para este tipo de dados.

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 100. Propriedades de apriorinode

Propriedades de apriorinode	Valores	Descrição da propriedade
consequents	*field*	Os modelos a priori utilizam Subsequentes e Antecedentes ao invés dos campos de destino e de entrada padrão. Os campos de peso e de frequência não são utilizados. Consulte o tópico "Propriedades Comuns do Nó de Modelagem" na página 161 para obter mais informações.

Tabela 100. Propriedades de apriorinode (continuação)

Propriedades de apriorinode	Valores	Descrição da propriedade
antecedents	[*field1 ... fieldN*]	
min_supp	*number*	
min_conf	*number*	
max_antecedents	*number*	
true_flags	*flag*	
optimize	Speed Memory	
use_transactional_data	*flag*	
contiguous	*flag*	
id_field	*string*	
content_field	*string*	
mode	Simple Expert	
evaluation	RuleConfidence DifferenceToPrior ConfidenceRatio InformationDifference NormalizedChiSquare	
lower_bound	*número*	
optimize	Speed Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.

Propriedades de associationrulesnode

O nó Regras de Associação é semelhante ao nó a priori, no entanto, ao contrário do a priori, o nó Regras de Associação pode processar dados da lista. Além disso, o nó Regras de Associação poderá ser utilizado com o IBM SPSS Analytic Server para processar Big Data e aproveitar processamento paralelo mais rápido.

Tabela 101. Propriedades de associationrulesnode

Propriedades de 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	*integer*	O número máximo de condições que podem ser incluídas em uma única regra. Mínimo 1, máximo 9.
max_rule_predictions	*integer*	O número máximo de predições que podem ser incluídas em uma única regra. Mínimo 1, máximo 5.
max_num_rules	*integer*	O número máximo de regras que podem ser consideradas parte da construção de regra. Mínimo 1, máximo 10.000.

Tabela 101. Propriedades de associationrulesnode (continuação)

Propriedades de associationrulesnode	**Tipo de dados**	**Descrição da propriedade**
rule_criterion_top_n	Confidence Rulesupport Lift Conditionsupport Implementabilidade	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	*número*	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	*número*	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	*número*	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	*integer*	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.
exclude_rules	*Booleano*	Utilizado para selecionar uma lista de campos relacionados a partir da qual você não deseja que o modelo crie regras. Exemplo: set :gsarsnode.exclude_rules = [[[field1,field2, field3]],[[field4, field5]]] - em que cada lista de campos separados por [] é uma linha na tabela.
num_bins	*integer*	Configura o número de categorias automáticas para os quais os campos contínuos são categorizados. Mínimo 2, máximo 10.
max_list_length	*integer*	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*	

Tabela 101. Propriedades de associationrulesnode (continuação)

Propriedades de associationrulesnode	Tipo de dados	Descrição da propriedade
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	*integer*	Se upto for configurado em rules_to_display, configure o número de regras para exibir 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*	
word_cloud_sort	Confidence Rulesupport Lift Conditionsupport Implementabilidade	
word_cloud_display	*integer*	Minimum 1, maximum 20
max_predictions	*integer*	O número máximo de regras que podem ser aplicadas a cada entrada na escoragem.
criterion	Confidence Rulesupport Lift Conditionsupport Implementabilidade	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 Classificador Automático cria e compara um número de modelos diferentes de resultados binários (sim ou não, rotatividade ou não rotatividade, e assim por diante), permitindo escolher a melhor abordagem para uma análise específica. Diversos algoritmos de modelagem são suportados, possibilitando selecionar os métodos que deseja utilizar, as opções específicas para cada um deles 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 102. Propriedades de autoclassifiernode.

Propriedades de autoclassifiernode	**Valores**	**Descrição da propriedade**
target	*field*	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 161 para obter mais informações.
ranking_measure	Accuracy Area_under_curve Lucro Lift Num_variables	
ranking_dataset	Training Teste	
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	*sinalizador*	
enable_accuracy_limit	*sinalizador*	
accuracy_limit	*número inteiro*	Número inteiro entre 0 e 100.
enable_ area_under_curve _limit	*sinalizador*	
area_under_curve_limit	*número*	Número real entre 0,0 e 1,0.
enable_profit_limit	*sinalizador*	
profit_limit	*número*	Número inteiro maior que 0.
enable_lift_limit	*sinalizador*	
lift_limit	*número*	Número real maior que 1,0.
enable_number_of_variables_limit	*sinalizador*	
number_of_variables_limit	*número*	Número inteiro maior que 0.
use_fixed_cost	*sinalizador*	
fixed_cost	*número*	Número real maior que 0,0.
variable_cost	*field*	
use_fixed_revenue	*sinalizador*	
fixed_revenue	*número*	Número real maior que 0,0.
variable_revenue	*field*	
use_fixed_weight	*sinalizador*	
fixed_weight	*número*	Número real maior que 0,0.

Tabela 102. Propriedades de autoclassifiernode (continuação).

Propriedades de autoclassifiernode	Valores	Descrição da propriedade
variable_weight	*field*	
lift_percentile	*número*	Número inteiro entre 0 e 100.
enable_model_build_time_limit	*sinalizador*	
model_build_time_limit	*número*	Número inteiro configurado para o número de minutos para limitar o tempo gasto para construir cada modelo individual.
enable_stop_after_time_limit	*sinalizador*	
stop_after_time_limit	*número*	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	*sinalizador*	
use_costs	*sinalizador*	
<algorithm>	*sinalizador*	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" para obter mais informações.

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 Classificador Automático são `cart`, `chaid`, `quest`, `c50`, `logreg`, `decisionlist`, `bayesnet`, `discriminant`, `svm` e `knn`.

Os nomes de algoritmo para o nó Numeração Automática são `cart`, `chaid`, `neuralnetwork`, `genlin`, `svm`, `regression`, `linear` e `knn`.

Os nomes de algoritmo para o nó Cluster Automático 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 que possuem características semelhantes. O nó funciona da mesma maneira que outros nós de modelagem automatizados, permitindo experimentá-los com as diversas combinações de opções em uma única passagem de modelagem. Os modelos podem ser comparados utilizando medidas básicas com as quais é possível tentar filtrar e classificar a utilidade dos modelos de cluster e fornecer uma medida com base na importância de campos específicos.

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 103. Propriedades de autoclusternode

Propriedades de autoclusternode	Valores	Descrição da propriedade
evaluation	*field*	**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.
ranking_measure	Silhueta Num_clusters Size_smallest_cluster Size_largest_cluster Smallest_to_largest Importância	
ranking_dataset	Training Test	
summary_limit	*integer*	Número de modelos para lista no relatório. Especifique um número inteiro entre 1 e 100.
enable_silhouette_limit	*flag*	
silhouette_limit	*integer*	Número inteiro entre 0 e 100.
enable_number_less_limit	*flag*	
number_less_limit	*number*	Número real entre 0,0 e 1,0.
enable_number_greater_limit	*flag*	
number_greater_limit	*number*	Número inteiro maior que 0.
enable_smallest_cluster_limit	*flag*	

Tabela 103. Propriedades de autoclusternode (continuação)

Propriedades de autoclusternode	Valores	Descrição da propriedade
smallest_cluster_units	Percentage Counts	
smallest_cluster_limit_percentage	*number*	
smallest_cluster_limit_count	*integer*	Número inteiro maior que 0.
enable_largest_cluster_limit	*flag*	
largest_cluster_units	Percentage Contagens	
largest_cluster_limit_percentage	*number*	
largest_cluster_limit_count	*integer*	
enable_smallest_largest_limit	*flag*	
smallest_largest_limit	*number*	
enable_importance_limit	*flag*	
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>	*flag*	Ativa ou desativa o uso de um algoritmo específico.
<algorithm>.<property>	*string*	Configura um valor da propriedade para um algoritmo específico. Consulte o tópico "Configurando Propriedades de Algoritmo" na página 168 para obter mais informações.

Propriedades de autonumericnode

O nó Numeração Automática estima e compara modelos de resultados de intervalo numérico contínuos utilizando vários métodos diferentes. O nó funciona da mesma maneira que o nó Classificador Automático, permitindo escolher os algoritmos a serem utilizados e experimentá-los com as diversas combinações de opções em uma única passagem de modelagem. Os algoritmos suportados incluem redes neurais, Árvore C&R, CHAID, regressão linear, regressão linear generalizada e Support Vector Machines (SVMs). Os modelos podem ser comparados com base na correlação, no erro relativo ou no 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 104. Propriedades de autonumericnode

Propriedades de autonumericnode	Valores	Descrição da propriedade
custom_fields	*flag*	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 161 para obter mais informações.
inputs	*[field1 ... field2]*	
partition	*campo*	
use_frequency	*flag*	
frequency_field	*campo*	
use_weight	*flag*	
weight_field	*campo*	
use_partitioned_data	*flag*	Se um campo de partição for definido, somente os dados de treinamento são utilizados para construção de modelo.
ranking_measure	Correlação NumberOfFields	
ranking_dataset	Teste Training	
number_of_models	*integer*	Número de modelos a serem incluídos no nugget do modelo. Especifique um número inteiro entre 1 e 100.
calculate_variable_importance	*flag*	
enable_correlation_limit	*flag*	
correlation_limit	*integer*	
enable_number_of_fields_limit	*flag*	
number_of_fields_limit	*integer*	
enable_relative_error_limit	*flag*	
relative_error_limit	*integer*	
enable_model_build_time_limit	*flag*	
model_build_time_limit	*integer*	
enable_stop_after_time_limit	*flag*	
stop_after_time_limit	*integer*	
stop_if_valid_model	*flag*	
<algorithm>	*flag*	Ativa ou desativa o uso de um algoritmo específico.
<algorithm>.<property>	*string*	Configura um valor da propriedade para um algoritmo específico. Consulte o tópico "Configurando Propriedades de Algoritmo" na página 168 para obter mais informações.

Propriedades de bayesnetnode

O nó Rede Bayesian permite construir um modelo de probabilidade ao combinar uma evidência observada e registrada com conhecimento do mundo real para estabelecer a probabilidade das ocorrências. O nó foca nas redes Tree Augmented Naïve Bayes (TAN) e Markov Blanket que são utilizadas 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 105. Propriedades de bayesnetnode

Propriedades de 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 161 para obter mais informações.
continue_training_existing_model	*sinalizador*	
structure_type	TAN MarkovBlanket	Selecione a estrutura a ser utilizada ao construir a rede bayesiana.
use_feature_selection	*sinalizador*	
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	*flag*	
all_probabilities	*flag*	
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.

Tabela 105. Propriedades de bayesnetnode (continuação)

Propriedades de bayesnetnode	Valores	Descrição da propriedade
maximum_number_inputs	*number*	Especifica o número máximo de campos de entrada a serem utilizados na construção da rede bayesiana.
calculate_variable_importance	*sinalizador*	
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	
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 106. Propriedades de buildr.

Propriedades de buildr	Valores	Descrição da propriedade
build_syntax	*sequência*	Sintaxe do script R para construção de modelo.
score_syntax	*string*	Sintaxe do script R para escoragem de modelo.
convert_flags	StringsAndDoubles LogicalValues	Opção para converter os campos de sinalização.
convert_datetime	*sinalizador*	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	*sinalizador*	Opção para converter valores omissos em valor NA R.
output_html	*sinalizador*	Opção para exibir gráficos em uma guia no nugget do modelo R.
output_text	*sinalizador*	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. Diversas divisões em mais de dois subgrupos são permitidas.

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 107. Propriedades de c50node

Propriedades de 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 161 para obter mais informações.
output_type	DecisionTree RuleSet	
group_symbolics	*flag*	
use_boost	*flag*	
boost_num_trials	*number*	
use_xval	*flag*	
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	*flag*	
costs	*estruturado*	Esta é uma propriedade estruturada.
use_winnowing	*flag*	
use_global_pruning	*flag*	On (True), por padrão

Tabela 107. Propriedades de c50node (continuação)

Propriedades de c50node	Valores	Descrição da propriedade
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de carmanode

O modelo do CARMA extrai um conjunto de regras de dados sem a necessidade de especificar campos de entrada ou de destino. Ao contrário do a priori o nó CARMA oferece configurações de construção para suporte da regra (suporte para ambos antecedente e subsequente) ao invés de apenas o suporte a antecedente. Isso significa que as regras geradas podem ser utilizadas para uma variedade maior de aplicativos, por exemplo, para localizar uma lista de produtos ou serviços (antecedentes) cujo subsequente 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 108. Propriedades de carmanode

Propriedades de 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 de frequência não são utilizados. Consulte o tópico "Propriedades Comuns do Nó de Modelagem" na página 161 para obter mais informações.
id_field	*campo*	Campo utilizado como o campo de ID para construção de modelo.
contiguous	*flag*	Utilizado para especificar se os IDs no campo ID são contíguos.
use_transactional_data	*flag*	

Tabela 108. Propriedades de carmanode (continuação)

Propriedades de carmanode	Valores	Descrição da propriedade
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	*flag*	Exclui regras com diversos subsequentes. O padrão é False.
use_pruning	*flag*	O padrão é False.
pruning_value	*number*	O padrão é 500.
vary_support	*flag*	
estimated_transactions	*integer*	
rules_without_antecedents	*flag*	

Propriedades de cartnode

O nó Árvore de Classificação e Regressão (C&R) gera uma árvore de decisão que permite prever ou classificar observações futuras. O método utiliza particionamento recursivo para dividir os registros de treinamento em segmentos ao minimizar 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. Os campos de destino e de entrada podem ser intervalos numéricos ou categóricos (nominal, ordinal ou sinalizadores) e todas as divisões são binárias (somente 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 109. Propriedades de cartnode

Propriedades de **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 161 para obter mais informações.
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	O psm é utilizado para conjuntos de dados muito grandes e requer uma conexão com o Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*string*	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	*integer*	Profundidade máxima da árvore, de 0 a 1000. Usado apenas se use_max_depth = Custom.
prune_tree	*flag*	Poda a árvore para evitar super ajuste.
use_std_err	*flag*	Utiliza a diferença máxima em risco (nos Erros Padrão).
std_err_multiplier	*number*	Diferença máxima.
max_surrogates	*number*	Máximo de substitutos.
use_percentage	*flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*flag*	
costs	*estruturado*	Propriedade estruturada.
priors	Data Equal Custom	

Tabela 109. Propriedades de cartnode (continuação)

Propriedades de cartnode	Valores	Descrição da propriedade
custom_priors	*estruturado*	Propriedade estruturada.
adjust_priors	*flag*	
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	*flag*	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	*flag*	Replica a opção de resultados.
seed	*number*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de chaidnode

O nó CHAID gera árvores de decisão usando estatísticas qui-quadrado para identificar divisões ideais. Diferentemente dos nós Árvore C&R e QUEST, o 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.

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 110. Propriedades de chaidnode

Propriedades de 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 161 para obter mais informações.
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	O psm é utilizado para conjuntos de dados muito grandes e requer uma conexão com o Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*string*	
method	Chaid ExhaustiveChaid	
use_max_depth	Default Custom	
max_depth	*integer*	Profundidade máxima da árvore, de 0 a 1000. Usado apenas se use_max_depth = Custom.
use_percentage	*flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*flag*	
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.

Tabela 110. Propriedades de chaidnode (continuação)

Propriedades de chaidnode	Valores	Descrição da propriedade
large_boost	*flag*	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	*flag*	Ajusta valores de significância usando o método de Bonferroni.
split_merged_categories	*flag*	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	*integer*	
seed	*number*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	
maximum_number_of_models	*integer*	

Propriedades de coxregnode

O nó 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*) de acordo com os valores fornecidos para as 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 111. Propriedades de coxregnode

Propriedades de coxregnode	Valores	Descrição da propriedade
survival_time	*campo*	Os modelos de regressão de Cox requerem um único campo contendo os tempos de sobrevivência.

Tabela 111. Propriedades de coxregnode (continuação)

Propriedades de coxregnode	Valores	Descrição da propriedade
target	*field*	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 161 para obter mais informações.
method	Inserir Stepwise BackwardsStepwise	
groups	*campo*	
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	
removal_criterion	LR Wald Condicional	
probability_entry	*número*	
probability_removal	*número*	
output_display	EachStep LastStep	
ci_enable	*flag*	
ci_value	90 95 99	
correlation	*flag*	
display_baseline	*flag*	
survival	*flag*	
hazard	*flag*	

Tabela 111. Propriedades de coxregnode (continuação)

Propriedades de coxregnode	Valores	Descrição da propriedade
log_minus_log	*flag*	
one_minus_survival	*flag*	
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 de decisionlistnode

O nó Lista de Decisão identifica os subgrupos, ou segmentos, que mostram uma probabilidade maior ou menor de um resultado binário fornecido com relação à população geral. Por exemplo, é possível procurar por clientes que forem menos propensos a migrarem para o concorrente ou que responderão favoravelmente a uma campanha. É possível incorporar o conhecimento dos negócios no modelo ao incluir seus próprios segmentos customizados e visualizar modelos alternativos lado a lado para comparar os resultados. Os modelos de Lista de Decisão consistem em uma lista de regras em que 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 112. Propriedades de decisionlistnode

Propriedades de 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 161 para obter mais informações.
model_output_type	Modelo 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	*string*	Se não for especificado, será assumido o valor true para sinalizadores.
max_rules	*integer*	O número máximo de segmentos, exceto o restante.
min_group_size	*integer*	Tamanho mínimo do segmento.
min_group_size_pct	*number*	Tamanho mínimo do segmento como uma porcentagem.

Tabela 112. Propriedades de decisionlistnode (continuação)

Propriedades de decisionlistnode	Valores	Descrição da propriedade
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	*integer*	
mode	Simple Expert	
bin_method	EqualWidth EqualCount	
bin_count	*number*	
max_models_per_cycle	*integer*	Procura largura para listas.
max_rules_per_cycle	*integer*	Procura largura para regras de segmento.
segment_growth	*number*	
include_missing	*flag*	
final_results_only	*flag*	
reuse_fields	*flag*	Permite que atributos (campos de entrada que aparecem nas regras) sejam reutilizados.
max_alternatives	*integer*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de discriminantnode

A análise discriminante faz suposições mais rígidas do que a regressão logística, mas pode ser uma alternativa ou um complemento poderoso para 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 113. Propriedades de discriminantnode

Propriedades de 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 de frequência não são utilizados. Consulte o tópico "Propriedades Comuns do Nó de Modelagem" na página 161 para obter mais informações.

Tabela 113. Propriedades de discriminantnode (continuação)

Propriedades de discriminantnode	Valores	Descrição da propriedade
method	Inserir Stepwise	
mode	Simple Expert	
prior_probabilities	AllEqual ComputeFromSizes	
covariance_matrix	WithinGroups SeparateGroups	
means	*flag*	Opções de estatísticas na caixa de diálogo Saída Avançada.
univariate_anovas	*flag*	
box_m	*flag*	
within_group_covariance	*flag*	
within_groups_correlation	*flag*	
separate_groups_covariance	*flag*	
total_covariance	*flag*	
fishers	*flag*	
unstandardized	*flag*	
casewise_results	*flag*	Opções de classificação na caixa de diálogo Saída Avançada.
limit_to_first	*number*	O valor padrão é 10.
summary_table	*flag*	
leave_one_classification	*flag*	
combined_groups	*flag*	
separate_groups_covariance	*flag*	Opção de matrizes de **Covariância de grupos separados**.
territorial_map	*flag*	
combined_groups	*flag*	Opção de gráfico **Grupos combinados**.
separate_groups	*flag*	Opção de gráfico **Grupos separados**.
summary_of_steps	*flag*	
F_pairwise	*flag*	
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.

Tabela 113. Propriedades de discriminantnode (continuação)

Propriedades de discriminantnode	Valores	Descrição da propriedade
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de 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 executam a melhor tarefa de capturar a variação no conjunto inteiro de campos, em que os componentes são ortogonais (perpendiculares) entre si. A análise fatorial tenta identificar fatores subjacentes que explicam o padrão de correlações dentro de um conjunto de campos observados. Para ambas as abordagens, o objetivo é encontrar um número pequeno de campos derivados que efetivamente resumem 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 114. Propriedades de factornode

Propriedades de 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 de frequência não são utilizados. Consulte o tópico "Propriedades Comuns do Nó de Modelagem" na página 161 para obter mais informações.
method	PC ULS GLS ML PAF Alfa Imagem	
mode	Simple Expert	
max_iterations	*number*	
complete_records	*flag*	
matrix	Correlação Covariância	
extract_factors	ByEigenvalues ByFactors	
min_eigenvalue	*number*	
max_factor	*number*	
rotation	None Varimax DirectOblimin Equamax Quartimax Proporção máxima	
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.
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	*flag*	
hide_values	*flag*	
hide_below	*number*	

Propriedades de featureselectionnode

O nó Seleção de Recurso seleciona campos de entrada para remoção com base em um conjunto de critérios (como a porcentagem de valores omissos) e, em seguida, classifica a importância das entradas restantes com relação a um destino especificado. Por exemplo, dado um conjunto de dados com centenas de entradas em potencial, quais delas mais provavelmente serão úteis para modelar os 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 Recursos, consulte “Exemplo de Script Independente: Gerando um Modelo de Seleção de Variável” na página 4.

Tabela 115. propriedades de featureselectionnode

Propriedades de **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 de frequência não são utilizados. Consulte o tópico “Propriedades Comuns do Nó de Modelagem” na página 161 para obter mais informações.
screen_single_category	*flag*	Se True, seleciona campos que tiverem muitos registros que caem na mesma categoria com relação ao número total de registros.
max_single_category	*number*	Especifica o limite utilizado quando screen_single_category é True.
screen_missing_values	*flag*	Se True, seleciona campos com muitos valores ausentes, expresso como uma porcentagem do número total de registros.
max_missing_values	*number*	
screen_num_categories	*flag*	Se True, seleciona campos com muitas categorias com relação ao número total de registros.
max_num_categories	*number*	
screen_std_dev	*flag*	Se True, seleciona campos com um desvio padrão menor ou igual ao mínimo especificado.
min_std_dev	*number*	

Tabela 115. propriedades de featureselectionnode (continuação)

Propriedades de **featureselectionnode**	Valores	Descrição da propriedade
screen_coeff_of_var	*flag*	Se True, seleciona campos com um coeficiente de variação 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	*string*	Especifica o rótulo para a classificação não importante.
marginal_label	*string*	
important_label	*string*	
selection_mode	ImportanceLevel ImportanceValue TopN	
select_important	*flag*	Quando selection_mode for configurado para ImportanceLevel, especifica se campos importantes devem ser selecionados.
select_marginal	*flag*	Quando selection_mode for configurado para ImportanceLevel, especifica se campos marginais devem ser selecionados.
select_unimportant	*flag*	Quando selection_mode for configurado para ImportanceLevel, especifica se campos não importantes devem ser selecionados.
importance_value	*number*	Quando selection_mode for configurado para ImportanceValue, especifica o valor de corte a ser utilizado. Aceita valores de 0 a 100.
top_n	*integer*	Quando selection_mode for configurado para TopN, especifica o valor de corte a ser utilizado. Aceita valores de 0 a 1000.

Propriedades de genlinnode

O modelo Linear Generalizado expande o modelo linear geral para que a variável dependente seja linearmente relacionada aos fatores e covariáveis por meio de uma função de ligação especificada. Além disso, o modelo permite à variável dependente ter uma distribuição não normal. Ele cobre a funcionalidade de um grande número de modelos estatísticos, incluindo regressão linear, regressão logística, modelos de log-linear para dados de contagem e modelos de sobrevivência de intervalo censurado.

exemplo

```
node = stream.create("genlin", "My node")
node.setPropertyValue("model_type", "MainAndAllTwoWayEffects")
node.setPropertyValue("offset_type", "Variable")
node.setPropertyValue("offset_field", "Claimant")
```

Tabela 116. Propriedades de genlinnode

Propriedades de 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 161 para obter mais informações.
use_weight	*flag*	
weight_field	*campo*	O tipo de campo é apenas contínuo.
target_represents_trials	*flag*	
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	*flag*	
mode	Simple Expert	
distribuição	BINOMIAL GAMMA IGAUSS NEGBIN NORMAL POISSON TWEEDIE MULTINOMIAL	IGAUSS: Gaussiana inversa. NEGBIN: Binomial negativo.
negbin_para_type	Specify Estimate	
negbin_parameter	*number*	O valor padrão é 1. Deve conter um número real não negativo.
tweedie_parameter	*number*	

Tabela 116. Propriedades de genlinnode (continuação)

Propriedades de genlinnode	Valores	Descrição da propriedade
link_function	IDENTITY CLOGLOG LOG LOGC LOGIT NEGBIN NLOGLOG ODDSPOWER PROBIT POWER CUMCAUCHIT CUMCLOGLOG CUMLOGIT CUMNLOGLOG CUMPROBIT	CLOGLOG: Log-log complementar. LOGC: Complemento de log. NEGBIN: Binomial negativo. NLOGLOG: Log-log negativo. CUMCAUCHIT: Cauchit acumulativo. CUMCLOGLOG: Log-log complementar acumulativo. CUMLOGIT: Logit acumulativo. CUMNLOGLOG: Log-log negativo acumulativo. CUMPROBIT: Probito acumulativo.
power	*number*	O valor deve ser um número real diferente de zero.
method	Híbrido Fisher NewtonRaphson	
max_fisher_iterations	*number*	O valor padrão é 1; somente números inteiros positivos são permitidos.
scale_method	MaxLikelihoodEstimate Deviance PearsonChiSquare FixedValue	
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	*flag*	
start_iteration	*number*	O valor padrão é 20; somente números inteiros positivos são permitidos.
estimates_change	*flag*	
estimates_change_min	*number*	O valor padrão é 1E-006; somente números positivos são permitidos.
estimates_change_type	Absolute Relativo	
loglikelihood_change	*flag*	
loglikelihood_change_min	*number*	Apenas números positivos permitidos.
loglikelihood_change_type	Absolute Relativo	
hessian_convergence	*flag*	
hessian_convergence_min	*number*	Apenas números positivos permitidos.
hessian_convergence_type	Absolute Relative	

Tabela 116. Propriedades de genlinnode (continuação)

Propriedades de genlinnode	Valores	Descrição da propriedade
case_summary	*flag*	
contrast_matrices	*flag*	
descriptive_statistics	*flag*	
estimable_functions	*flag*	
model_info	*flag*	
iteration_history	*flag*	
goodness_of_fit	*flag*	
print_interval	*number*	O valor padrão é 1; deve ser um número inteiro positivo.
model_summary	*flag*	
lagrange_multiplier	*flag*	
parameter_estimates	*flag*	
include_exponential	*flag*	
covariance_estimates	*flag*	
correlation_estimates	*flag*	
analysis_type	TypeI TypeIII TypeIAndTypeIII	
statistics	Wald LR	
citype	Wald Perfil	
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	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validation	

Propriedades de glmmnode

Um modelo linear generalizado misto (GLMM) 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 de longitude não normais.

Tabela 117. Propriedades de glmmnode.

Propriedades glmmnode	Valores	Descrição da propriedade
residual_subject_spec	*structured*	A combinação de valores dos campos categóricos especificados que definem exclusivamente assuntos no conjunto de dados
repeated_measures	*structured*	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	*sinalizador*	Indica se deve ser utilizado um destino definido no nó de envio de dados (false) ou um destino customizado especificado por target_field (true).
target_field	*field*	Campo a ser utilizado como destino se custom_target for true.
use_trials	*sinalizador*	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	*field*	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	*sinalizador*	Indica se a categoria de referência customizada deve ser utilizada para uma variável resposta categórica. O padrão é false.

Tabela 117. Propriedades de glmmnode (continuação).

Propriedades glmmnode	Valores	Descrição da propriedade
target_reference_value	*sequência*	Categoria de referência a ser utilizada 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 a partir da lista fornecida pelo target_distribution.
target_distribution	Normal Binomial Multinomial Gama Inverse NegativeBinomial Poisson	Distribuição de valores para o destino quando dist_link_combination for Custom.
link_function_type	Identidade LogC Log CLOGLOG Logit NLOGLOG PROBIT POWER CAUCHIT	Função Link para relacionar valores de destino para pred Se target_distribution for Binomial será possível usar qualquer uma das funções de ligação listadas. Se target_distribution for Multinomial, será possível utilizar CLOGLOG, CAUCHIT, LOGIT, NLOGLOG ou PROBIT. Se target_distribution for algo diferente de Binomial ou Multinomial, será possível utilizar IDENTITY, LOG ou POWER.
link_function_param	*número*	Valor do parâmetro da função de ligação a ser utilizado. Aplicável apenas se normal_link_function ou link_function_type for POWER.
use_predefined_inputs	*sinalizador*	Indica se os campos de efeito fixo devem ser aqueles campos de envio de dados definidos como campos de entrada (true) ou aqueles a partir de fixed_effects_list (false). O padrão é false.
fixed_effects_list	*structured*	Se use_predefined_inputs for false, especifica os campos de entrada a serem usados como campos de efeito fixo.
use_intercept	*sinalizador*	Se true (padrão), inclui a interceptação no modelo.
random_effects_list	*structured*	Lista de campos para especificar como efeitos aleatórios.
regression_weight_field	*field*	Campo a ser utilizado como campo de ponderação da análise.
use_offset	Nenhum offset_value offset_field	Indica como o deslocamento for especificado. O valor None significa que nenhum deslocamento é utilizado.

Tabela 117. Propriedades de glmmnode (continuação).

Propriedades glmmnode	Valores	Descrição da propriedade
offset_value	*número*	Valor a ser utilizado para deslocamento se use_offset for configurado para offset_value.
offset_field	*field*	Campo a ser utilizado para o valor de deslocamento se use_offset for configurado para offset_field.
target_category_order	Ascending Descending Data	Ordenação de classificação para variáveis resposta categórica. O valor Data especifica usar a ordem de classificação localizada nos dados. O padrão é Ascending.
inputs_category_order	Ascendente Descending Data	Ordenação de classificação para preditores categóricos. O valor Data especifica usar a ordem de classificação localizada nos dados. O padrão é Ascending.
max_iterations	*integer*	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_coeffecients	Model Robust	Método para calcular a matriz de covariância de estimativa de parâmetro.
use_p_converge	*flag*	Opção para a convergência de parâmetro.
p_converge	*number*	Em branco, ou qualquer valor positivo.
p_converge_type	Absoluto Relativo	
use_l_converge	*sinalizador*	Opção para convergência de log da verossimilhança.
l_converge	*número*	Em branco, ou qualquer valor positivo.
l_converge_type	Absoluto Relativo	
use_h_converge	*sinalizador*	Opção para convergência da Hessiana.
h_converge	*número*	Em branco, ou qualquer valor positivo.
h_converge_type	Absoluto Relativo	
max_fisher_steps	*integer*	
singularity_tolerance	*número*	
use_model_name	*sinalizador*	Indica se deve especificar um nome customizado para o modelo (true) ou utilizar 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 utilizado.

Tabela 117. Propriedades de glmmnode (continuação).

Propriedades glmmnode	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.
score_category_probabilities	*sinalizador*	Se true, produz probabilidades previstas para variáveis resposta categórica. O padrão é false.
max_categories	*número inteiro*	Se score_category_probabilities for true, especifica o número máximo de categorias a serem salvas.
score_propensity	*sinalizador*	Se true, produz escores de propensão para campos de destino de sinalização que indicam a probabilidade do resultado "true" para o campo.
emeans	*structure*	Para cada campo categórico na lista de efeitos fixos, especifica se deve produzir médias marginais estimadas.
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 de kmeansnode

O nó K-Médias armazena em cluster o conjunto de dados em grupos distintos (ou clusters). O método define um número fixo de clusters, designa iterativamente registros para clusters e ajusta os centros do cluster até que o refinamento adicional não consiga mais melhorar o modelo. Ao invés de tentar prever um resultado, o *k-médias* utiliza 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 118. Propriedades de kmeansnode

Propriedades de 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 de frequência não são utilizados. Consulte o tópico "Propriedades Comuns do Nó de Modelagem" na página 161 para obter mais informações.
num_clusters	*number*	
gen_distance	*flag*	
cluster_label	String Number	
label_prefix	*string*	
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 de knnnode

O nó *k*-Nearest Neighbor (KNN) associa um novo caso à categoria ou valor dos *k* objetos mais próximos a ele no espaço do preditor, em que *k* é um número inteiro. Os casos semelhantes estão próximos uns dos outros e os casos diferentes estão 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 119. Propriedades de knnnode

Propriedades de knnnode	Valores	Descrição da propriedade
análise	PredictTarget IdentifyNeighbors	
objective	Equilíbrio Velocidade Accuracy Custom	
normalize_ranges	*flag*	
use_case_labels	*flag*	Caixa de seleção para ativar a próxima opção.
case_labels_field	*campo*	
identify_focal_cases	*flag*	Caixa de seleção para ativar a próxima opção.
focal_cases_field	*campo*	
automatic_k_selection	*flag*	
fixed_k	*integer*	Ativado somente se automatic_k_selectio for False.
minimum_k	*integer*	Ativado somente se automatic_k_selectio for True.
maximum_k	*integer*	
distance_computation	Euclidiano CityBlock	
weight_by_importance	*flag*	
range_predictions	Média Mediana	
perform_feature_selection	*flag*	
forced_entry_inputs	[*field1 ... fieldN*]	
stop_on_error_ratio	*flag*	
number_to_select	*integer*	
minimum_change	*number*	
validation_fold_assign_by_field	*flag*	
number_of_folds	*integer*	Ativado somente se validation_fold_assign_by_field for False.
set_random_seed	*flag*	
random_seed	*number*	
folds_field	*campo*	Ativado somente se validation_fold_assign_by_field for True.
all_probabilities	*flag*	
save_distances	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de kohonennode

O nó Kohonen gera um tipo de rede neural que pode ser utilizado para armazenar em cluster o conjunto de dados em grupos distintos. Quando a rede for totalmente treinada, os registros que forem semelhantes deverão estar bem próximos no mapa de saída, ao passo que registros que forem diferentes estarão afastados. É possível examinar o número de observações capturadas por cada unidade no nugget do modelo para identificar as unidades fortes. Isto poderá dar uma ideia 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 120. Propriedades de kohonennode

Propriedades de 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 161 para obter mais informações.
continue	*flag*	
show_feedback	*flag*	
stop_on	Default Time	
time	*number*	
optimize	Velocidade Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.
cluster_label	*flag*	
mode	Simple Expert	
width	*number*	
length	*number*	

Tabela 120. Propriedades de kohonennode (continuação)

Propriedades de kohonennode	Valores	Descrição da propriedade
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 lineares preveem uma variável resposta contínua com base em relacionamentos lineares entre o destino 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 121. Propriedades de linearnode.

Propriedades de linearnode	Valores	Descrição da propriedade
target	*field*	Especifica um campo de destino único.
inputs	[*field1 ... fieldN*]	Campos do preditor utilizados pelo modelo.
continue_training_existing_model	*flag*	
objective	Standard Bagging Boosting psm	O psm é utilizado para conjuntos de dados muito grandes e requer uma conexão com o Servidor.
use_auto_data_preparation	*flag*	
confidence_level	*número*	
model_selection	ForwardStepwise BestSubsets None	
criteria_forward_stepwise	AICC Fstatistics AdjustedRSquare ASE	

Tabela 121. Propriedades de linearnode (continuação).

Propriedades de linearnode	Valores	Descrição da propriedade
probability_entry	*número*	
probability_removal	*número*	
use_max_effects	*flag*	
max_effects	*número*	
use_max_steps	*flag*	
max_steps	*número*	
criteria_best_subsets	AICC AdjustedRSquare ASE	
combining_rule_continuous	Média Median	
component_models_n	*número*	
use_random_seed	*flag*	
random_seed	*número*	
use_custom_model_name	*flag*	
custom_model_name	*string*	
use_custom_name	*flag*	
custom_name	*string*	
tooltip	*string*	
keywords	*string*	
annotation	*string*	

Propriedades de linearasnode

Os modelos de regressão lineares preveem uma variável resposta contínua com base em relacionamentos lineares entre o destino e um ou mais preditores.

Tabela 122. Propriedades de linearasnode

Propriedades de 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	*sinalizador*	O valor padrão é TRUE.
intercept	*flag*	O valor padrão é TRUE.
detect_2way_interaction	*flag*	Especifica se uma interação bilateral deve ser considerada ou não. O valor padrão é TRUE.

Tabela 122. Propriedades de linearasnode (continuação)

Propriedades de linearasnode	Valores	Descrição da propriedade
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 nenhum	O método de seleção do modelo a ser utilizado. O valor padrão é ForwardStepwise.
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	*número*	O efeito que possuir o menor valor-p inferior a esse limite de pin especificado será incluído no modelo. O valor padrão é 0.05.
pout	*número*	Quaisquer efeitos no modelo com um valor-p maior que esse limite de pout especificado são removidos. O valor padrão é 0.10.
use_custom_max_effects	*sinalizador*	Especifica se o número máximo de efeitos deve ser usado no modelo final. O valor padrão é FALSE.
max_effects	*número*	O número máximo de efeitos a ser usado no modelo final. O valor padrão é 1.
use_custom_max_steps	*sinalizador*	Especifica se o número máximo de etapas deve ser usado. O valor padrão é FALSE.
max_steps	*número*	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

A regressão logística é uma técnica estatística para classificar registros com base em valores de campo de entrada. Ela é semelhante a uma regressão linear, porém usa um campo de variável resposta categórica ao invés 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..." seção
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..." seção
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..." seção
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 123. Propriedades de logregnode.

Propriedades de logregnode	Valores	Descrição da propriedade
target	*field*	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 161 para obter mais informações.
logistic_procedure	Binomial Multinomial	
include_constant	*flag*	
mode	Simple Expert	
method	Inserir Stepwise Avançar Voltar BackwardsStepwise	
binomial_method	Inserir Avançar Backwards	
model_type	MainEffects FullFactorial Custom	Quando FullFactorial é especificado como o tipo de modelo, os métodos de progresso não serão executados, mesmo se especificados. Ao invés disso, Enter será o método utilizado. Se o tipo de modelo for configurado para Custom, mas nenhum campo customizado estiver especificado, um modelo de efeitos principal será construído.
custom_terms	[*[BP Sex][BP][Age]*]	
multinomial_base_category	*sequência*	Especifica como a categoria de referência é determinada.
binomial_categorical_input	*sequência*	

Tabela 123. Propriedades de logregnode (continuação).

Propriedades de logregnode	Valores	Descrição da propriedade
binomial_input_contrast	Indicador Simple Difference Helmert Repetido Polinomial 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	*número*	
all_probabilities	*flag*	
tolerance	1.0E-5 1.0E-6 1.0E-7 1.0E-8 1.0E-9 1.0E-10	
min_terms	*número*	
use_max_terms	*flag*	
max_terms	*número*	
entry_criterion	Score LR	
removal_criterion	LR Wald	
probability_entry	*número*	
probability_removal	*número*	
binomial_probability_entry	*número*	
binomial_probability_removal	*número*	
requirements	HierarchyDiscrete HierarchyAll Containment None	
max_iterations	*número*	
max_steps	*número*	
p_converge	1.0E-4 1.0E-5 1.0E-6 1.0E-7 1.0E-8 0	

Tabela 123. Propriedades de logregnode (continuação).

Propriedades de logregnode	Valores	Descrição da propriedade
l_converge	1.0E-1 1.0E-2 1.0E-3 1.0E-4 1.0E-5 0	
delta	*número*	
iteration_history	*flag*	
history_steps	*número*	
summary	*flag*	
likelihood_ratio	*flag*	
asymptotic_correlation	*flag*	
goodness_fit	*flag*	
parameters	*flag*	
confidence_interval	*número*	
asymptotic_covariance	*flag*	
classification_table	*flag*	
stepwise_summary	*flag*	
info_criteria	*flag*	
monotonicity_measures	*flag*	
binomial_output_display	at_each_step at_last_step	
binomial_goodness_of_fit	*flag*	
binomial_parameters	*flag*	
binomial_iteration_history	*flag*	
binomial_classification_plots	*flag*	
binomial_ci_enable	*flag*	
binomial_ci	*número*	
binomial_residual	outliers all	
binomial_residual_enable	*flag*	
binomial_outlier_threshold	*número*	
binomial_classification_cutoff	*número*	
binomial_removal_criterion	LR Wald Conditional	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	

Propriedades de neuralnetnode

Cuidado: Uma versão mais recente do nó de modelagem Rede Neural, 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 124. Propriedades de neuralnetnode

Propriedades de 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 161 para obter mais informações.
method	Rápido Dinâmico Multiple Podar ExhaustivePrune RBFN	
prevent_overtrain	*flag*	
train_pct	*number*	
set_random_seed	*flag*	
random_seed	*number*	
mode	Simple Expert	
stop_on	Default Accuracy Ciclos Time	Modo de parada.
accuracy	*number*	Precisão da parada.

Tabela 124. Propriedades de neuralnetnode (continuação)

Propriedades de neuralnetnode	Valores	Descrição da propriedade
cycles	*number*	Ciclos para treinamento.
time	*number*	Tempo de treinamento (minutos).
continue	*flag*	
show_feedback	*flag*	
binary_encode	*flag*	
use_last_model	*flag*	
gen_logfile	*flag*	
logfile_name	*string*	
alpha	*number*	
initial_eta	*number*	
high_eta	*number*	
low_eta	*number*	
eta_decay_cycles	*number*	
hid_layers	Um Dois Árvore	
hl_units_one	*number*	
hl_units_two	*number*	
hl_units_three	*number*	
persistence	*number*	
m_topologies	*string*	
m_non_pyramids	*flag*	
m_persistence	*number*	
p_hid_layers	Um Dois Árvore	
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*	
p_overall_pers	*number*	
r_persistence	*number*	
r_num_clusters	*number*	
r_eta_auto	*flag*	
r_alpha	*number*	
r_eta	*number*	

Tabela 124. Propriedades de neuralnetnode (continuação)

Propriedades de neuralnetnode	Valores	Descrição da propriedade
optimize	Speed Memory	Use para especificar se a construção de modelo deve ser otimizada para velocidade ou para memória.
calculate_variable_importance	*flag*	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	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de neuralnetworknode

O nó Rede Neural utiliza um modelo simplificado da maneira com que o cérebro humano processa informações. Ele funciona ao simular um grande número de unidades de processamento interconectadas que lembram versões de neurônios abstratas. As redes neurais são estimadores de função geral poderosos que requerem conhecimento mínimo em estatística ou matemática para treinamento ou aplicação.

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 125. Propriedades de neuralnetworknode

Propriedades de neuralnetworknode	Valores	Descrição da propriedade
destino	[*field1 ... fieldN*]	Especifica campos de destino.
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	*flag*	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	*flag*	Continua treinando o modelo existente.
objective	Standard Bagging Boosting psm	O psm é utilizado para conjuntos de dados muito grandes e requer uma conexão com o Servidor.
method	MultilayerPerceptron RadialBasisFunction	
use_custom_layers	*flag*	

Tabela 125. Propriedades de neuralnetworknode (continuação)

Propriedades de neuralnetworknode	Valores	Descrição da propriedade
first_layer_units	*number*	
second_layer_units	*number*	
use_max_time	*flag*	
max_time	*number*	
use_max_cycles	*flag*	
max_cycles	*number*	
use_min_accuracy	*flag*	
min_accuracy	*number*	
combining_rule_categorical	Voting HighestProbability HighestMeanProbability	
combining_rule_continuous	Mean Median	
component_models_n	*number*	
overfit_prevention_pct	*number*	
use_random_seed	*flag*	
random_seed	*number*	
missing_values	listwiseDeletion missingValueImputation	
use_model_name	*boolean*	
model_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*number*	
score_propensity	*flag*	
use_custom_name	*flag*	
custom_name	*string*	
tooltip	*string*	
keywords	*string*	
annotation	*string*	

Propriedades de questnode

O nó QUEST fornece um método de classificação binária para construir árvores de decisão, projetado para reduzir o tempo de processamento necessário para grandes análises de Árvore C&R enquanto também reduz a tendência localizada nos métodos de árvore de classificação para favorecer entradas que permitem mais divisões. Os campos de entrada podem ser intervalos numéricos (contínuo), ao passo que o campo de destino deve ser categórico. Todas as divisões 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 126. Propriedades de questnode

Propriedades de 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 161 para obter mais informações.
continue_training_existing_model	*flag*	
objective	Standard Boosting Bagging psm	O psm é utilizado para conjuntos de dados muito grandes e requer uma conexão com o Servidor.
model_output_type	Single InteractiveBuilder	
use_tree_directives	*flag*	
tree_directives	*string*	
use_max_depth	Default Custom	
max_depth	*integer*	Profundidade máxima da árvore, de 0 a 1000. Usado apenas se use_max_depth = Custom.
prune_tree	*flag*	Poda a árvore para evitar super ajuste.
use_std_err	*flag*	Utiliza a diferença máxima em risco (nos Erros Padrão).
std_err_multiplier	*number*	Diferença máxima.
max_surrogates	*number*	Máximo de substitutos.
use_percentage	*flag*	
min_parent_records_pc	*number*	
min_child_records_pc	*number*	
min_parent_records_abs	*number*	
min_child_records_abs	*number*	
use_costs	*flag*	
costs	*estruturado*	Propriedade estruturada.

Tabela 126. Propriedades de questnode (continuação)

Propriedades de questnode	Valores	Descrição da propriedade
priors	Data Equal Custom	
custom_priors	*estruturado*	Propriedade estruturada.
adjust_priors	*flag*	
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	Média Mediana	Regra de combinação padrão para variáveis resposta contínua.
large_boost	*flag*	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	*flag*	Replica a opção de resultados.
seed	*number*	
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	
adjusted_propensity_partition	Teste Validação	

Propriedades de regressionnode

A regressão linear é uma técnica de estatística comum para resumir dados e fazer predições ao ajustar uma linha ou superfície reta que minimiza as discrepâncias entre os 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..." seção
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..." seção
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 127. Propriedades de regressionnode

Propriedades de 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 161 para obter mais informações.
method	Inserir Stepwise Voltar Avançar	
include_constant	*flag*	
use_weight	*flag*	
weight_field	*campo*	
mode	Simple Expert	
complete_records	*flag*	
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 : usa a probabilidade de F useF: utiliza o valor F
probability_entry	*number*	

Tabela 127. Propriedades de regressionnode (continuação)

Propriedades de regressionnode	Valores	Descrição da propriedade
probability_removal	*number*	
F_value_entry	*number*	
F_value_removal	*number*	
selection_criteria	*flag*	
confidence_interval	*flag*	
covariance_matrix	*flag*	
collinearity_diagnostics	*flag*	
regression_coefficients	*flag*	
exclude_fields	*flag*	
durbin_watson	*flag*	
model_fit	*flag*	
r_squared_change	*flag*	
p_correlations	*flag*	
descriptives	*flag*	
calculate_variable_importance	*flag*	

Propriedades de sequencenode

O nó Sequência descobre as regras de associação nos dados sequenciais ou orientados a tempo. Uma sequência é uma lista de conjuntos de itens que tendem a ocorrer em uma ordem previsível. Por exemplo, um cliente que compra uma lâmina de barbear e uma loção pós-barba poderá comprar um creme de barbear na próxima compra. O nó Sequência é baseado no algoritmo de regras de associação do CARMA que utiliza um método de duas passagens eficiente 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 128. Propriedades de sequencenode

Propriedades de 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 de frequência não são utilizados. Consulte o tópico "Propriedades Comuns do Nó de Modelagem" na página 161 para obter mais informações.
time_field	*campo*	
use_time_field	*flag*	
content_fields	[*field1 ... fieldn*]	
contiguous	*flag*	
min_supp	*number*	
min_conf	*number*	
max_size	*number*	
max_predictions	*number*	
mode	Simple Expert	
use_max_duration	*flag*	
max_duration	*number*	
use_gaps	*flag*	
min_item_gap	*number*	
max_item_gap	*number*	
use_pruning	*flag*	
pruning_value	*number*	
set_mem_sequences	*flag*	
mem_sequences	*integer*	

Propriedades de slrmnode

O nó Self-Learning Response Model (SLRM) permite construir um modelo no qual um novo caso único, ou um pequeno número de casos novos, podem ser usados para estimar novamente o modelo sem precisar retreinar esse modelo 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 129. Propriedades de slrmnode

Propriedades de 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 161 para obter mais informações.
target_response	*campo*	O Tipo deve ser sinalizador.
continue_training_existing_model	*flag*	
target_field_values	*flag*	Utiliza tudo: Usa todos os valores da origem. Especifique: Selecione valores necessários.
target_field_values_specify	*[field1 ... fieldN]*	
include_model_assessment	*flag*	
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	*flag*	
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	*flag*	
calculate_variable_importance	*flag*	

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. Esse nó requer uma cópia licenciada do IBM SPSS Statistics.

As propriedades desse nó são descritas em "Propriedades de statisticsmodelnode" na página 300.

Propriedades de stpnode

O nó Spatio-Temporal Prediction (STP) utiliza dados que contêm dados do local, campos de entrada para predição (preditores), um campo de hora e um campo de destino. Cada local possui várias linhas nos dados que representam os valores de cada preditor em cada momento da medição. Após os dados serem analisados, eles podem ser usados para prever valores de destino em qualquer local dentro dos dados de formato que são usados na análise.

Tabela 130. Propriedades de stpnode

Propriedades de 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 utilizado na saída para rotular os locais escolhidos em local
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	
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 depende do tipo de armazenamento do campo que é escolhido como o time_field para o modelo.
period_start	*integer*	
start_month	January February March April May June July August September October November December	O mês em que o modelo iniciará a indexação (por exemplo, se configurado para March, mas o primeiro registro no conjunto de dados for January, o modelo ignorará os dois primeiros registros e iniciará a indexação em março.

Tabela 130. Propriedades de stpnode (continuação)

Propriedades de stpnode	Tipo de dados	Descrição da propriedade
week_begins_on	Domingo 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	*integer*	O mínimo é 1 e o máximo é 7, em incrementos de 1.
hours_per_day	*integer*	O número de horas que o modelo conta para um dia. Se isto for configurado para 10, o modelo começará a indexar no horário day_begins_at e continuará a indexação por 10 horas, em seguida, irá para o próximo valor que corresponder ao valor de day_begins_at, e assim por diante.
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.
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 um tipo de intervalo seconds, o modelo criará um índice dos dados a cada 30 segundos.
data_matches_interval	*Boolean*	Se configurado como N, a conversão dos dados para o interval_type regular ocorre antes de o modelo ser construído. Se seus dados já estiverem no formato correto, e se o interval_type e quaisquer configurações associadas corresponderem aos seus dados, configure isto como Y para evitar a conversão ou agregação de seus dados. Configurar isso como Y desativa todos os controles de Agregação.

Tabela 130. Propriedades de stpnode (continuação)

Propriedades de **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],[]..] Demo: [['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 utilizada com esse campo.
Guia **Configurações Básicas**		
include_intercept	*flag*	
max_autoregressive_lag	*integer*	O mínimo é 1 e o máximo é 5, em incrementos de 1. Este é o número de registros anteriores necessários para uma predição. Portanto, se configurado para 5, por exemplo, os 5 registros anteriores serão utilizados 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.
estimation_method	Parametric Nonparametric	O método para modelar a matriz de covariâncias espacial
parametric_model	Gaussian Exponential PoweredExponential	Parâmetro de ordem para o modelo de covariância espacial Parametric
exponential_power	*number*	Nível de energia para o modelo PoweredExponential. Mínimo 1, máximo 2.
Guia **Avançado**		
max_missing_values	*integer*	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.

Tabela 130. Propriedades de stpnode (continuação)

Propriedades de stpnode	Tipo de dados	Descrição da propriedade
Guia **Saída**		
model_specifications	*flag*	
temporal_summary	*flag*	
location_summary	*flag*	Determina se a tabela Resumo do Local é incluída na saída do modelo.
model_quality	*flag*	
test_mean_structure	*flag*	
mean_structure_coefficients	*flag*	
autoregressive_coefficients	*flag*	
test_decay_space	*flag*	
parametric_spatial_covariance	*flag*	
correlations_heat_map	*flag*	
correlations_map	*flag*	
location_clusters	*flag*	
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	*integer*	O limite superior para o número de clusters que podem ser incluídos na saída de modelo.
Guia **Opções de Modelo**		
use_model_name	*flag*	
model_name	*string*	
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 classificar dados em um dos dois grupos sem super ajuste. O SVM funciona 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 131. Propriedades de svmnode.

Propriedades de svmnode	Valores	Descrição da propriedade
all_probabilities	*flag*	
stopping_criteria	1.0E-1 1.0E-2 1.0E-3 (default) 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	*número*	Utilizado apenas se o nível de medição do campo de destino for Continuos.
kernel	RBF(padrão) Polynomial Sigmoid Linear	Tipo de função de kernel usado para a transformação.
rbf_gamma	*número*	Utilizado apenas se kernel for RBF.
gamma	*número*	Utilizado apenas se Kernel fora Polynomial ou Sigmoide.
bias	*número*	
degree	*número*	Utilizado apenas se kernel for Polynomial.
calculate_variable_importance	*flag*	
calculate_raw_propensities	*flag*	
calculate_adjusted_ propensities	*flag*	
adjusted_propensity_partition	Teste 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 132. Propriedades de tcmnode

Propriedades de 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*]	

Tabela 132. Propriedades de tcmnode (continuação)

Propriedades de tcmnode	Valores	Descrição da propriedade
candidate_inputs	[*f1 ... fN*]	
forced_inputs	[*f1 ... fN*]	
use_timestamp	Timestamp Período	
input_interval	None Unknown Ano Trimestre Mês Semana Day Hora Hour_nonperiod Minuto Minute_nonperiod Segundo Second_nonperiod	
period_field	*string*	
period_start_value	*integer*	
num_days_per_week	*integer*	
start_day_of_week	Domingo Segunda-feira Terça-feira Quarta-feira Quinta-feira Sexta-feira Sábado	
num_hours_per_day	*integer*	
start_hour_of_day	*integer*	
timestamp_increments	*integer*	
cyclic_increments	*integer*	
cyclic_periods	*lista*	
output_interval	None Ano Trimestre Mês Semana Day Hora Minuto Segundo	
is_same_interval	Same Notsame	
cross_hour	*Booleano*	
aggregate_and_distribute	*lista*	
aggregate_default	Média Sum Mode Min Max	

Tabela 132. Propriedades de tcmnode (continuação)

Propriedades de tcmnode	Valores	Descrição da propriedade
distribute_default	Mean Sum	
group_default	Mean Sum Mode Min Max	
missing_imput	Linear_interp Series_mean K_mean K_meridian Linear_trend Nenhum	
k_mean_param	*integer*	
k_median_param	*integer*	
missing_value_threshold	*integer*	
conf_level	*integer*	
max_num_predictor	*integer*	
max_lag	*integer*	
epsilon	*número*	
threshold	*integer*	
is_re_est	*Booleano*	
num_targets	*integer*	
percent_targets	*integer*	
fields_display	*lista*	
series_display	*lista*	
network_graph_for_target	*Booleano*	
sign_level_for_target	*número*	
fit_and_outlier_for_target	*Booleano*	
sum_and_para_for_target	*Booleano*	
impact_diag_for_target	*Booleano*	
impact_diag_type_for_target	Effect Cause Ambos	
impact_diag_level_for_target	*integer*	
series_plot_for_target	*Booleano*	
res_plot_for_target	*Boolean*	
top_input_for_target	*Boolean*	
forecast_table_for_target	*Booleano*	
same_as_for_target	*Booleano*	
network_graph_for_series	*Booleano*	
sign_level_for_series	*número*	
fit_and_outlier_for_series	*Booleano*	

Tabela 132. Propriedades de tcmnode (continuação)

Propriedades de tcmnode	Valores	Descrição da propriedade
sum_and_para_for_series	*Booleano*	
impact_diagram_for_series	*Booleano*	
impact_diagram_type_for_series	Effect Cause Ambos	
impact_diagram_level_for_series	*integer*	
series_plot_for_series	*Booleano*	
residual_plot_for_series	*Booleano*	
forecast_table_for_series	*Booleano*	
outlier_root_cause_analysis	*Booleano*	
causal_levels	*integer*	
outlier_table	Interactive Pivot Ambos	
rmsp_error	*Booleano*	
bic	*Booleano*	
r_square	*Booleano*	
outliers_over_time	*Booleano*	
series_transormation	*Booleano*	
use_estimation_period	*Booleano*	
estimation_period	Times Observation	
observations	*lista*	
observations_type	Mais recente Mais Antigo	
observations_num	*integer*	
observations_exclude	*integer*	
extend_records_into_future	*Booleano*	
forecastperiods	*integer*	
max_num_distinct_values	*integer*	
display_targets	FIXEDNUMBER PERCENTAGE	
goodness_fit_measure	ROOTMEAN BIC RSQUARE	
top_input_for_series	*Booleano*	
aic	*Booleano*	
rmse	*Booleano*	

Propriedades de timeseriesnode

O nó Séries Temporais estima modelos Média Móvel Integrada AutoRegressiva (ARIMA) univariados de suavização exponencial e modelos ARIMA multivariados (ou de função de transferência) para os dados de séries temporais e produz previsões de desempenho futuro. Um nó Séries Temporais deve sempre ser 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 133. Propriedades de timeseriesnode

Propriedades de **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 161 para obter mais informações.
continue	*flag*	
method	ExpertModeler Exsmooth Arima Reuse	
expert_modeler_method	*flag*	
consider_seasonal	*flag*	
detect_outliers	*flag*	
expert_outlier_additive	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_innovational	*flag*	
expert_outlier_level_shift	*flag*	
expert_outlier_transient	*flag*	
expert_outlier_seasonal_additive	*flag*	
expert_outlier_local_trend	*flag*	
expert_outlier_additive_patch	*flag*	
exsmooth_model_type	Simple HoltsLinearTrend BrownsLinearTrend DampedTrend SimpleSeasonal WintersAdditive WintersMultiplicative	

Tabela 133. Propriedades de timeseriesnode (continuação)

Propriedades de **timeseriesnode**	Valores	Descrição da propriedade
exsmooth_transformation_type	None SquareRoot NaturalLog	
arima_p	*integer*	
arima_d	*integer*	
arima_q	*integer*	
arima_sp	*integer*	
arima_sd	*integer*	
arima_sq	*integer*	
arima_transformation_type	None SquareRoot NaturalLog	
arima_include_constant	*flag*	
tf_arima_p. *fieldname*	*integer*	Para funções de transferência.
tf_arima_d. *fieldname*	*integer*	Para funções de transferência.
tf_arima_q. *fieldname*	*integer*	Para funções de transferência.
tf_arima_sp. *fieldname*	*integer*	Para funções de transferência.
tf_arima_sd. *fieldname*	*integer*	Para funções de transferência.
tf_arima_sq. *fieldname*	*integer*	Para funções de transferência.
tf_arima_delay. *fieldname*	*integer*	Para funções de transferência.
tf_arima_transformation_type. *fieldname*	None SquareRoot NaturalLog	Para funções de transferência.
arima_detect_outlier_mode	None Automático	
arima_outlier_additive	*flag*	
arima_outlier_level_shift	*flag*	
arima_outlier_innovational	*flag*	
arima_outlier_transient	*flag*	
arima_outlier_seasonal_additive	*flag*	
arima_outlier_local_trend	*flag*	
arima_outlier_additive_patch	*flag*	
conf_limit_pct	*real*	
max_lags	*integer*	
events	*campos*	

Tabela 133. Propriedades de timeseriesnode (continuação)

Propriedades de timeseriesnode	Valores	Descrição da propriedade
scoring_model_only	*flag*	Utilize para modelos com números muito grandes (dezenas de milhares) de séries temporais.

Propriedades de treeasnode

O nó Árvore-AS estará disponível apenas se você tiver uma conexão com o IBM SPSS Analytic Server. Este nó é semelhante ao nó CHAID existente, no entanto, o nó Árvore-AS é projetado para processar Big Data para criar uma árvore única e exibe o modelo resultante no visualizador de saída que foi incluído no 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 134. Propriedades de treeasnode

Propriedades de treeasnode	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 161 para obter mais informações.
method	chaid exhaustive_chaid	
max_depth	*integer*	Profundidade máxima da árvore, de 0 a 20. O valor padrão é 5.
num_bins	*integer*	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	*integer*	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	*número*	Nível de significância para divisão. O valor deve estar entre 0,05 e 0,95.
merge_alpha	*número*	Nível de significância para mesclagem. O valor deve estar entre 0,05 e 0,95.
bonferroni_adjustment	*sinalizador*	Ajusta valores de significância usando o método de Bonferroni.

Tabela 134. Propriedades de treeasnode (continuação)

Propriedades de treeasnode	Valores	Descrição da propriedade
effect_size_threshold_cont	*número*	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	*número*	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	*sinalizador*	Permite redivisão de categorias mescladas.
grouping_sig_level	*número*	Utilizado para determinar como os grupos de nós são formados ou como nós incomuns são identificados.
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	*número*	O valor padrão é 2, o mínimo é 1 e o máximo é 100, em incrementos de 1. O valor de ramificação pai deve ser superior à ramificação filha.
min_child_records_pc	*número*	O valor padrão é 1, o mínimo é 1 e o máximo é 100, em incrementos de 1.
min_parent_records_abs	*número*	O valor padrão é 100. O mínimo é 1 e o máximo é 100, em incrementos de 1. O valor de ramificação pai deve ser superior à ramificação filha.
min_child_records_abs	*número*	O valor padrão é 50. O mínimo é 1 e o máximo é 100, em incrementos de 1.
epsilon	*número*	Mudança mínima nas frequências de célula esperadas.
max_iterations	*número*	Iterações máximas para convergência.
use_costs	*sinalizador*	
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	nenhum linear square custom	**Nota:** Ativado somente para destinos ordinais. Configure os valores padrão na matriz de custos.
calculate_conf	*sinalizador*	
display_rule_id	*sinalizador*	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 utiliza um método de clusterização em duas etapas. A primeira etapa faz uma única passagem através dos dados para compactar os dados de entrada brutos em um conjunto gerenciável de subclusters. A segunda etapa usa um método de armazenamento em cluster hierárquico para mesclar progressivamente os subclusters em clusters cada vez maiores. O TwoStep tem a vantagem de estimar automaticamente o número ideal de clusters para os dados de treinamento. Ele pode manipular tipos de campo combinados e grandes conjuntos de dados de maneira 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 135. Propriedades de twostepnode

Propriedades de **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 161 para obter mais informações.
standardize	*flag*	
exclude_outliers	*flag*	
percentage	*number*	
cluster_num_auto	*flag*	
min_num_clusters	*number*	
max_num_clusters	*number*	
num_clusters	*number*	
cluster_label	String Number	
label_prefix	*string*	
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) dentro de um conjunto de dados que, de outra forma, não seriam 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 136. Propriedades de twostepAS

Propriedades de **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	integer	Default=2
max_num_clusters	número inteiro	Default=15
num_clusters	integer	Default=5
clustering_criterion	AIC BIC	
automatic_clustering_method	use_clustering_criterion_setting Distance_jump Mínimo Máximo	
feature_importance_method	use_clustering_criterion_setting effect_size	
use_random_seed	Booleano	
random_seed	integer	
distance_measure	Euclidiano Loglikelihood	
include_outlier_clusters	Booleano	Default=True
num_cases_in_feature_tree_leaf_is_less_than	integer	Default=10
top_perc_outliers	integer	Default=5
initial_dist_change_threshold	integer	Default=0
leaf_node_maximum_branches	integer	Default=8
non_leaf_node_maximum_branches	integer	Default=8
max_tree_depth	integer	Default=3
adjustment_weight_on_measurement_level	integer	Default=6
memory_allocation_mb	number	Default=512
delayed_split	Booleano	Default=True
fields_to_standardize	[*f1 ... fN*]	

Tabela 136. Propriedades de twostepAS (continuação)

Propriedades de twostepAS	**Valores**	**Descrição da propriedade**
adaptive_feature_selection	Booleano	Default=True
featureMisPercent	integer	Default=70
coefRange	number	Default=0.05
percCasesSingleCategory	integer	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
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 69 para obter mais informações.

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 como criar script do próprio nó de modelagem, consulte "Propriedades de anomalydetectionnode" na página 161.

Tabela 137. Propriedades de applyanomalydetectionnode.

Propriedades de **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	*sinalizador*	Indica se os registros são descartados a partir da saída ou não.
discard_anomalous_records	*sinalizador*	Indicador que determina se registros anômalos ou *não* anômalos devem ser descartados. O padrão é off, o que significa que registros *não* anômalos são descartados. Caso contrário, se for on, registros anômalos serão descartados. Esta propriedade será ativada somente se a propriedade discard_records for 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 como criar script do próprio nó de modelagem, consulte "Propriedades de apriorinode" na página 163.

Tabela 138. Propriedades de applyapriorinode.

Propriedades de **applyapriorinode**	Valores	Descrição da propriedade
max_predictions	*número (inteiro)*	
ignore_unmatached	*sinalizador*	
allow_repeats	*sinalizador*	
check_basket	NoPredictions Predictions NoCheck	
criterion	Confidence Suporte 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 164.

Tabela 139. Propriedades de applyassociationrulesnode

Propriedades de applyassociationrulesnode	**Tipo de dados**	**Descrição da propriedade**
max_predictions	*integer*	O número máximo de regras que podem ser aplicadas a cada entrada na escoragem.
criterion	Confidence Rulesupport Lift Conditionsupport Implementabilidade	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 de script neste nugget do modelo é *applyautoclassifiernode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de autoclassifiernode" na página 166

Tabela 140. Propriedades de applyautoclassifiernode.

Propriedades de applyautoclassifiernode	**Valores**	**Descrição da propriedade**
flag_ensemble_method	Voting 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_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 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.

Tabela 140. Propriedades de applyautoclassifiernode (continuação).

Propriedades de **applyautoclassifiernode**	Valores	Descrição da propriedade
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 169.

Propriedades de applyautonumericnode

Os nós de modelagem Numeração Automática podem ser utilizados para gerar um nugget do modelo de Numeração Automática. O nome de script deste nugget do modelo é *applyautonumericnode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de autonumericnode" na página 170

Tabela 141. Propriedades de applyautonumericnode.

Propriedades de **applyautonumericnode**	Valores	Descrição da propriedade
calculate_standard_error	*sinalizador*	

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 172.

Tabela 142. Propriedades de applybayesnetnode.

Propriedades de **applybayesnetnode**	Valores	Descrição da propriedade
all_probabilities	*sinalizador*	
raw_propensity	*sinalizador*	
adjusted_propensity	*sinalizador*	
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	

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 174.

Tabela 143. Propriedades de applyc50node.

Propriedades de applyc50node	Valores	Descrição da propriedade
sql_generate	Never NoMissingValues	Usado para configurar as opções de geração de SQL durante a execução do conjunto de regras.
calculate_conf	*flag*	Disponível quando a geração de SQL está ativada; essa propriedade inclui cálculos de confiança na árvore gerada.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

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 de carmanode" na página 175.

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 176.

Tabela 144. Propriedades de applycartnode.

Propriedades de applycartnode	Valores	Descrição da propriedade
sql_generate	Never MissingValues NoMissingValues	Usado para configurar as opções de geração de SQL durante a execução do conjunto de regras.
calculate_conf	*flag*	Disponível quando a geração de SQL está ativada; essa propriedade inclui cálculos de confiança na árvore gerada.
display_rule_id	*flag*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

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 de chaidnode" na página 178.

Tabela 145. Propriedades de applychaidnode.

Propriedades de applychaidnode	Valores	Descrição da propriedade
sql_generate	Never MissingValues	
calculate_conf	*flag*	
display_rule_id	*flag*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

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 de coxregnode" na página 180.

Tabela 146. Propriedades de applycoxregnode.

Propriedades de applycoxregnode	Valores	Descrição da propriedade
future_time_as	Intervalos Campos	
time_interval	*número*	
num_future_times	*número inteiro*	
time_field	*campo*	
past_survival_time	*campo*	
all_probabilities	*sinalizador*	
cumulative_hazard	*sinalizador*	

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 de decisionlistnode" na página 182.

Tabela 147. Propriedades de applydecisionlistnode.

Propriedades de applydecisionlistnode	Valores	Descrição da propriedade
enable_sql_generation	*sinalizador*	Quando é true, o IBM SPSS Modeler tenta enviar por push o modelo de Lista de Decisão de volta para SQL.
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	

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 de discriminantnode" na página 183.

Tabela 148. Propriedades de applydiscriminantnode.

Propriedades de applydiscriminantnode	Valores	Descrição da propriedade
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	

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 de factornode" na página 185.

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 187.

Tabela 149. Propriedades de applyfeatureselectionnode.

Propriedades de 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 188.

Tabela 150. Propriedades de applygeneralizedlinearnode.

Propriedades de applygeneralizedlinearnode	Valores	Descrição da propriedade
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	

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 192.

Tabela 151. Propriedades de applyglmmnode.

Propriedades de 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.
score_category_probabilities	*sinalizador*	Se configurado como True, produzirá as probabilidades previstas para variáveis resposta categórica. Um campo é criado para cada categoria. O padrão é False.
max_categories	*número inteiro*	Número máximo de categorias para as quais as probabilidades serão previstas. Utilizado apenas se score_category_probabilities for True.
score_propensity	*sinalizador*	Se configurado como True, produzirá escores de propensão bruta (probabilidade do resultado de "True") para modelos com destinos de sinalizador. Se as partições estiverem em vigor, também produzirá escores de propensão ajustada com base na partição de teste. O padrão é False.

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 de kmeansnode" na página 195.

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 196.

Tabela 152. Propriedades de applyknnnode.

Propriedades de applyknnnode	Valores	Descrição da propriedade
all_probabilities	*sinalizador*	
save_distances	*sinalizador*	

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 174.

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 199.

Tabela 153. Propriedades de applylinearnode.

Propriedades de linear	Valores	Descrição da propriedade
use_custom_name	*flag*	
custom_name	*sequência*	
enable_sql_generation	*flag*	

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 200.

Tabela 154. Propriedades de applylinearasnode

Propriedades de 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 201.

Tabela 155. Propriedades de applylogregnode.

Propriedades de applylogregnode	Valores	Descrição da propriedade
calculate_raw_propensities	*flag*	
calculate_conf	*flag*	
enable_sql_generation	*flag*	

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 206.

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 156. Propriedades de applyneuralnetnode.

Propriedades de applyneuralnetnode	Valores	Descrição da propriedade
calculate_conf	*sinalizador*	Disponível quando a geração de SQL está ativada; essa propriedade inclui cálculos de confiança na árvore gerada.
enable_sql_generation	*sinalizador*	
nn_score_method	Difference SoftMax	
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	

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 como criar o script do próprio nó de modelagem, consulte "Propriedades de neuralnetworknode" na página 208.

Tabela 157. Propriedades de applyneuralnetworknode

Propriedades de applyneuralnetworknode	Valores	Descrição da propriedade
use_custom_name	*flag*	
custom_name	*string*	
confidence	onProbability onIncrease	
score_category_probabilities	*flag*	
max_categories	*number*	
score_propensity	*flag*	

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 209.

Tabela 158. Propriedades de applyquestnode.

Propriedades de applyquestnode	Valores	Descrição da propriedade
sql_generate	Never MissingValues NoMissingValues	
calculate_conf	*flag*	
display_rule_id	*flag*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
calculate_raw_propensities	*flag*	
calculate_adjusted_propensities	*flag*	

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 173.

Tabela 159. Propriedades de applyr

Propriedades de applyr	Valores	Descrição da Propriedade
score_syntax	*string*	Sintaxe do script R para escoragem de modelo.
convert_flags	StringsAndDoubles LogicalValues	Opção para converter os campos de sinalização.
convert_datetime	*sinalizador*	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	*sinalizador*	Opção para converter valores omissos em valor NA R.

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 211.

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 214.

Tabela 160. Propriedades de applyselflearningnode.

Propriedades de applyselflearningnode	Valores	Descrição da propriedade
max_predictions	*número*	
randomization	*número*	
scoring_random_seed	*número*	
sort	ascending descending	Especifica se ofertas com as escoragens mais altas ou mais baixas serão exibidas primeiro.
model_reliability	*sinalizador*	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 de sequencenode" na página 213.

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 219.

Tabela 161. Propriedades de applysvmnode.

Propriedades de applysvmnode	Valores	Descrição da propriedade
all_probabilities	*sinalizador*	
calculate_raw_propensities	*sinalizador*	
calculate_adjusted_propensities	*sinalizador*	

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 de stpnode" na página 215.

Tabela 162. Propriedades de applystpnode

Propriedades de 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 220.

Tabela 163. Propriedades de applytcmnode

Propriedades de applytcmnode	Valores	Descrição da propriedade
ext_future	*boolean*	
ext_future_num	*integer*	
noise_res	*boolean*	
conf_limits	*boolean*	
target_fields	*lista*	
target_series	*lista*	

Propriedades de applytimeseriesnode

Os nós de modelagem Séries Temporais podem ser utilizados para gerar um nugget do modelo Séries Temporais. 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 de timeseriesnode" na página 224.

Tabela 164. Propriedades de applytimeseriesnode.

Propriedades de applytimeseriesnode	Valores	Descrição da propriedade
calculate_conf	*sinalizador*	
calculate_residuals	*sinalizador*	

Propriedades de applytreeasnode

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 *applytreenode*. Para obter mais informações sobre como criar script do próprio nó de modelagem, consulte "Propriedades de treeasnode" na página 226.

Tabela 165. Propriedades de applytreeasnode

Propriedades de applytreeasnode	Valores	Descrição da propriedade
calculate_conf	*flag*	Esta propriedade inclui cálculos confiança na árvore gerada.
display_rule_id	*flag*	Inclui um campo na saída de escoragem que indica o ID do nó terminal para o qual cada registro é designado.
sql_generate	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 228.

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*. 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 twostepAS" na página 229.

Capítulo 15. Propriedades do Nó de Modelagem de Banco de Dados

O IBM SPSS Modeler suporta integração com mineração de dados e com as ferramentas de modelagem disponíveis de fornecedores de banco de dados, incluindo o Microsoft SQL Server Analysis Services, Oracle Data Mining, IBM DB2 InfoSphere Warehouse e o 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 166. Propriedades comuns do nó Microsoft

Propriedades Comuns do Nó Microsoft	Valores	Descrição da Propriedade
`analysis_database_name`	*string*	Nome do banco de dados de Serviços de Análise.
`analysis_server_name`	*string*	Nome do host de Serviços de Análise.
`use_transactional_data`	*flag*	Especifica se os dados de entrada estão em formato tabular ou transacional.
`inputs`	*lista*	Campos de entrada de dados tabulares.

Tabela 166. 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 245 para obter mais informações.
with_drillthrough	*flag*	Com a opção Drill through.

Árvore de Decisão da MS

Não há propriedades específicas definidas para nós do tipo mstreenode. Consulte as propriedades comuns da Microsoft no início desta seção.

Cluster da 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 167. Propriedades de msassocnode

Propriedades de 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 168. Propriedades de mssequenceclusternode

Propriedades de 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 da Microsoft possui parâmetros específicos que podem ser configurados utilizando 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 da MS

Tabela 169. Propriedades da Árvore de Decisão da MS.

Propriedades de 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.

Tabela 169. Propriedades da Árvore de Decisão da MS (continuação).

Propriedades de applymstreenode	Valores	Descrição
datasource	*sequência*	Nome da origem de dados (DSN) do ODBC do SQL Server.
sql_generate	*flag*	Ativa a geração de SQL.

Regressão linear da MS

Tabela 170. Propriedades da Regressão Linear da MS.

Propriedades de 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 171. Propriedades de Rede Neural da MS.

Propriedades de 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 172. Propriedades da Regressão Logística da MS.

Propriedades de 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 173. Séries Temporais da MS.

Propriedades de 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.

Tabela 173. Séries Temporais da MS (continuação).

Propriedades de applymstimeseriesnode	**Valores**	**Descrição**
start_from	new_prediction historical_ prediction	Especifica se predições futuras ou predições históricas devem ser feitas
new_step	*número*	Define período de tempo inicial para predições futuras.
historical_step	*número*	Define período de tempo inicial para predições históricas.
end_step	*número*	Define período de tempo final para predições.

Cluster de Sequências da MS

Tabela 174. Propriedades de Armazenamento em Cluster de Sequências da MS.

Propriedades de 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 175. Propriedades comuns do nó Oracle.

Propriedades Comuns do Nó Oracle	**Valores**	**Descrição da Propriedade**
target	*field*	
inputs	*Lista de campos*	
partition	*field*	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	*sinalizador*	
model_name	*sequência*	Nome customizado para o novo modelo.
use_partitioned_data	*sinalizador*	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	*field*	

Tabela 175. Propriedades comuns do nó Oracle (continuação).

Propriedades Comuns do Nó Oracle	Valores	Descrição da Propriedade
auto_data_prep	*sinalizador*	Ativa ou desativa o recurso de preparação automática de dados da Oracle (apenas bancos de dados 11g).
costs	*structured*	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	*sinalizador*	
prediction_probability	*sequência*	
use_prediction_set	*sinalizador*	

Naive Bayes da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oranbnode.

Tabela 176. Propriedades de oranbnode.

Propriedades de oranbnode	Valores	Descrição da Propriedade
singleton_threshold	*número*	0.0 a 1.0.*
pairwise_threshold	*número*	0.0 a 1.0.*
priors	Data Equal Custom	
custom_priors	*structured*	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.

Adaptive Bayes da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oraabnnode.

Tabela 177. Propriedades de oraabnnode.

Propriedades de oraabnnode	Valores	Descrição da Propriedade
model_type	SingleFeature MultiFeature NaiveBayes	
use_execution_time_limit	*sinalizador*	*
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	

Tabela 177. Propriedades de oraabnnode (continuação).

Propriedades de oraabnnode	Valores	Descrição da Propriedade
custom_priors	*structured*	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 178. Propriedades de orasvmnode.

Propriedades de orasvmnode	Valores	Descrição da Propriedade
active_learning	Enable Disable	
kernel_function	Linear Gaussiano System	
normalization_method	zscore minmax none	
kernel_cache_size	*número inteiro*	Apenas kernel gaussiano. O valor deve ser maior que 0.*
convergence_tolerance	*número*	O valor deve ser maior que 0.*
use_standard_deviation	*sinalizador*	Apenas kernel gaussiano.*
standard_deviation	*número*	O valor deve ser maior que 0.*
use_epsilon	*sinalizador*	Apenas modelos de regressão.*
epsilon	*número*	O valor deve ser maior que 0.*
use_complexity_factor	*sinalizador*	*
complexity_factor	*número*	*
use_outlier_rate	*sinalizador*	Apenas variante de Classe Um.*
outlier_rate	*número*	Apenas variante de Classe Um. 0.0 a 1.0.*
weights	Data Equal Custom	
custom_weights	*structured*	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 179. Propriedades e oraglmnode.

Propriedades de oraglmnode	Valores	Descrição da Propriedade
normalization_method	zscore minmax none	
missing_value_handling	ReplaceWithMean UseCompleteRecords	
use_row_weights	*sinalizador*	*
row_weights_field	*field*	*
save_row_diagnostics	*sinalizador*	*
row_diagnostics_table	*sequência*	*
coefficient_confidence	*número*	*
use_reference_category	*sinalizador*	*
reference_category	*sequência*	*
ridge_regression	Auto Off On	*
parameter_value	*número*	*
vif_for_ridge	*sinalizador*	*

* 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 180. Propriedades de oradecisiontreenode.

Propriedades de oradecisiontreenode	Valores	Descrição da Propriedade
use_costs	*sinalizador*	
impurity_metric	Entropy Gini	
term_max_depth	*número inteiro*	2–20.*
term_minpct_node	*número*	0.0–10.0.*
term_minpct_split	*número*	0.0–20.0.*
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	*sinalizador*	*

* A propriedade será ignorada se mode for configurado para Simple.

Cluster-O da Oracle

As propriedades a seguir estão disponíveis para nós do tipo oraoclusternode.

Tabela 181. Propriedades de oraoclusternode.

Propriedades de oraoclusternode	Valores	Descrição da Propriedade
max_num_clusters	*número inteiro*	O valor deve ser maior que 0.
max_buffer	*número inteiro*	O valor deve ser maior que 0.*
sensitivity	*número*	0.0 a 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 182. Propriedades de orakmeansnode.

Propriedades de orakmeansnode	Valores	Descrição da Propriedade
num_clusters	*número inteiro*	O valor deve ser maior que 0.
normalization_method	zscore minmax none	
distance_function	Euclidiano Cosseno	
iterations	*número inteiro*	0–20.*
conv_tolerance	*número*	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	*número*	0.0 a 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 183. Propriedades de oranmfnode.

Propriedades de oranmfnode	Valores	Descrição da Propriedade
normalization_method	minmax none	
use_num_features	*sinalizador*	*
num_features	*número inteiro*	0–1. O valor padrão é estimado a partir dos dados pelo algoritmo.*
random_seed	*número*	*
num_iterations	*número inteiro*	0–500.*
conv_tolerance	*número*	0.0–0.5.*
display_all_features	*sinalizador*	*

* 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 184. Propriedades de oraapriorinode.

Propriedades de oraapriorinode	Valores	Descrição da Propriedade
content_field	*field*	
id_field	*field*	
max_rule_length	*número inteiro*	2–20.
min_confidence	*número*	0.0–1.0.
min_support	*número*	0.0–1.0.
use_transactional_data	*sinalizador*	

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 185. Propriedades de oraainode.

Propriedades de oraainode	Valores	Descrição da Propriedade
custom_fields	*sinalizador*	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	*sinalizador*	Quando selection_mode for configurado para ImportanceLevel, especifica se campos importantes devem ser selecionados.
important_label	*sequência*	Especifica o rótulo para a classificação "importante".
select_marginal	*sinalizador*	Quando selection_mode for configurado para ImportanceLevel, especifica se campos marginais devem ser selecionados.
marginal_label	*sequência*	Especifica o rótulo para a classificação "marginal".
important_above	*número*	0.0–1.0.
select_unimportant	*sinalizador*	Quando selection_mode for configurado para ImportanceLevel, especifica se campos não importantes devem ser selecionados.
unimportant_label	*sequência*	Especifica o rótulo para a classificação "não importante".
unimportant_below	*número*	0.0–1.0.

Tabela 185. Propriedades de oraainode (continuação).

Propriedades de oraainode	Valores	Descrição da Propriedade
importance_value	*número*	Quando selection_mode for configurado para ImportanceValue, especifica o valor de corte a ser utilizado. Aceita valores de 0 a 100.
top_n	*número*	Quando selection_mode for configurado para TopN, especifica o valor de corte a ser utilizado. 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.

Naive Bayes da Oracle

Não há propriedades específicas definidas para nós do tipo applyoranbnode.

Adaptive Bayes da Oracle

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 186. Propriedades do applyoradecisiontreenode

Propriedades de applyoradecisiontreenode	Valores	Descrição da Propriedade
use_costs	*flag*	
display_rule_ids	*flag*	

Cluster-O da 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 187. Propriedades de applyoranmfnode

Propriedades de applyoranmfnode	Valores	Descrição da Propriedade
display_all_features	*flag*	

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 de Modelagem para o IBM DB2

Propriedades do Nó de Modelagem IBM DB2

As propriedades a seguir são comuns aos nós de modelagem de banco de dados do IBM InfoSphere Warehouse (ISW).

Tabela 188. Propriedades comuns do nó ISW.

Propriedades comuns do nó ISW	Valores	Descrição da Propriedade
entradas	*Lista de campos*	
datasource		
username		
password		
epassword		
enable_power_options	*flag*	
power_options_max_memory	*integer*	O valor deve ser maior que 32.
power_options_cmdline	*string*	
mining_data_custom_sql	*string*	
logical_data_custom_sql	*string*	
mining_settings_custom_sql		

Árvore de Decisão da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imtreenode.

Tabela 189. Propriedades de db2imtreenode.

Propriedades de db2imtreenode	Valores	Descrição da Propriedade
target	*field*	
perform_test_run	*sinalizador*	
use_max_tree_depth	*sinalizador*	
max_tree_depth	*número inteiro*	Valor maior que 0.
use_maximum_purity	*sinalizador*	
maximum_purity	*número*	Número entre 0 e 100.
use_minimum_internal_cases	*sinalizador*	
minimum_internal_cases	*número inteiro*	Valor maior que 1.
use_costs	*sinalizador*	
costs	*structured*	Propriedade estruturada no formato: [[drugA drugB 1.5] [drugA drugC 2.1]], em que os argumentos entre [] são custos previstos reais.

Associação da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imassocnode.

Tabela 190. Propriedades de db2imassocnode.

Propriedades de db2imassocnode	Valores	Descrição da Propriedade
use_transactional_data	*sinalizador*	
id_field	*field*	
content_field	*field*	
data_table_layout	basic limited_length	
max_rule_size	*número inteiro*	O valor deve ser maior que 2.
min_rule_support	*número*	0% a 100%
min_rule_confidence	*número*	0% a 100%
use_item_constraints	*sinalizador*	
item_constraints_type	Include Exclude	
use_taxonomy	*sinalizador*	
taxonomy_table_name	*sequência*	O nome da tabela DB2 para armazenar detalhes de taxonomia.
taxonomy_child_column_name	*sequência*	O nome da coluna filha na tabela de taxonomia. A coluna filha contém os nomes dos itens ou os nomes das categorias.
taxonomy_parent_column_name	*sequência*	O nome da coluna pai na tabela de taxonomia. A coluna pai contém os nomes de categoria.
load_taxonomy_to_table	*sinalizador*	Controla se as informações de taxonomia armazenadas no IBM SPSS Modeler devem ser transferidas por upload para a tabela de taxonomia no tempo de construção de modelo. Observe que a tabela de taxonomia será eliminada se ela já existir. As informações de taxonomia são armazenadas com o nó de construção de modelo e podem ser editadas utilizando os botões **Editar Categorias** e **Editar Taxonomia**.

Sequência da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imsequencenode.

Tabela 191. Propriedades de db2imsequencenode.

Propriedades de db2imsequencenode	Valores	Descrição da Propriedade
id_field	*field*	
group_field	*field*	
content_field	*field*	
max_rule_size	*número inteiro*	O valor deve ser maior que 2.
min_rule_support	*número*	0% a 100%
min_rule_confidence	*número*	0% a 100%
use_item_constraints	*sinalizador*	
item_constraints_type	Include Exclude	

Tabela 191. Propriedades de db2imsequencenode (continuação).

Propriedades de db2imsequencenode	Valores	Descrição da Propriedade
use_taxonomy	*sinalizador*	
taxonomy_table_name	*sequência*	O nome da tabela DB2 para armazenar detalhes de taxonomia.
taxonomy_child_column_name	*sequência*	O nome da coluna filha na tabela de taxonomia. A coluna filha contém os nomes dos itens ou os nomes das categorias.
taxonomy_parent_column_name	*sequência*	O nome da coluna pai na tabela de taxonomia. A coluna pai contém os nomes de categoria.
load_taxonomy_to_table	*sinalizador*	Controla se as informações de taxonomia armazenadas no IBM SPSS Modeler devem ser transferidas por upload para a tabela de taxonomia no tempo de construção de modelo. Observe que a tabela de taxonomia será eliminada se ela já existir. As informações de taxonomia são armazenadas com o nó de construção de modelo e podem ser editadas utilizando os botões **Editar Categorias** e **Editar Taxonomia**.

Regressão da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imregnode.

Tabela 192. Propriedades de db2imregnode.

Propriedades de db2imregnode	Valores	Descrição da Propriedade
target	*field*	
regression_method	transform linear polynomial rbf	Consulte a próxima tabela para obter propriedades que se aplicarão somente se regression_method for configurado para rbf.
perform_test_run	*field*	
limit_rsquared_value	*sinalizador*	
max_rsquared_value	*número*	Valor entre 0,0 e 1,0.
use_execution_time_limit	*sinalizador*	
execution_time_limit_mins	*número inteiro*	Valor maior que 0.
use_max_degree_polynomial	*sinalizador*	
max_degree_polynomial	*número inteiro*	
use_intercept	*sinalizador*	
use_auto_feature_selection_method	*sinalizador*	
auto_feature_selection_method	normal adjusted	
use_min_significance_level	*sinalizador*	
min_significance_level	*número*	
use_min_significance_level	*sinalizador*	

As propriedades a seguir se aplicarão apenas se regression_method for configurado para rbf.

Tabela 193. Propriedades de db2imregnode se regression_method for configurado para rbf.

Propriedades de db2imregnode	Valores	Descrição da Propriedade
use_output_sample_size	*sinalizador*	Se true, configura automaticamente o valor como o padrão.
output_sample_size	*número inteiro*	O padrão é 2. O mínimo é 1.
use_input_sample_size	*sinalizador*	Se true, configura automaticamente o valor como o padrão.
input_sample_size	*número inteiro*	O padrão é 2. O mínimo é 1.
use_max_num_centers	*sinalizador*	Se true, configura automaticamente o valor como o padrão.
max_num_centers	*número inteiro*	O padrão é 20. O mínimo é 1.
use_min_region_size	*sinalizador*	Se true, configura automaticamente o valor como o padrão.
min_region_size	*número inteiro*	O padrão é 15. O mínimo é 1.
use_max_data_passes	*sinalizador*	Se true, configura automaticamente o valor como o padrão.
max_data_passes	*número inteiro*	O padrão é 5. O mínimo é 2.
use_min_data_passes	*sinalizador*	Se true, configura automaticamente o valor como o padrão.
min_data_passes	*número inteiro*	O padrão é 5. O mínimo é 2.

Cluster da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imclusternode.

Tabela 194. Propriedades de db2imclusternode.

Propriedades de db2imclusternode	**Valores**	**Descrição da Propriedade**
cluster_method	demographic kohonen birch	
kohonen_num_rows	*número inteiro*	
kohonen_num_columns	*número inteiro*	
kohonen_passes	*número inteiro*	
use_num_passes_limit	*sinalizador*	
use_num_clusters_limit	*sinalizador*	
max_num_clusters	*número inteiro*	Valor maior que 1.

Tabela 194. Propriedades de db2imclusternode (continuação).

Propriedades de db2imclusternode	Valores	Descrição da Propriedade
birch_dist_measure	log_likelihood euclidean	O padrão é log_likelihood.
birch_num_cfleaves	*número inteiro*	O padrão é 1000.
birch_num_refine_passes	*número inteiro*	O padrão é 3 e o mínimo é 1.
use_execution_time_limit	*sinalizador*	
execution_time_limit_mins	*número inteiro*	Valor maior que 0.
min_data_percentage	*número*	0% a 100%
use_similarity_threshold	*sinalizador*	
similarity_threshold	*número*	Valor entre 0,0 e 1,0.

Naive Bayes da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imnbsnode.

Tabela 195. Propriedades de db2imnbnode.

Propriedades de db2imnbnode	Valores	Descrição da Propriedade
perform_test_run	*sinalizador*	
probability_threshold	*número*	O padrão é 0,001. O valor mínimo é 0 e o valor máximo é 1.000
use_costs	*sinalizador*	
costs	*structured*	Propriedade estruturada no formato: [[drugA drugB 1.5] [drugA drugC 2.1]], em que os argumentos entre [] são custos previstos reais.

Regressão logística da ISW

As propriedades a seguir estão disponíveis para nós do tipo db2imlognode.

Tabela 196. Propriedades de db2imlognode.

Propriedades de db2imlognode	Valores	Descrição da Propriedade
perform_test_run	*sinalizador*	
use_costs	*sinalizador*	
costs	*structured*	Propriedade estruturada no formato: [[drugA drugB 1.5] [drugA drugC 2.1]], em que os argumentos entre [] são custos previstos reais.

Séries temporais da ISW

Nota: O parâmetro de campos de entrada não é utilizado para este nó. Se o parâmetro de campos de entrada for localizado no script, um aviso será exibido para informar que o nó possui *hora* e *destinos* como campos de entrada, mas não há campos de entrada.

As propriedades a seguir estão disponíveis para nós do tipo db2imtimeseriesnode.

Tabela 197. Propriedades de db2imtimeseriesnode.

Propriedades de db2imtimeseriesnode	**Valores**	**Descrição da Propriedade**
time	*field*	Número inteiro, hora ou data permitidos.
destino	*lista de campos*	
forecasting_algorithm	arima exponential_ smoothing seasonal_trend_ decomposition	
forecasting_end_time	auto número inteiro date time	
use_records_all	*booleano*	Se false, use_records_start e use_records_end deverão ser configurados.
use_records_start	*número inteiro / hora / data*	Depende do tipo de campo de hora
use_records_end	*número inteiro / hora / data*	Depende do tipo de campo de hora
interpolation_method	nenhum linear exponential_splines cubic_splines	

Propriedades de Nugget do Modelo do IBM DB2

As propriedades a seguir são para os nuggets do modelo criados utilizando os modelos do IBM DB2 ISW.

Árvore de Decisão da ISW

Não há propriedades específicas definidas para nós do tipo applydb2imtreenode.

Associação da ISW

O nugget do modelo não pode ser aplicado no script.

Sequência da ISW

O nugget do modelo não pode ser aplicado no script.

Regressão da ISW

Não há propriedades específicas definidas para nós do tipo applydb2imregnode.

Cluster da ISW

Não há propriedades específicas definidas para nós do tipo applydb2imclusternode.

Naive Bayes da ISW

Não há propriedades específicas definidas para nós do tipo applydb2imnbnode.

Regressão logística da ISW

Não há propriedades específicas definidas para nós do tipo `applydb2imlognode`.

Séries temporais da ISW

O nugget do modelo não pode ser aplicado no script.

Propriedades do Nó de Modelagem para IBM Netezza Analytics

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 198. Propriedades comuns do nó Netezza.

Propriedades Comuns do Nó Netezza	Valores	Descrição da Propriedade
`custom_fields`	*sinalizador*	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.
`target`	*campo*	Campo de destino (contínuo ou categórico).
`record_id`	*campo*	Campo a ser utilizado como identificador de registro exclusivo.
`use_upstream_connection`	*sinalizador*	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`	*sinalizador*	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`	*sinalizador*	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`	*sinalizador*	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 199. Propriedades de netezzadectreenode.

Propriedades de `netezzadectreenode`	Valores	Descrição da Propriedade
`impurity_measure`	`Entropia` `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`	*número*	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`	*número*	Se `prune_tree_options` for configurado para `partitionTrainingData`, 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 `partitionTrainingData`; o padrão é 1.
`pruning_table`	*sequência*	Nome da tabela de um conjunto de remoção separado para estimar a precisão do modelo.

Tabela 199. Propriedades de netezzadectreenode (continuação).

Propriedades de netezzadectreenode	Valores	Descrição da Propriedade
compute_probabilities	*sinalizador*	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 200. Propriedades de netezzakmeansnode.

Propriedades de netezzakmeansnode	Valores	Descrição da Propriedade
distance_measure	Euclidiano 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 201. Propriedades de netezzabayesnode.

Propriedades de netezzabayesnode	Valores	Descrição da Propriedade
base_index	*número inteiro*	Identificador numérico designado ao primeiro campo de entrada para gerenciamento interno; o padrão é 777.
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	*sinalizador*	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 202. Propriedades de netezzanaivebayesnode.

Propriedades de netezzanaivebayesnode	Valores	Descrição da Propriedade
compute_probabilities	*sinalizador*	Se true, produz um campo de nível de confiança (probabilidade), assim como o campo de predição.
use_m_estimation	*sinalizador*	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 203. Propriedades de netezzaknnnode.

Propriedades de `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`	`Euclidiano`
	`Manhattan`
	`Canberra`
	`Máximo`
		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.
`standardize_measurements`	*sinalizador*	Se true, padroniza as medições para campos de entrada contínuos antes de calcular valores de distância.
`use_coresets`	*sinalizador*	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 204. Propriedades de netezzadivclusternode.

Propriedades de `netezzadivclusternode`	Valores	Descrição da Propriedade

`distance_measure`	`Euclidiano`
	`Manhattan`
	`Canberra`
	`Máximo`
		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 205. Propriedades de netezzapcanode.

Propriedades de netezzapcanode	Valores	Descrição da Propriedade
center_data	*sinalizador*	Se true (padrão), executa a centralização de dados (também conhecida como "subtração média") antes da análise.
perform_data_scaling	*sinalizador*	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	*sinalizador*	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 206. Propriedades de netezzaregtreenode.

Propriedades de 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	*número*	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.
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	*número*	Se prune_tree_options for configurado para PercTrainingData, especifica a porcentagem de dados a serem utilizados para treinamento.

Tabela 206. Propriedades de netezzaregtreenode (continuação).

Propriedades de netezzaregtreenode	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 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	*sinalizador*	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 207. Propriedades de netezzalineregressionnode.

Propriedades de netezzalineregressionnode	Valores	Descrição da Propriedade
use_svd	*sinalizador*	Se true, utiliza a matriz de Decomposição em Valores Singulares ao invés da matriz original para maior velocidade e precisão numérica.
include_intercept	*sinalizador*	Se true (padrão), aumenta a precisão geral da solução.
calculate_model_diagnostics	*sinalizador*	Se true, calcula os diagnósticos no modelo.

Série de tempo Netezza

As propriedades a seguir estão disponíveis para nós do tipo netezzatimeseriesnode.

Tabela 208. Propriedades de netezzatimeseriesnode.

Propriedades de 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.

Tabela 208. Propriedades de netezzatimeseriesnode (continuação).

Propriedades de netezzatimeseriesnode	**Valores**	**Descrição da Propriedade**
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.
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

earliest_time
latest_time
	integer
	date
	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'`

Tabela 208. Propriedades de netezzatimeseriesnode (continuação).

Propriedades de 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.
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'

Tabela 208. Propriedades de netezzatimeseriesnode (continuação).

Propriedades de netezzatimeseriesnode	**Valores**	**Descrição da Propriedade**
period	*número inteiro*	Comprimento do ciclo sazonal, especificado em conjunto com units_period. Não aplicável para análise espectral.
units_period	ms s mín. 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	*integer* *date* *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.
forecast_times	*integer* *date* *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	*sinalizador*	Identifica se valores históricos devem ser incluídos na saída.
include_interpolated_values	*sinalizador*	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 209. Propriedades de netezzaglmnode.

Propriedades de **netezzaglmnode**	Valores	Descrição da Propriedade

`dist_family`	`bernoulli`
	`gaussian`
	`poisson`
	`negativebinomial`
	`wald`
	`gamma`
		Tipo de distribuição; o padrão é `berneui`.

`dist_params`	*número*	Valor de parâmetro de distribuição a ser utilizado. Aplicável apenas se `distribution` for `Negativebinomial`.
`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`	*field*	Nome da tabela de banco de dados na qual o modelo linear generalizado Netezza será armazenado.
`maxit`	*integer*	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`	*número*	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.

`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`.

Tabela 209. Propriedades de netezzaglmnode (continuação).

Propriedades de netezzaglmnode	Valores	Descrição da Propriedade
link_params	*number*	Valor do parâmetro da função de ligação a ser utilizado. Aplicável apenas se link_function for power ou oddspower.
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	*sinalizador*	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 210. Propriedades comuns do nugget do modelo Netezza

Propriedades Comuns do Nugget do Modelo Netezza	Valores	Descrição da Propriedade
connection	*string*	A sequência de conexões com o banco de dados Netezza no qual o modelo é armazenado.
table_name	*string*	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 211. Nomes de script de nuggets do modelo Netezza

Nugget do Modelo	Nome de Script
Árvore de Decisão	applynetezzadectreenode
K-Médias	applynetezzakmeansnode
Rede Bayesiana	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 entre si.

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 212. Propriedades de analysisnode.

Propriedades de 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	*sinalizador*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser utilizado.
output_format	Text (*.txt*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída.
by_fields	*lista*	
full_filename	*sequência*	Se for saída de disco, de dados ou HTML, o nome do arquivo de saída.

Tabela 212. Propriedades de analysisnode (continuação).

Propriedades de analysisnode	Tipo de dados	Descrição da propriedade
coincidence	*sinalizador*	
performance	*sinalizador*	
evaluation_binary	*flag*	
confidence	*sinalizador*	
threshold	*número*	
improve_accuracy	*número*	
inc_user_measure	*sinalizador*	
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 e podem ser classificados e utilizados 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 213. Propriedades de dataauditnode.

Propriedades de dataauditnode	Tipo de dados	Descrição da propriedade
custom_fields	*flag*	
fields	*[field1 ... fieldN]*	
overlay	*field*	
display_graphs	*flag*	Utilizado para desativar a exibição de gráficos na matriz de saída ativada ou desativada.
basic_stats	*flag*	

Tabela 213. Propriedades de dataauditnode (continuação).

Propriedades de **dataauditnode**	Tipo de dados	Descrição da propriedade
advanced_stats	*flag*	
median_stats	*flag*	
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 utilizado para definir valores discrepantes.
outlier_detection_std_extreme	*number*	Se outlier_detection_method for std, especifica o número a ser utilizado para definir valores extremos.
outlier_detection_iqr_outlier	*número*	Se outlier_detection_method for iqr, especifica o número a ser utilizado para definir valores discrepantes.
outlier_detection_iqr_extreme	*number*	Se outlier_detection_method for iqr, especifica o número a ser utilizado para definir valores extremos.
use_output_name	*flag*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	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	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída.
paginate_output	*flag*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*número*	Quando utilizado com paginate_output, especifica as linhas por página de saída.
full_filename	*sequência*	

Propriedades de matrixnode

O nó Matriz cria uma tabela que mostra relacionamentos entre campos. Ele é usado mais normalmente para mostrar o relacionamento entre dois campos simbólicos, como também pode mostrar relacionamentos entre campos sinalizadores 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 214. Propriedades de matrixnode.

Propriedades de matrixnode	Tipo de dados	Descrição da propriedade
fields	Selected Flags Numerics	
row	*field*	
column	*field*	
include_missing_values	*flag*	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	*string*	
function	Sum Média Min Max SDev	
sort_mode	Não classificado Ascendente Descending	
highlight_top	*número*	Se diferente de zero, então true.
highlight_bottom	*número*	Se diferente de zero, então true.

Tabela 214. Propriedades de matrixnode (continuação).

Propriedades de matrixnode	Tipo de dados	Descrição da propriedade
display	[Counts Expected Residuais RowPct ColumnPct TotalPct]	
include_totals	*flag*	
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	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Output (*.cou*)	Utilizado para especificar o tipo de saída. Os formatos Formatted e Delimited podem selecionar o modificador transposed que transpõe as linhas e as colunas na tabela.
paginate_output	*flag*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*número*	Quando utilizado com paginate_output, especifica as linhas por página de saída.
full_filename	*string*	

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 215. Propriedades de meansnode.

Propriedades de meansnode	Tipo de dados	Descrição da propriedade
means_mode	BetweenGroups BetweenFields	Especifica o tipo de estatística de médias a ser executado nos dados.

Tabela 215. Propriedades de meansnode (continuação).

Propriedades de meansnode	Tipo de dados	Descrição da propriedade
test_fields	[field1 ... fieldn]	Especifica o campo de teste quando means_mode for configurado para BetweenGroups.
grouping_field	*field*	Especifica o campo de agrupamento.
paired_fields	[[field1 field2] [field3 field4] ...]	Especifica os pares de campo a serem utilizados quando means_mode for configurado para BetweenFields.
label_correlations	*sinalizador*	Especifica se os rótulos de correlação são mostrados na saída. Esta configuração se aplica apenas quando means_mode for configurado para 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*	
weak_below_probability	*número*	Quando correlation_mode é configurado para 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	*número*	O valor de corte para correlações fortes.
weak_below_absolute	*número*	Quando correlation_mode é configurado para 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	*número*	O valor de corte para correlações fortes.
unimportant_label	*sequência*	
marginal_label	*sequência*	
important_label	*sequência*	
unimportant_below	*número*	O valor de corte para importância de campo baixa. Este deve ser um valor entre 0 e 1 – por exemplo, 0,90.
important_above	*número*	
use_output_name	*sinalizador*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Nome a ser utilizado.

Tabela 215. Propriedades de meansnode (continuação).

Propriedades de meansnode	Tipo de dados	Descrição da propriedade
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. É possível incluir valores de dados e de outra saída condicional utilizando expressões do CLEM no modelo.

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 216. Propriedades de reportnode.

Propriedades de 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*)	Utilizado para especificar o tipo de saída.
use_output_name	*flag*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	Se use_output_name for true, especifica o nome a ser utilizado.
text	*sequência*	
full_filename	*sequência*	
highlights	*flag*	
title	*string*	
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 217. Propriedades de routputnode

Propriedades de routputnode	Tipo de dados	Descrição da propriedade
syntax	*string*	
convert_flags	StringsAndDoubles LogicalValues	
convert_datetime	*flag*	
convert_datetime_class	POSIXct POSIXlt	
convert_missing	*flag*	
output_name	Automático Personalizado	
custom_name	*string*	
output_to	Screen Arquivo	
output_type	Gráfico Texto	
full_filename	*string*	
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, é possível utilizar esse nó para calcular as estatísticas para um campo chamado *age* e, em seguida, utilizar a média geral de *age* em expressões do CLEM ao inserir 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 218. Propriedades de setglobalsnode.

Propriedades de 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	*sinalizador*	
show_preview	*sinalizador*	

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 219. Propriedades de simevalnode.

Propriedades de simevalnode	Tipo de dados	Descrição da propriedade
target	*campo*	
iteration	*campo*	
presorted_by_iteration	*boolean*	
max_iterations	*number*	
tornado_fields	*[field1...fieldN]*	
plot_pdf	*boolean*	
plot_cdf	*boolean*	
show_ref_mean	*boolean*	
show_ref_median	*boolean*	
show_ref_sigma	*boolean*	
num_ref_sigma	*number*	
show_ref_pct	*boolean*	
ref_pct_bottom	*number*	
ref_pct_top	*number*	
show_ref_custom	*boolean*	
ref_custom_values	*[number1...numberN]*	
category_values	Category Probabilities Both	
category_groups	Categories Iterations	
create_pct_table	*boolean*	
pct_table	Quartiles Intervals Custom	

Tabela 219. Propriedades de simevalnode (continuação).

Propriedades de simevalnode	Tipo de dados	Descrição da propriedade
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 220. Propriedades de simfitnode.

Propriedades de simfitnode	Tipo de dados	Descrição da propriedade
build	Node XMLExport Both	
use_source_node_name	*boolean*	
source_node_name	*string*	O nome customizado do nó de origem que está sendo gerado ou atualizado.
use_cases	Todos LimitFirstN	
use_case_limit	*integer*	
fit_criterion	AndersonDarling KolmogorovSmirnov	
num_bins	*integer*	
parameter_xml_filename	*string*	
generate_parameter_import	*boolean*	

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..." seção
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 221. Propriedades de statisticsnode.

Propriedades de **statisticsnode**	Tipo de dados	Descrição da propriedade
use_output_name	*flag*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	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	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 SErr Median Mode]	
correlation_mode	Probability Absolute	Especifica se as correlações devem ser rotuladas por probabilidade ou valor absoluto.
label_correlations	*flag*	
weak_label	*sequência*	
medium_label	*sequência*	
strong_label	*sequência*	
weak_below_probability	*número*	Quando correlation_mode é configurado para 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	*número*	O valor de corte para correlações fortes.
weak_below_absolute	*número*	Quando correlation_mode é configurado para 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	*número*	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. Esse nó requer uma cópia licenciada do IBM SPSS Statistics.

As propriedades desse nó são descritas em “Propriedades de statisticsoutputnode” na página 300.

Propriedades de tablenode

O nó Tabela exibe os dados em formato de tabela, que também podem ser gravados em um arquivo. Isso é útil a qualquer momento em que precisar inspecionar seu 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 222. Propriedades de tablenode.

Propriedades de 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	*sinalizador*	Especifica se um nome de saída customizado é usado.
output_name	*sequência*	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	Formatted (*.tab*) Delimited (*.csv*) HTML (*.html*) Saída (*.cou*)	Utilizado para especificar o tipo de saída.
transpose_data	*sinalizador*	Transpõe os dados antes de exportar para que as linhas representem campos e as colunas representam registros.
paginate_output	*sinalizador*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*número*	Quando utilizado 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.

Tabela 222. Propriedades de tablenode (continuação).

Propriedades de tablenode	Tipo de dados	Descrição da propriedade
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.
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" MES AAAA t T AAAA ss SM AAAA	Configura o formato de data para o campo (aplica-se apenas aos campos com armazenamento DATE ou TIMESTAMP).

Tabela 222. Propriedades de tablenode (continuação).

Propriedades de 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 de transformnode

O nó Transformar permite selecionar e visualizar os resultados das transformações antes de aplicá-las aos 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 223. Propriedades de transformnode.

Propriedades de 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.

Tabela 223. Propriedades de transformnode (continuação).

Propriedades de transformnode	Tipo de dados	Descrição da propriedade
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	*sequência*	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	*flag*	Quando o output_format é HTML, faz com que a saída seja separada em páginas.
lines_per_page	*número*	Quando utilizado com paginate_output, especifica as linhas por página de saída.
full_filename	*sequência*	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 224. Propriedades comuns do nó de exportação

Propriedade	Valores	Descrição da propriedade
publish_path	*string*	Insere o nome raiz a ser utilizado para os arquivos de imagem e de parâmetro publicados.
publish_metadata	*sinalizador*	Especifica se um arquivo de metadados é produzido, descrevendo as entradas e saídas da imagem e seus modelos de dados.
publish_use_parameters	*sinalizador*	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 Analytic Server permite executar um fluxo no Hadoop Distributed File System (HDFS).

exemplo

```
node = stream.create("asexport", "My node")
node.setPropertyValue("data_source", "Drug1n")
node.setPropertyValue("export_mode", "overwrite")
```

Tabela 225. Propriedades de asexport.

Propriedades de asexport	Tipo de dados	Descrição da propriedade
data_source	*sequência*	O nome da origem de dados.
export_mode	*string*	Especifica anexar (**append**) dados exportados à origem de dados existente ou gravar (**overwrite**) a origem de dados existente.

Propriedades de cognosexportnode

O nó de exportação do IBM Cognos BI exporta dados em um formato que pode ser lido por bancos de dados do Cognos BI.

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 226. Propriedades de cognosexportnode

Propriedades de 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"]` • `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	*string*	O caminho e o nome do pacote do Cognos para o qual você está exportando dados, por exemplo: `/Public Folders/MyPackage`
cognos_datasource	*string*	
cognos_export_mode	`Publish` `ExportFile`	
cognos_filename	*string*	

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 gravar em uma origem de dados ODBC, a origem de dados deverá existir e você deverá ter permissão de gravação para ela.

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 227. Propriedades de databaseexportnode.

Propriedades de databaseexportnode	Tipo de dados	Descrição da propriedade
datasource	*sequência*	
username	*sequência*	
password	*sequência*	

Tabela 227. Propriedades de databaseexportnode (continuação).

Propriedades de databaseexportnode	Tipo de dados	Descrição da propriedade
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 51 para obter mais informações.
table_name	*sequência*	
write_mode	Create Anexar 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.
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	Banco de Dados Add	
drop_existing_table	*flag*	
delete_existing_rows	*flag*	
default_string_size	*número inteiro*	
type		Propriedade estruturada utilizada para configurar o tipo de esquema.
generate_import	*flag*	
use_custom_create_table_command	*flag*	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	*flag*	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	*número*	Especifica o número de registros a serem enviados para o banco de dados antes de confirmar na memória.

Tabela 227. Propriedades de databaseexportnode (continuação).

Propriedades de databaseexportnode	Tipo de dados	Descrição da propriedade
bulk_loading	Off ODBC External	Especifica o tipo de carregamento em massa. Opções adicionais para ODBC e External são listadas abaixo.
not_logged	*flag*	
odbc_binding	Row Column	Especifica a ligação de linha ou a ligação de coluna para carregamento em massa por meio do ODBC.
loader_delimit_mode	Tab Espaço 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	*flag*	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	*flag*	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	*flag*	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	*flag*	Um sinalizador True 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	*flag*	Especifica se um determinado campo é uma chave primária.
use_custom_create_index_command	*flag*	Se true, ativa uma consulta SQL customizada para todos os índices.

Tabela 227. Propriedades de databaseexportnode (continuação).

Propriedades de databaseexportnode	Tipo de dados	Descrição da propriedade
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"	*flag*	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"	*string*	Especifica a consulta SQL customizada utilizada para o índice especificado. Veja os exemplos após a tabela a seguir.
indexes.INDEXNAME.remove	*flag*	Se True, remove o índice especificado do conjunto de índices.
table_space	*sequência*	Especifica o espaço de tabelas que será criado.
use_partition	*flag*	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 a `CREATE TABLE MYTABLE (...) COMPRESS YES;` em SQL). As propriedades `use_compression` e `compression_mode` são fornecidas para suportar esse recurso, conforme a seguir.

Tabela 228. Propriedades de databaseexportnode usando recursos de compactação.

Propriedades de databaseexportnode	Tipo de dados	Descrição da propriedade
use_compression	*Boolean*	Se configurado para True, cria tabelas para exportação com compactação.
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 IBM SPSS Data Collection gera dados no formato utilizado pelo software de pesquisa de mercado do IBM SPSS Data Collection. O IBM SPSS Data Collection Data Library deve ser instalado para usar 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 229. Propriedades de datacollectionexportnode

Propriedades de **datacollectionexportnode**	Tipo de dados	Descrição da propriedade
metadata_file	*string*	O nome do arquivo de metadados para exportar.
merge_metadata	Substituir MergeCurrent	
enable_system_variables	*flag*	Especifica se o arquivo *.mdd* exportado deve incluir as variáveis do sistema IBM SPSS Data Collection.
casedata_file	*string*	O nome do arquivo *.sav* para o qual os dados do caso são exportados.
generate_import	*flag*	

Propriedades de excelexportnode

O nó de exportação do Excel gera dados no formato de arquivo .xlsx do Microsoft Excel. 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 230. Propriedades de excelexportnode

Propriedades de **excelexportnode**	Tipo de dados	Descrição da propriedade
full_filename	*string*	
excel_file_type	Excel2007	
export_mode	Create Anexar	
inc_field_names	*flag*	Especifica se nomes de campos devem ser incluídos na primeira linha da planilha.
start_cell	*string*	Especifica célula inicial para exportação.
worksheet_name	*string*	O nome da planilha a ser gravada.
launch_application	*flag*	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	*flag*	Especifica se um nó de importação do Excel deve ser gerado que lerá o arquivo de dados exportado.

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 outro software de análise ou 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 231. Propriedades de outputfilenode

Propriedades de **outputfilenode**	Tipo de dados	Descrição da propriedade
full_filename	*string*	Nome do arquivo de saída.

Tabela 231. Propriedades de outputfilenode (continuação)

Propriedades de **outputfilenode**	Tipo de dados	Descrição da propriedade
write_mode	Substituir Anexar	
inc_field_names	*flag*	
use_newline_after_records	*flag*	
delimit_mode	Vírgula Tab Espaço Outro	
other_delimiter	*char*	
quote_mode	None Single Duplo Outro	
other_quote	*flag*	
generate_import	*flag*	
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 os 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 232. Propriedades de sasexportnode

Propriedades de **sasexportnode**	Tipo de dados	Descrição da propriedade
formato	Windows UNIX SAS7 SAS8	Campos de rótulo de propriedade variante
full_filename	*string*	
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	*flag*	

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 utilizado para arquivos em cache no IBM SPSS Modeler.

As propriedades desse nó são descritas em "Propriedades de statisticsexportnode" na página 301.

Propriedades do Nó tm1export

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 233. Propriedades do nó tm1export.

Propriedades do nó `tm1export`	Tipo de dados	Descrição da propriedade
`pm_host`	*string*	O nome do host. Por exemplo: `TM1_export.setPropertyValue("pm_host", 'http://9.191.86.82:9510/pmhub/pm')`
`tm1_connection`	*["field","field", ... ,"field"]*	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")`
`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: `[["param1", "value"],...,["paramN","value"]]` Por exemplo: `TM1_export.setPropertyValue("spssfield_ tm1element_mapping", [["plan_version","plan_version"], ["plan_department","plan_department"]])`

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 234. Propriedades de xmlexportnode

Propriedades de xmlexportnode	**Tipo de dados**	**Descrição da propriedade**
full_filename	*string*	(necessário) Caminho e o nome do arquivo completos do arquivo de exportação XML.
use_xml_schema	*flag*	Especifica se deseja usar um esquema XML (arquivo XSD ou DTD) para controlar a estrutura dos dados exportados.
full_schema_filename	*string*	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	*flag*	Gera um nó de origem XML que lerá o arquivo de dados exportado de volta no fluxo.
records	*string*	Expressão XPath que indica o limite do registro.
map	*string*	Mapeia o nome do campo para estrutura XML.

Capítulo 18. Propriedades do Nó do IBM SPSS Statistics

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 235. Propriedades de statisticsimportnode.

Propriedades de 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`	*sinalizador*	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. Esse nó requer uma cópia licenciada do 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 236. Propriedades de statisticstransformnode

Propriedades de statisticstransformnode	Tipo de dados	Descrição da propriedade
syntax	*string*	
check_before_saving	*flag*	Valida a sintaxe inserida antes de salvar as entradas. Exibirá uma mensagem de erro se a sintaxe for inválida.
default_include	*flag*	Consulte o tópico "Propriedades de filternode" na página 126 para obter mais informações.
include	*flag*	Consulte o tópico "Propriedades de filternode" na página 126 para obter mais informações.
new_name	*string*	Consulte o tópico "Propriedades de filternode" na página 126 para obter mais informações.

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. Esse nó requer uma cópia licenciada do 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 de statisticsmodelnode	Tipo de dados	Descrição da propriedade
syntax	*string*	
default_include	*flag*	Consulte o tópico "Propriedades de filternode" na página 126 para obter mais informações.
include	*flag*	Consulte o tópico "Propriedades de filternode" na página 126 para obter mais informações.
new_name	*string*	Consulte o tópico "Propriedades de filternode" na página 126 para obter mais informações.

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. Esse nó requer uma cópia licenciada do 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 237. Propriedades de statisticsoutputnode

Propriedades de **statisticsoutputnode**	Tipo de dados	Descrição da propriedade
mode	Dialog Syntax	Seleciona a opção "Diálogo do IBM SPSS Statistics" ou o Editor de Sintaxe
syntax	*string*	
use_output_name	*flag*	
output_name	*string*	
output_mode	Screen File	
full_filename	*string*	
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 utilizado para arquivos em cache no 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 238. Propriedades de statisticsexportnode.

Propriedades de **statisticsexportnode**	Tipo de dados	Descrição da propriedade
full_filename	*string*	
file_type	sav zsav	Arquivo de salvamento no formato *sav* ou *zsav*. Por exemplo: `statisticsexportnode.setPropertyValue("file_type","sav")`
encrypt_file	*flag*	Se o arquivo deve ou não ser protegido por senha.
senha	*string*	A senha.
launch_application	*flag*	

Tabela 238. Propriedades de statisticsexportnode (continuação).

Propriedades de statisticsexportnode	**Tipo de dados**	**Descrição da propriedade**
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	*flag*	

Capítulo 19. Propriedades do SuperNode

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 239. Propriedades de supernó de terminal

Nome da propriedade	Tipo de Propriedade /Lista de valores	Descrição da propriedade
execute_method	Script Normal	
script	*string*	

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 SuperNodes

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. Consulte o tópico Propriedades do Nó de Nugget do Modelo para obter mais informações.

Nota: Sob circunstâncias normais, referenciar modelos por nome *e* também por tipo é recomendado para evitar confusão.

Tabela 240. Nomes do Nugget do Modelo (Paleta de Modelagem).

Nome do modelo	**Modelo**
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
discriminant	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
neuralnetwork	Rede neural
quest	QUEST

Tabela 240. Nomes do Nugget do Modelo (Paleta de Modelagem) (continuação).

Nome do modelo	Modelo
regression	Regressão linear
sequence	Sequência
slrm	Modelo de resposta de autoaprendizado
statisticsmodel	Modelo do IBM SPSS Statistics
svm	Support Vector Machine
timeseries	Séries temporais
twostep	TwoStep

Tabela 241. Nomes do Nugget do Modelo (Paleta de Modelagem de Banco de Dados).

Nome do modelo	Modelo
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 da 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 da 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
netezzaregtree	Árvore de regressão Netezza
netezzatimeseries	Série de tempo Netezza
oraabn	Oracle Adaptive Bayes
oraai	AI da Oracle
oradecisiontree	Árvore de Decisão da Oracle
oraglm	GLM da Oracle
orakmeans	*k*-Médias da Oracle

Tabela 241. Nomes do Nugget do Modelo (Paleta de Modelagem de Banco de Dados) (continuação).

Nome do modelo	Modelo
oranb	Naive Bayes da Oracle
oranmf	NMF da Oracle
oraocluster	Cluster-O da 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. A partir dos menus, escolha:
   Ferramentas > **Opções do Usuário**
2. Clique na guia **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. Para obter uma lista completa dos formatos de exportação disponíveis para cada tipo de objeto de saída, consulte a descrição das propriedades para o nó que cria o tipo de saída, disponível no Propriedades Comuns do Nó Gráfico e no Propriedades do Nó de Saída .

Tabela 242. 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
evaluationoutput	Avaliação
histogramoutput	Histograma
matrixoutput	Matriz
meansoutput	Médias
multiplotoutput	Multigráficos
plotoutput	Gráfico
qualityoutput	Qualidade

Tabela 242. Tipos de objetos de saída e os nós que criam esses objetos. (continuação).

Tipo de objeto de saída	**Nó**
reportdocumentoutput	Este tipo de objeto não é de um nó, é a saída criada por um relatório do projeto
reportoutput	Relatório
statisticsprocedureoutput	Saída do Statistics
statisticsoutput	Statistics
tableoutput	Tabela
timeplotoutput	Plotagem 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 243. Mapeamento de script legado para script Python para literais e comentários.

Script anterior	Script Python
Número Inteiro, por exemplo, `4`	Igual
Flutuante, por exemplo, `0,003`	Igual
Sequências entre aspas simples, por exemplo, `'Hello'`	Igual **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 entre aspas duplas, por exemplo, `"Hello again"`	Igual **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"""`	Igual
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 244. 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`

Tabela 244. Mapeamento de script legado para script Python para operadores (continuação).

Script anterior	Script Python
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
e ou not(EXPR)	e ou 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 245. 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: ...
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

Tabela 245. Mapeamento de script legado para script Python para condicionais e loop (continuação).

Script anterior	Script Python
`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 246. 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*`)`

Executando Loop

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 legado, é possível utilizar o comando salve, 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 247. 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*)
`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 248. Mapeamento de script legado para script Python para operações de modelo.

Script anterior	Script Python
`open model path`	*taskrunner*.openModelFromFile(*path*, *autoManage*)
`save model como path`	*taskrunner*.saveModelToFile(*model*, *path*)
`retrieve model path`	*repository*.retrieveModel(*path*, *version*, *label*, *autoManage*)
`store model como 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 249. 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 como path`	*taskrunner*.saveDocumentToFile(*output*, *path*)
`retrieve output path`	*repository*.retrieveDocument(*path*, *version*, *label*, *autoManage*)
`store output como 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 produtos e serviços oferecidos no mundo todo.

É possível que a IBM não ofereça os produtos, serviços ou recursos discutidos nesta publicação em outros países. Consulte um representante IBM local para obter informações sobre 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 usados. Qualquer produto, programa ou serviço funcionalmente equivalente, que não infrinja nenhum direito de propriedade intelectual da IBM poderá 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 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
Av. Pasteur, 138-146, Botafogo
Rio de Janeiro. RJ
CEP 22290-240
Brasil

Para pedidos de licença relacionados a informações 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:

Licença de Propriedade Intelectual
Lei de Propriedade Intelectual
IBM Japan Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

O parágrafo a seguir não se aplica a nenhum país em que tais disposições não estejam de acordo com a legislação local: 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 A ELAS NÃO SE LIMITANDO, AS GARANTIAS IMPLÍCITAS DE NÃO-INFRAÇÃO, COMERCIALIZAÇÃO OU ADEQUAÇÃO A UM DETERMINADO PROPÓSITO. Alguns países não permitem a exclusão de garantias expressas ou implícitas em determinadas 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.

As referências nestas informações a websites não IBM são fornecidas apenas por conveniência e não representam de forma alguma um endosso a estes websites. Os materiais contidos nesses Web sites não fazem parte dos materiais deste produto IBM e a utilização desses Web sites é de inteira responsabilidade do Cliente.

A IBM pode usar 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:

IBM Software Group
ATTN: Licensing
200 W. Madison St.
Chicago, IL; 60606
CEP 22290-240

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 IBM Customer Agreement, Contrato de Licença do Programa Internacional IBM ou qualquer outro contrato equivalente.

Quaisquer dados de desempenho aqui contidos foram determinados em um ambiente controlado. Portanto, os resultados obtidos em outros ambientes operacionais podem variar significativamente. Algumas medidas podem ter sido tomadas em sistemas em nível de desenvolvimento e não há garantia de que estas medidas serão as mesmas em sistemas geralmente disponíveis. Além disso, algumas medidas podem ter sido estimadas através de extrapolação. Os resultados reais podem variar. Os usuários desta publicação devem verificar os dados aplicáveis para seu ambiente específico.

As informações relacionadas a produtos não IBM foram obtidas junto aos fornecedores destes produtos, de seus anúncios publicados ou de outras fontes disponíveis publicamente. A IBM não testou estes produtos e não pode confirmar a precisão de desempenho, compatibilidade nem qualquer outra reivindicação relacionada a produtos não IBM. Perguntas sobre os recursos de produtos não IBM devem ser encaminhadas aos fornecedores desses produtos.

Todas as instruções relativas aos objetivos ou intenção futura da IBM estão sujeitas a alterações ou cancelamento sem aviso prévio e representam apenas metas e objetivos.

Essas informações contêm exemplos de dados e relatórios usados em operações diárias de negócios. Para ilustrá-los da forma mais completa possível, os exemplos incluem nomes de indivíduos, empresas, marcas e produtos. Todos estes nomes são fictícios e qualquer semelhança com nomes e endereços usados por uma empresa real é mera coincidência.

Se essas informações estiverem sendo exibidas em formato eletrônico, as fotografias e ilustrações coloridas poderão não aparecer.

Marcas comerciais

IBM, o logotipo IBM e ibm.com são marcas comerciais ou marcas registradas da International Business Machines Corp., registradas em muitas jurisdições em todo o mundo. Outros nomes de produtos e serviços podem ser marcas comerciais da IBM ou de outras empresas. Uma lista atual de marcas comerciais da IBM trademarks está disponível na web em "Informações de Copyright e marcas comerciais" em www.ibm.com/legal/copytrade.shtml.

Intel, logotipo Intel, Intel Inside, logotipo Intel Inside, Intel Centrino, logotipo Intel Centrino, Celeron, Intel Xeon, Intel SpeedStep, Itanium e Pentium são marcas registradas ou marcas comerciais da Intel Corporation ou suas subsidiárias nos Estados Unidos e 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 em outros países.

Java e todas as marcas e logotipos baseados em Java são marcas comerciais ou marcas registradas da Oracle e/ou suas afiliadas.

Outros nomes de produtos e serviços podem ser marcas comerciais da IBM ou de outras empresa.

Índice Remissivo

N

O

P

R

S

T

V

Impresso no Brasil