



Universidade do Minho
Escola de Ciências

UNIVERSIDADE DO MINHO
MESTRADO EM MATEMÁTICA E COMPUTAÇÃO

Sistemas Baseados em Similaridade

Ficha Prática Individual 5

Hugo Filipe de Sá Rocha (PG52250)

8 de novembro de 2023

Conteúdo

1	Contextualização	3
1.1	Aspetos a considerar	3
2	Concepção das soluções	4
2.1	Tarefa 1	4
2.2	Tarefa 2 (tratamento de dados)	7
2.2.1	Alínea a)	7
2.2.2	Alínea b)	9
2.2.3	Alínea c)	11
2.2.4	Alínea d)	13
2.2.5	Alínea e)	14
2.2.6	Alínea f)	16
2.3	Tarefa 3	17
2.3.1	Alínea a)	17
2.3.2	Alínea b)	18
2.3.3	Alínea c)	19
2.4	Tarefa 4 (<i>tuning</i> do modelo criado)	20
2.4.1	Alínea a)	20
2.4.2	Alínea b)	22
2.4.3	Alínea c)	23
2.4.4	Alínea d)	25
2.5	Tarefa 5	27
2.6	Tarefa 6	30
2.7	<i>Workflow</i> completo para o caso de uma <i>Random Forest</i> (com uso de meta-nodos)	31

Capítulo 1

Contextualização

Uma multinacional na área do retalho possui o histórico de vendas semanais de 17 das suas lojas em diferentes regiões do país, sendo que cada loja contém vários departamentos (desporto, cozinha, produtos alimentícios e higiene pessoal, entre outros). A empresa realiza também vários eventos promocionais ao longo do ano, normalmente precedendo feriados importantes. A empresa pretende agora extrair informação relevante dos *datasets* e desenvolver um modelo de *machine learning* que, com base num conjunto relevante de *features*, permita estimar as vendas mensais de cada uma das suas lojas. A empresa disponibiliza dois *datasets* para a concepção do modelo onde um deles contém informação sobre cada uma das lojas, incluindo o seu tipo e tamanho, enquanto que o outro contém dados referentes às vendas semanais de cada departamento de cada loja, a data e um *boolean* indicando se houve um feriado durante essa semana. Com base nisto, o modelo deve estimar as vendas mensais de um terceiro conjunto de dados desconhecido.

1.1 Aspetos a considerar

Com este enunciado prático, pretende-se que seja feito o *tuning* de modelos baseados em árvore, abordando parâmetros nominais e numéricos como a medida de qualidade, o método de *pruning* e o número mínimo de registos por nodo, entre outros.

Capítulo 2

Concepção das soluções

2.1 Tarefa 1

- Carregar, no *Knime*, os dois primeiros *datasets*, juntá-los e explorar os dados utilizando vistas gráficas que permitam perceber a análise efetuada;

Como é tradicional, para leitura dos dois *datasets* de treino utilizei o nodo *CSV Reader* e o nodo *Table Reader*, visto que um dos ficheiros está no formato *csv* e o outro no formato *table*. Após isso, utilizei o nodo *Joiner* para juntar os dados pelo atributo *Store*, fazendo um *Inner Join* entre as duas tabelas. Além disso, foram aplicados alguns nodos para exploração e análise dos dados, sendo eles: *Statistics*, *Data Explorer*, *Box Plot* e *Rank Correlation*.

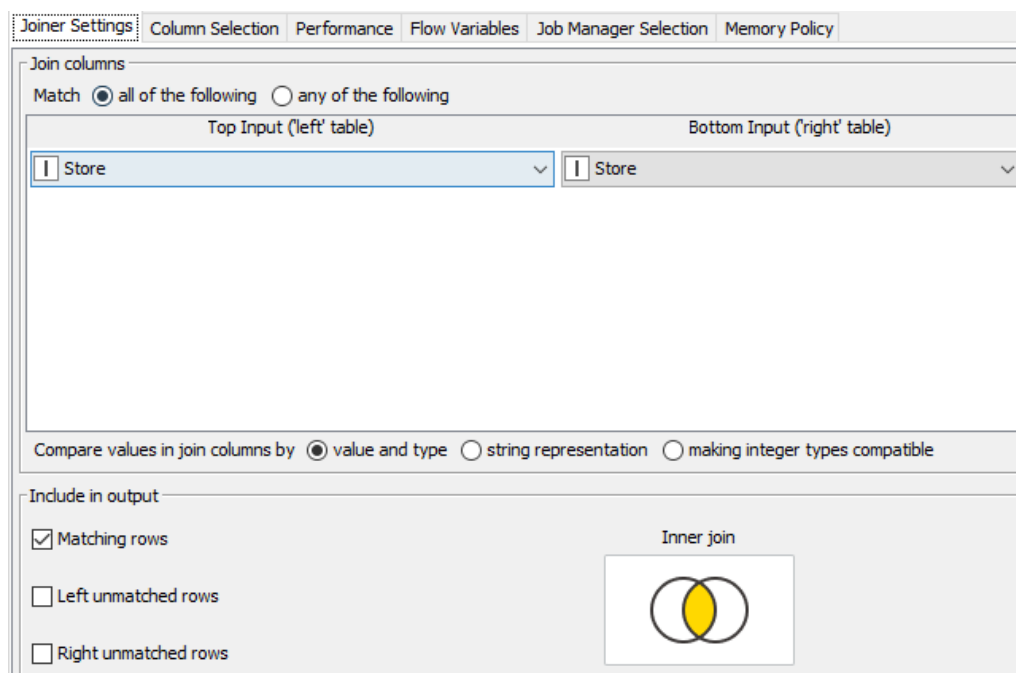
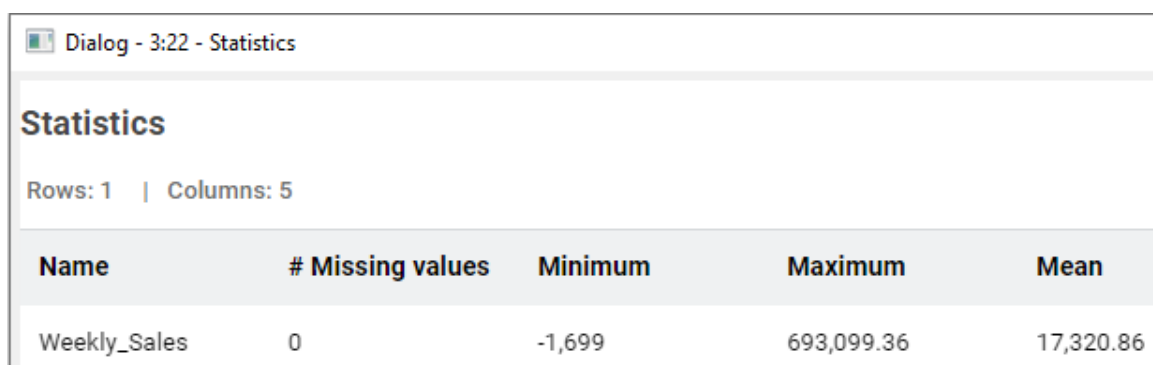


Figura 2.1: Configuração do nodo *Joiner*.

- No nodo **Statistics** podemos, por exemplo, analisar o *dataset* no que toca a *missing values* ou a estatísticas sobre os atributos. A título de exemplo, nas imagens abaixo podemos ver que o *dataset* **não possui *missing values*** e que, na análise ao atributo ***Weekly_Sales***, os valores variam entre o mínimo, **-1699**, e o máximo, **693099.36** obtendo-se uma média de **17320.86**.



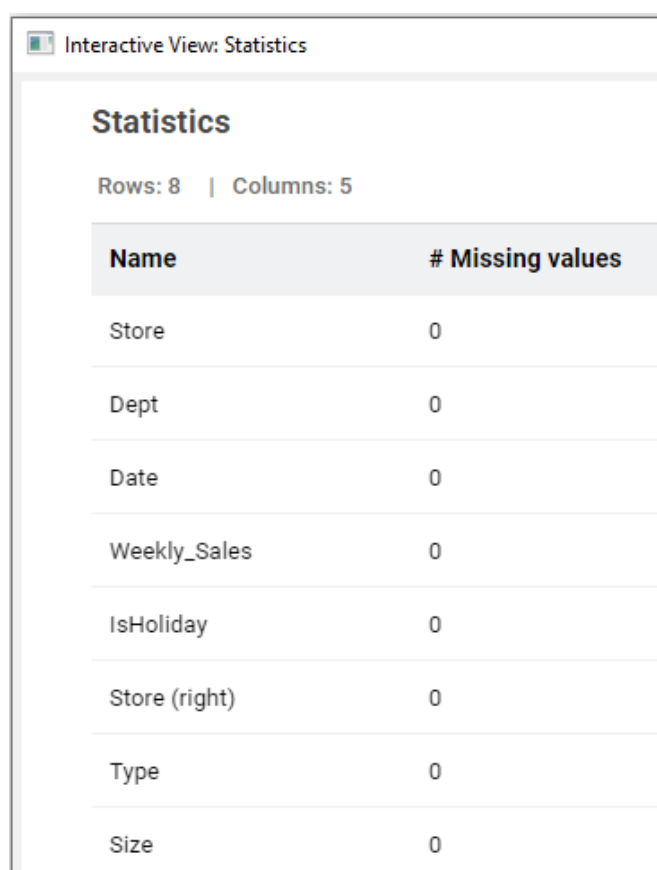
Dialog - 3:22 - Statistics

Statistics

Rows: 1 | Columns: 5

Name	# Missing values	Minimum	Maximum	Mean
Weekly_Sales	0	-1,699	693,099.36	17,320.86

Figura 2.2: Mínimo, máximo e média dos valores de *Weekly_Sales*.



Interactive View: Statistics

Statistics

Rows: 8 | Columns: 5

Name	# Missing values
Store	0
Dept	0
Date	0
Weekly_Sales	0
IsHoliday	0
Store (right)	0
Type	0
Size	0

Figura 2.3: Ausência de *missing values*.

- No nodo **Data Explorer**, assim como no nodo *Statistics*, podemos obter estatísticas sobre os atributos bem como o histograma para cada um deles. A título de exemplo, segue abaixo o histograma do atributo **isHoliday**, onde podemos observar que a esmagadora maioria dos dias não são feriados comparativamente aos que são, como era expectável. Além disso, observa-se também, no histograma do atributo **Type** que os valores **A** e **B** aparecem numa proporção semelhante.

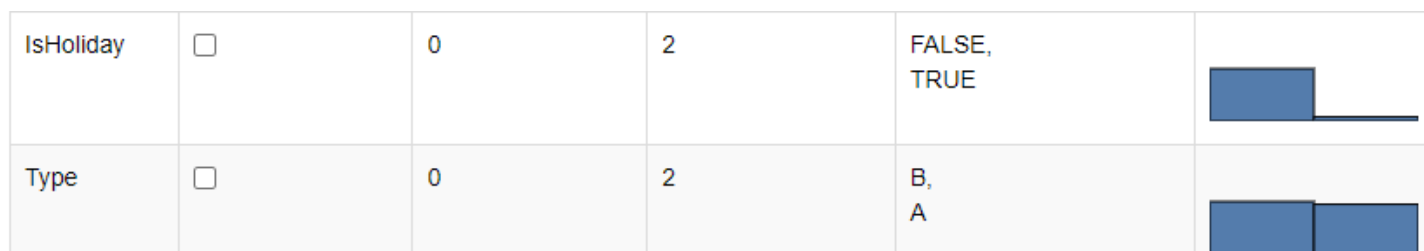


Figura 2.4: Histograma de *isHoliday* e de *Type*.

- No nodo **Box Plot**, o objetivo passou por procurar **outliers** nos dados. Na imagem abaixo, podemos observar dois ou três *outliers* relativos ao atributo **Weekly_Sales**, onde optei por não os remover do *dataset*.

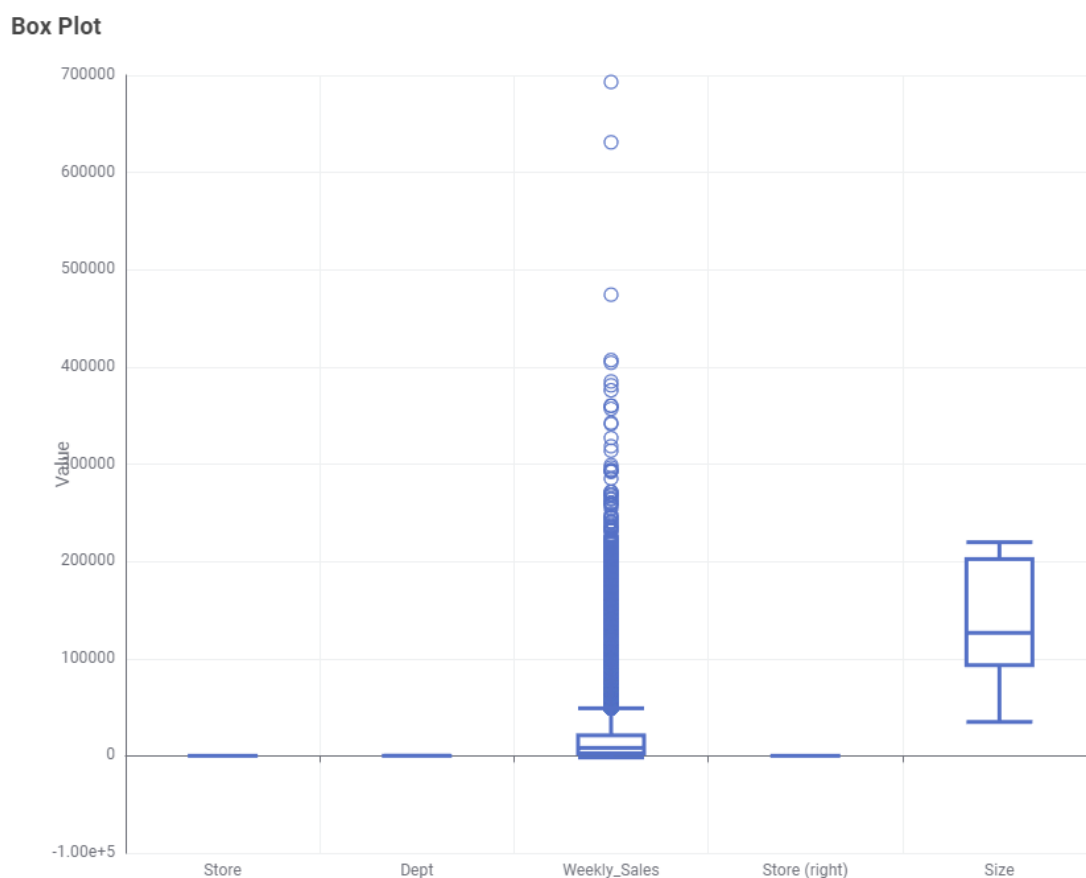


Figura 2.5: Detecção de *outliers*.

- No nodo **Rank Correlation**, procurou-se explorar a correlação entre todos os atributos (dois a dois) e percebe-se que os atributos **Type** e **Size** têm uma correlação negativa forte de **-0.8672**. Visto que foi feito um *Inner Join* entre as tabelas pelo atributo *Store*, como é de esperar, o atributo *Store* proveniente da primeira tabela tem uma correlação de 1 com o atributo *Store (right)* que é proveniente da segunda tabela.

	Store	Dept	Date	W...	Is...	St...	Type	Size
Store	corr = -1							
Dept		corr = +1						
Date			corr = n/a					
Weekly_Sales				corr = -1				
IsHoliday					corr = +1			
Store (right)						corr = -1		
Type							corr = +1	
Size								corr = -1

Figura 2.6: Correlação entre atributos.

2.2 Tarefa 2 (tratamento de dados)

2.2.1 Alínea a)

- Fazer label encoding à feature *isHoliday* (1 deve corresponder ao valor *True*);

Antes de mais, comecei por remover o atributo *Store (right)* visto ter os mesmos valores do atributo *Store*, através do nodo *Column Filter*. Para fazer *label encoding* à feature *isHoliday* utilizei dois nodos: **String Manipulation** e **String to Number**, onde no primeiro fiz uso da função **replace** de forma a passar as *strings* 'FALSE' e 'TRUE' para as *strings* '0' e '1', respetivamente. Já no nodo *String to Number*, o objetivo passou por associar as *strings* '0' e '1' aos inteiros 0 e 1, respetivamente.

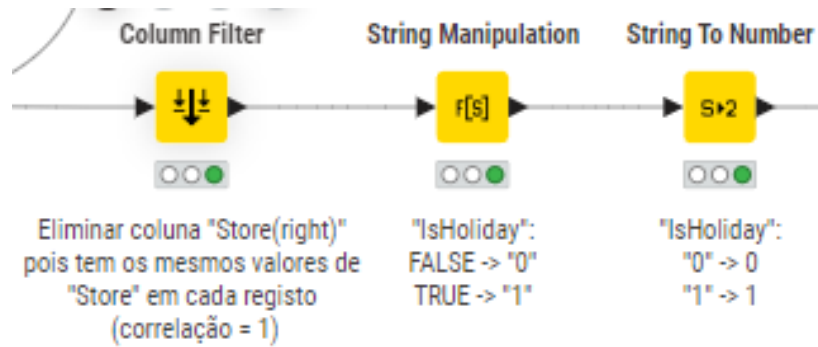


Figura 2.7: *Workflow* relativo à alínea a).

Dialog - 3:5 - String Manipulation ("IsHoliday":)

File

String Manipulation		Flow Variables	Job Manager Selection	Memory Policy
Column List ROWID ROWINDEX ROWCOUNT I Store I Dept S Date D Weekly_Sales S IsHoliday S Type I Size	Category Replace	Description Replaces all occurrences of a String within another String. Examples: <code>replace("abcabc", "ab", "") = "cc"</code> <code>replace("abcabc", "ab", "z") = "zczc"</code> <code>replace(null, *, *) = null</code> <code>replace("", *, *) = ""</code> <code>replace("any", null, *) = "any"</code> <code>replace("any", *, null) = "any"</code> <code>replace("any", "", *) = "any"</code> * can be any character sequence.		
Flow Variable List s knime.workspace	Function replace(str, search, replace) replace(str, search, replace, modifiers) replaceChars(str, chars, replace) replaceChars(str, chars, replace, modifiers) replaceUmlauts(str, omitE) toEmpty(str...) toNull(str...) urlDecode(str) urlEncode(str)			
Expression 1 <code>replace(replace(\$IsHoliday\$, "FALSE", "0"), "TRUE", "1")</code>				
<input type="radio"/> Append Column:		<input type="checkbox"/> Insert Missing As Null		
<input checked="" type="radio"/> Replace Column: S IsHoliday		<input checked="" type="checkbox"/> Syntax check on close		

Figura 2.8: Configuração do nodo *String Manipulation*.

Dialog - 3:6 - String To Number ("IsHoliday":)

Column Selection

Column selection

Manual Wildcard Regex

Search

Excludes

Date
Type

Any unknown columns

Includes

IsHoliday

Parsing options

Decimal separator

Thousands separator

Type

Number (Integer)

Figura 2.9: Configuração do nodo *String to Number*.

2.2.2 Alínea b)

- Adicionar, a cada registo, as *features* ano e mês;

Para este exercício utilizei dois nodos: *String to DateTime* e *Extract DateTime fields*. O primeiro serviu para passar o atributo *Date* de *String* para *Date*, passando a estar no formato YYYY-MM-DD. Após isso, o nodo *Extract DateTime fields* serviu para extrair o ano e o mês do atributo *Date*, criando dois novos atributos: *Year* e *Month (number)*.

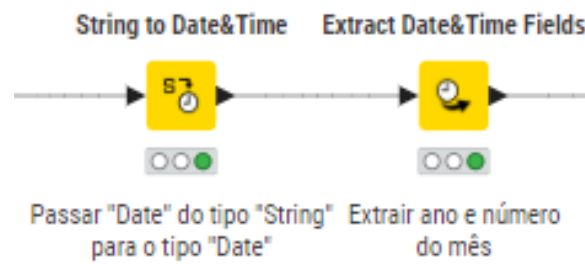


Figura 2.10: *Workflow* relativo à alínea b).

Dialog - 3:7 - String to Date&Time (Passar "Date" do tipo "String")

File

Options Flow Variables Job Manager Selection Memory Policy

☒ Manual Selection ☐ Wildcard/Regex Selection

Exclude

Filter

S Type

☐ Enforce exclusion

Include

Filter

S Date

☒ Enforce inclusion

> >> < <<

Replace/Append Selection

☐ Append selected columns Suffix of appended columns: (Date&Time)

☒ Replace selected columns

Type and Format Selection

New type: Date Date format: dd/MM/yyyy

Locale: pt-PT Content of the first cell: 05/02/2010

Guess data type and format

Figura 2.11: Configuração do nodo *String to DateTime*.

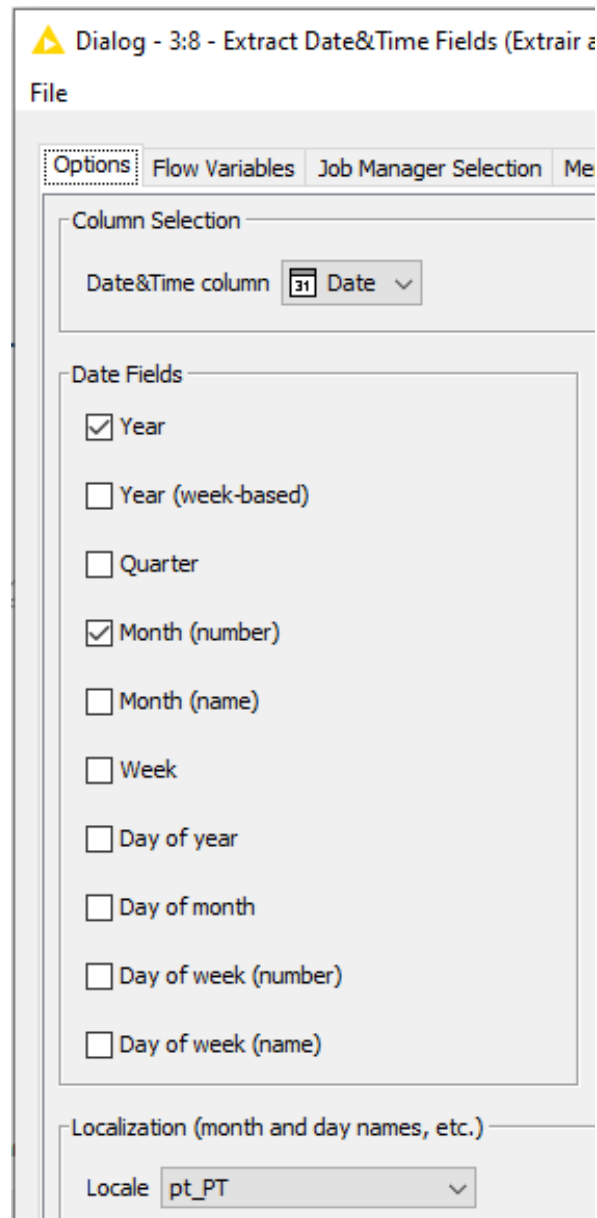


Figura 2.12: Configuração do nodo *Extract DateTime Fields*.

2.2.3 Alínea c)

- Agrupar os registos por loja, tipo, tamanho, ano e mês, agregando de forma a obter o somatório das vendas semanais de cada loja e a indicação da existência de feriados nesse mês;

Para este exercício, fiz uso do nodo **GroupBy** configurado de forma a que os dados sejam agrupados pelas *features* **Store**, **Type**, **Size**, **Year** e **Month (number)** no sentido de obter o somatório das vendas semanais de cada loja e a indicação da existência de feriados nesse mês.

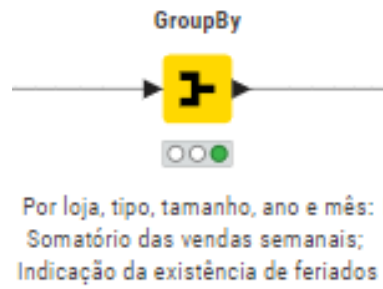


Figura 2.13: Nodo *GroupBy*.

Group settings

Available column(s)

Filter

I	Dept
31	Date
D	Weekly_Sales
I	IsHoliday

>

>>

<

<<

Group column(s)

Filter

I	Store
S	Type
I	Size
I	Year
I	Month (number)

Figura 2.14: Configuração do nodo *GroupBy*.

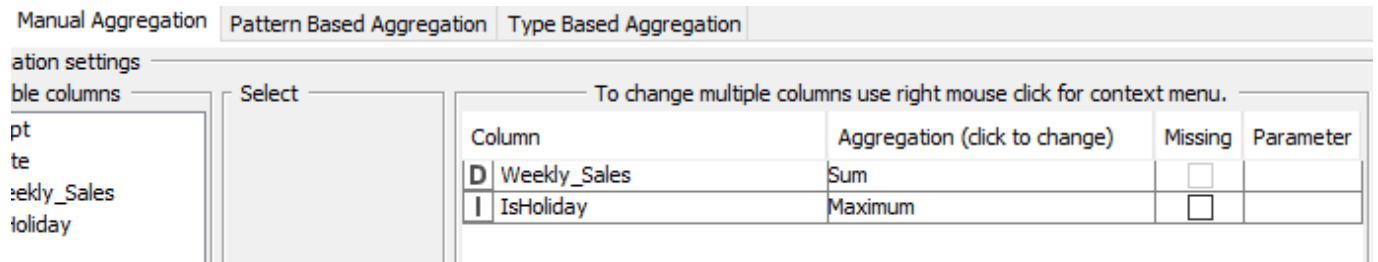


Figura 2.15: Configuração do nodo *GroupBy*.

2.2.4 Alínea d)

- Normalizar o somatório das vendas semanais utilizando a transformação linear *Min-Max* entre 0 e 1;

Para normalizar o somatório das vendas semanais utilizando a transformação linear *Min-Max* entre 0 e 1, bastou usar o nodo *Normalizer* para o atributo *Sum(Weekly_Sales)* com a transformação linear pedida.

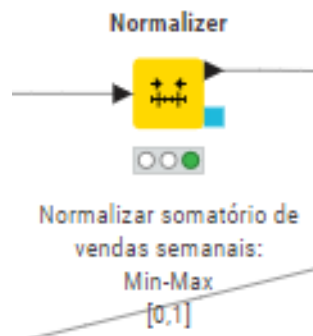


Figura 2.16: Nodo *Normalizer*.

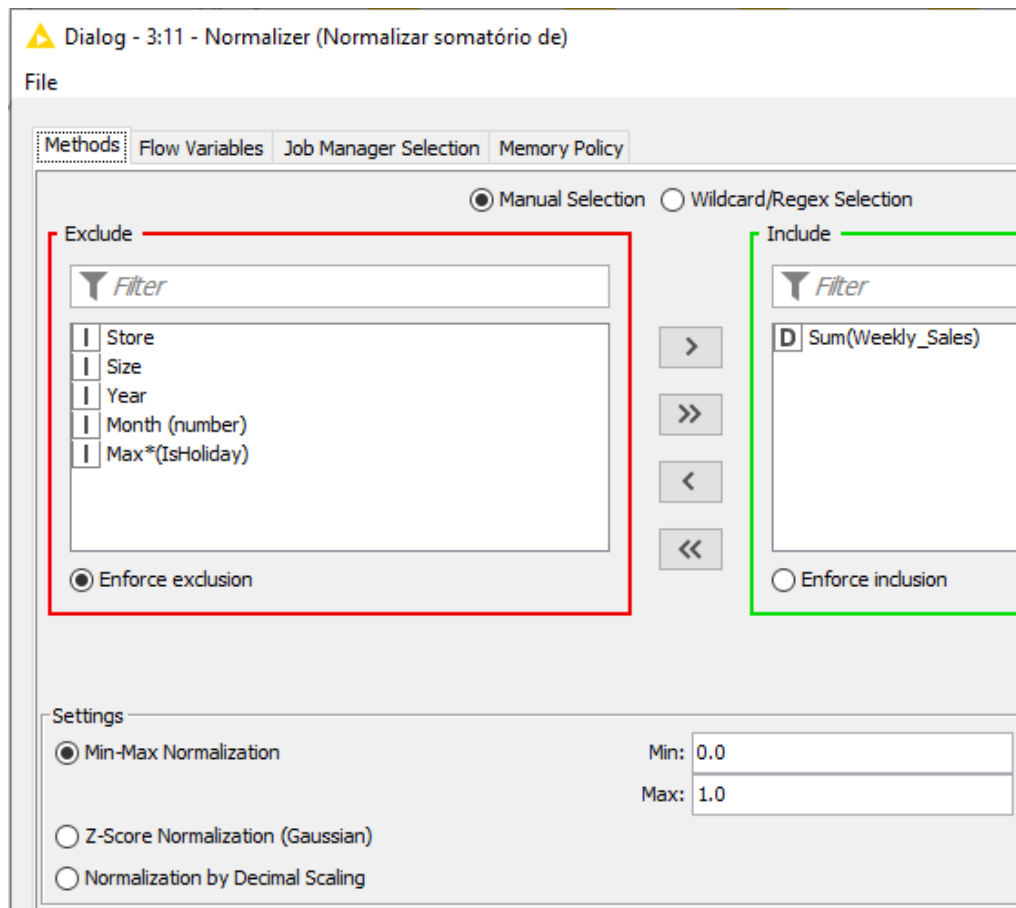


Figura 2.17: Configuração do nodo *Normalizer*.

2.2.5 Alínea e)

- Criar 4 *bins* de igual frequência sobre o valor normalizado no passo anterior (ligando a opção *replace target column(s)*);

Para criar 4 *bins* de igual frequência sobre o atributo *Sum(Weekly_Sales)*, utilizei o nodo *Auto-Binner* configurado como pedido no enunciado, nomeando também os *bins* como *Bin 1*, *Bin 2*, *Bin 3* e *Bin 4*.

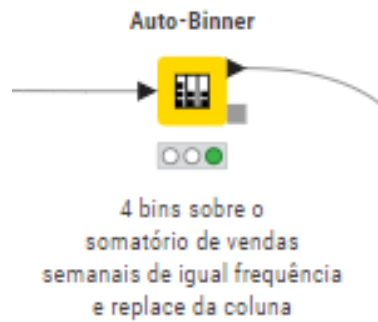


Figura 2.18: Nodo *Auto-Binner*.

Dialog - 3:12 - Auto-Binner (4 bins sobre o)

File

Auto Binner Settings Number Format Settings Flow Variables Job Manager Selection Memory Policy

☒ Manual Selection ☐ Wildcard/Regex Selection

Exclude

Filter

- Store
- Size
- Year
- Month (number)
- Max*(IsHoliday)

☒ Enforce exclusion

Include

Filter

- Sum(Weekly_Sales)

☐ Enforce inclusion

> >> < <<

Binning Method

☒ Fixed number of bins

Number of bins: 4

Equal: frequency

☐ Sample quantiles

Quantiles (comma separated): 0.0, 0.25, 0.5, 0.75, 1.0

Bin Naming

☒ Numbered e.g.: Bin 1, Bin 2, Bin 3

☐ Borders e.g.: [-10,0], (0,10], (10,20]

☐ Midpoints e.g.: -5, 5, 15

☐ Force integer bounds

☒ Replace target column(s)

Figura 2.19: Configuração do nodo *Auto-Binner*.

2.2.6 Alínea f)

- Renomear cada *bin* de forma a que o primeiro corresponda a *Low*, o segundo a *Medium*, o terceiro a *High* e o quarto a *Very High*.

Para renomear cada um dos *bins* em *Low*, *Medium*, *High* e *Very High* pela respetiva ordem, utilizei o nodo *String Manipulation* utilizando a função *replace* de forma recursiva para o efeito.

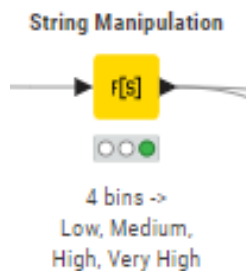


Figura 2.20: Nodo *String Manipulation*.

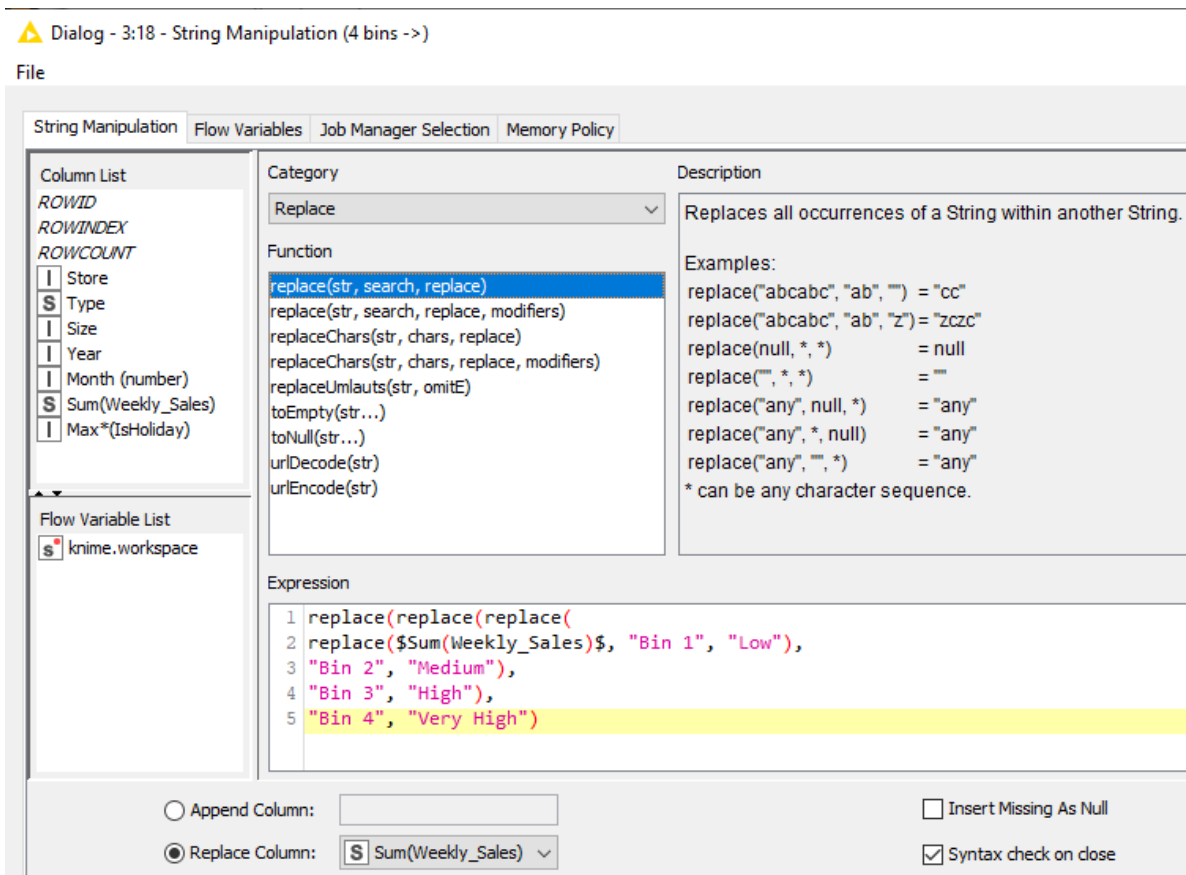


Figura 2.21: Configuração do nodo *String Manipulation*.

2.3 Tarefa 3

2.3.1 Alínea a)

- Treinar uma árvore de decisão;

Para treinar uma árvore de decisão utilizei o nodo *Decision Tree Learner* sobre os dados já tratados na tarefa anterior. Os parâmetros da atual árvore estão visíveis na imagem abaixo.

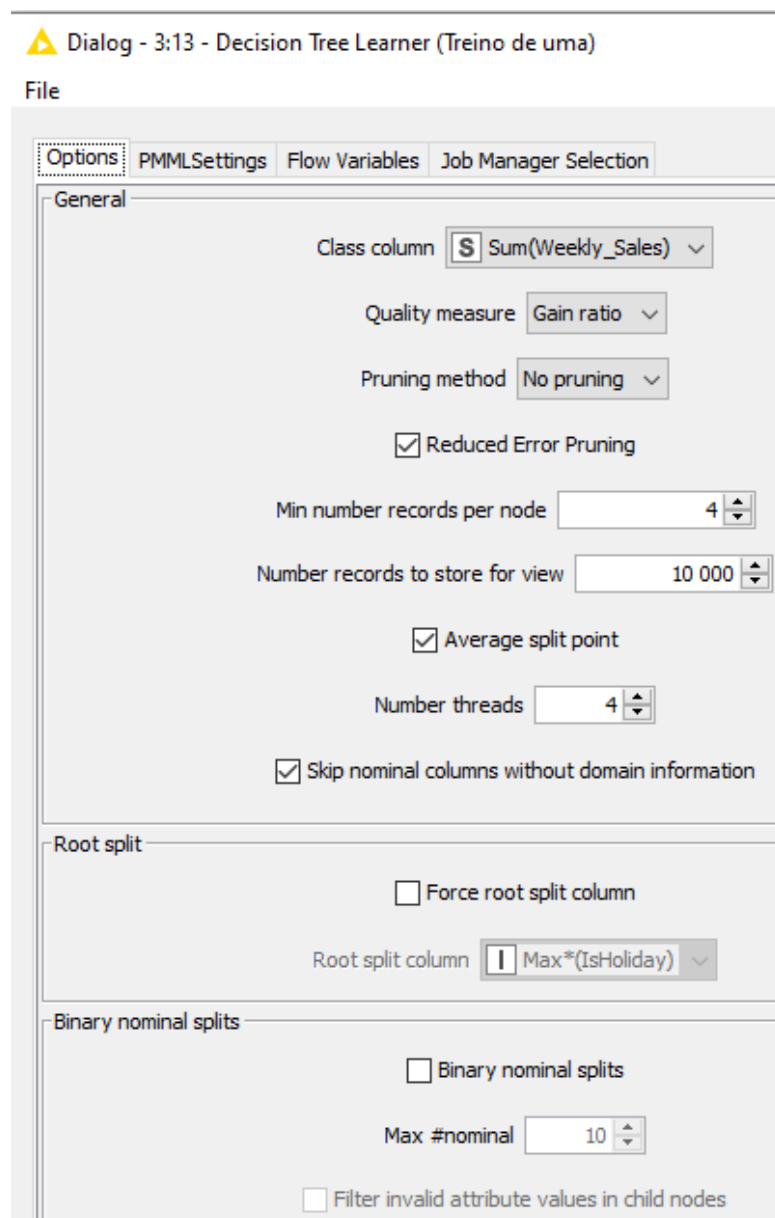


Figura 2.22: Configuração do nodo *Decision Tree Learner*.



Figura 2.23: Nodo *Decision Tree Learner*.

2.3.2 Alínea b)

- Carregar o *dataset* de teste e prever o valor de vendas de cada mês para cada uma das 17 lojas;

Neste exercício, como pedido, comecei por ler o *dataset* de teste através do nodo *CSV Reader*. Para prever o valor de vendas de cada mês para cada uma das 17 lojas, apliquei o nodo *Decision Tree Predictor* sobre os dados de teste com base no modelo treinado anteriormente.

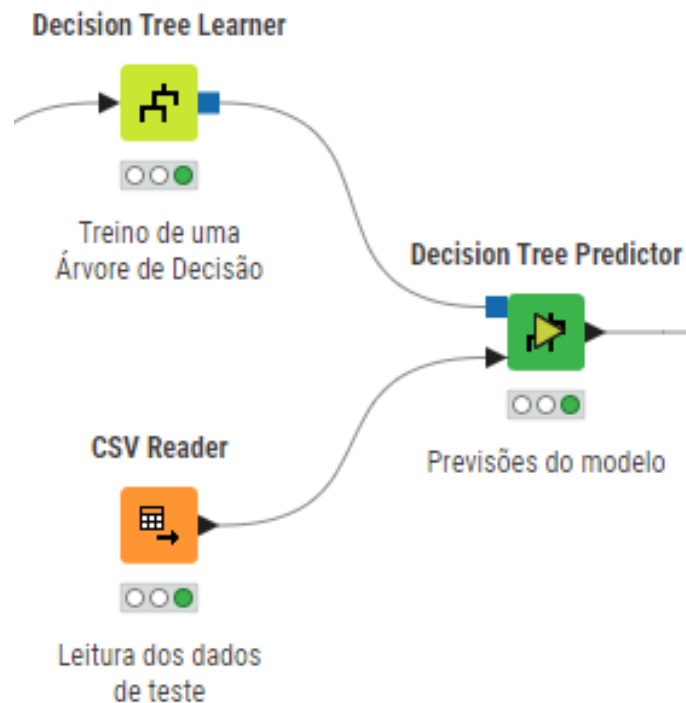


Figura 2.24: *Workflow* relativo à alínea b).

Month (number) <i>Number (integer)</i>	Sum(Weekly_Sales) <i>String</i>	Prediction (Sum(Weekly_Sales)) <i>String</i>
6	Very High	High
7	High	High
8	Very High	High
9	High	High
10	High	High

Figura 2.25: Previsões relativas à loja *Store = 1* (para o ano de 2012).

2.3.3 Alínea c)

- Mostrar, graficamente, uma tabela com a matriz de confusão do modelo.

Para obter a matriz de confusão do modelo, fiz uso do nodo **Scorer** configurado de tal forma que a primeira coluna corresponde aos valores reais das vendas de cada loja em cada mês e a segunda coluna às previsões desses mesmos valores.

Confusion Matrix - 3:21 - Scorer (Avaliação do mod...)				
File Hilite				
Sum(Weekl...	Very High	High	Low	Medium
Very High	14	6	0	0
High	7	13	0	4
Low	0	0	13	7
Medium	0	0	3	18
<div> <div>Correct classified: 58</div> <div>Wrong classified: 27</div> <div>Accuracy: 68,235%</div> <div>Error: 31,765%</div> <div>Cohen's kappa (κ): 0,577%</div> </div>				

Figura 2.26: Matriz de confusão do modelo.

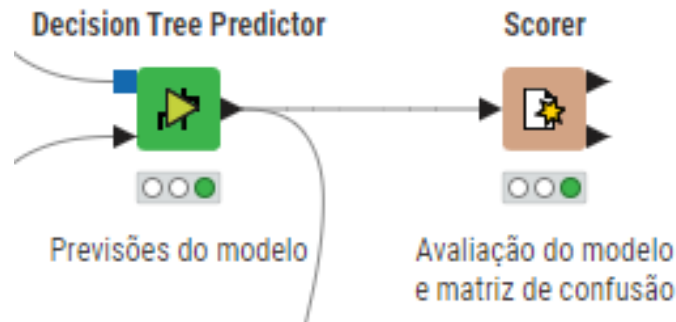


Figura 2.27: *Workflow* relativo à alínea c).

2.4 Tarefa 4 (*tuning* do modelo criado)

2.4.1 Alínea a)

- Experimentar todos os valores, entre 2 e 10, para o número mínimo de registros por nodo;

Neste exercício utilizei dois nodos, sendo eles: *Parameter Optimization Loop Start* e *Parameter Optimization Loop End*, onde, no primeiro, criei uma variável de fluxo *minRegistos* que vai percorrendo os valores inteiros de 2 a 10 e que está associada ao parâmetro *minNumberRecordsPerNode* do *Decision Tree Learner*. No final do *loop*, é possível ver a *accuracy* registada para cada valor do parâmetro, com os valores 3, 4, 5 e 6 a registarem a maior *accuracy*.

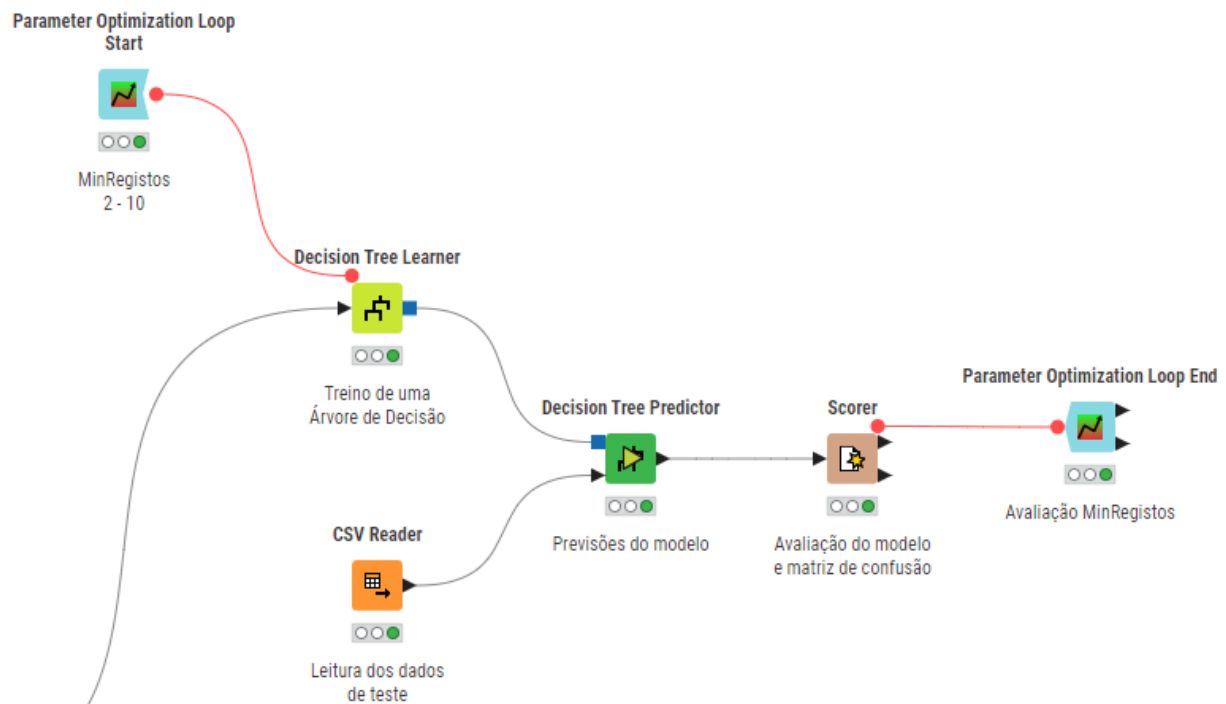


Figura 2.28: *Workflow* relativo à alínea a).

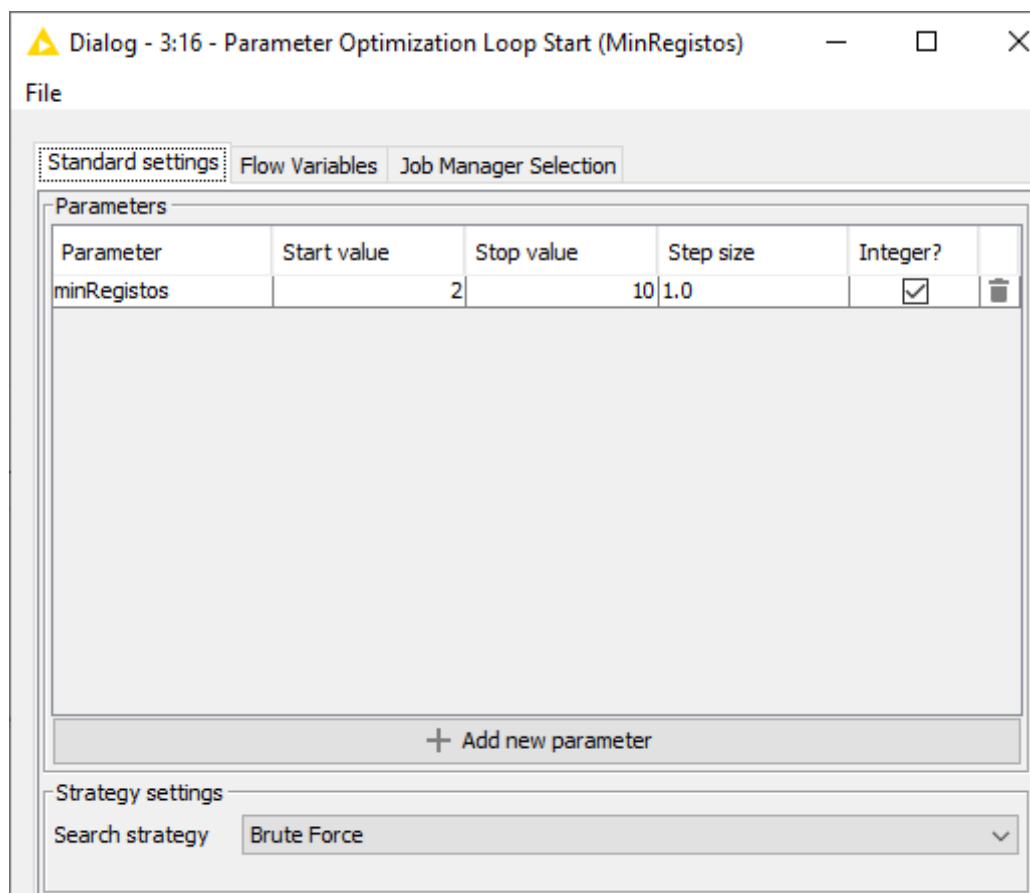


Figura 2.29: Configuração do nodo *Parameter Optimization Loop Start*.

Row...	minRegistos <i>Number (integer)</i>	Objective value <i>Number (double)</i>
Row1	3	0.682
Row2	4	0.682
Row3	5	0.682
Row4	6	0.682
Row5	7	0.671
Row6	8	0.671
Row8	10	0.659
Row0	2	0.647
Row7	9	0.647

Figura 2.30: *Accuracy* para cada valor do parâmetro.

2.4.2 Alínea b)

- Experimentar todas as possibilidades para a medida de qualidade;

Para a realização deste exercício decidi manter o valor do parâmetro *minNumberRecordsPerNode* a 4 visto ter sido um dos valores com maior *accuracy* na alínea anterior. Após isso, utilizei o nodo **Table Creator** para criar uma tabela que associa à *string qualityMeasure* os valores **Gain Ratio** e **Gini Index**. Depois, utilizei o nodo **Table Row to Variable Loop Start** para criar a variável de fluxo com base na tabela definida anteriormente e associei-a ao parâmetro *splitQualityMeasure* no *Decision Tree Learner*. No final do *loop*, é possível ver a *accuracy* registada para cada valor do parâmetro, com ambos os valores a registarem a mesma *accuracy*.

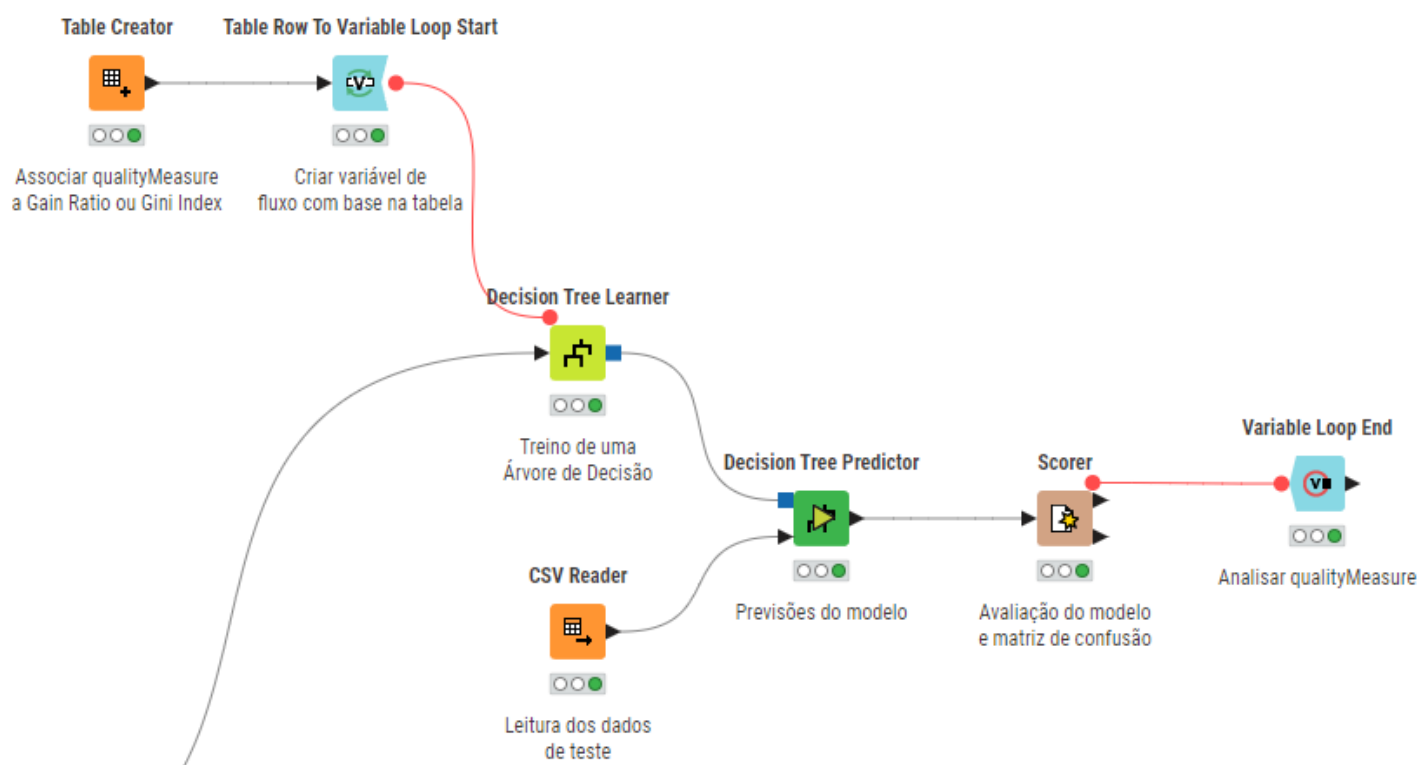


Figura 2.31: *Workflow* relativo à alínea b).

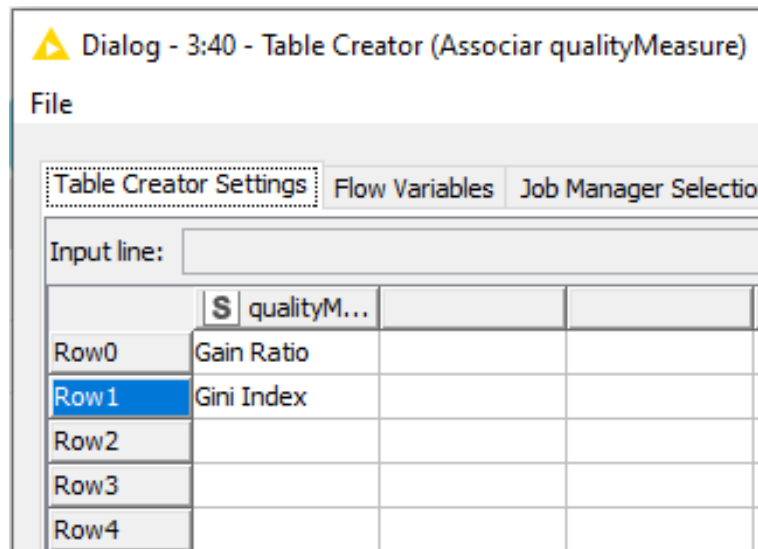


Figura 2.32: Configuração do nodo *Table Creator*.

Accuracy Number (double)	qualityMeasure String
0.682	Gain Ratio
0.682	Gini Index

Figura 2.33: *Accuracy* para cada valor do parâmetro.

2.4.3 Alínea c)

- Experimentar todas as possibilidades para o método de *pruning*;

Para a realização deste exercício decidi manter o valor do parâmetro *minNumberRecordsPerNode* a 4 e a medida de qualidade como *Gain Ratio*, visto serem alguns dos parâmetros que registaram maior *accuracy* nas alíneas anteriores. Após isso, utilizei o nodo **Table Creator** para criar uma tabela que associa à *string pruningMethod* os valores **No pruning** e **MDL**. Depois, utilizei o nodo **Table Row to Variable Loop Start** para criar a variável de fluxo com base na tabela definida anteriormente e associei-a ao parâmetro *pruningMethod* no *Decision Tree Learner*. No final do *loop*, é possível ver a *accuracy* registada para cada valor do parâmetro, com o valor **MDL** a registar uma *accuracy* superior.

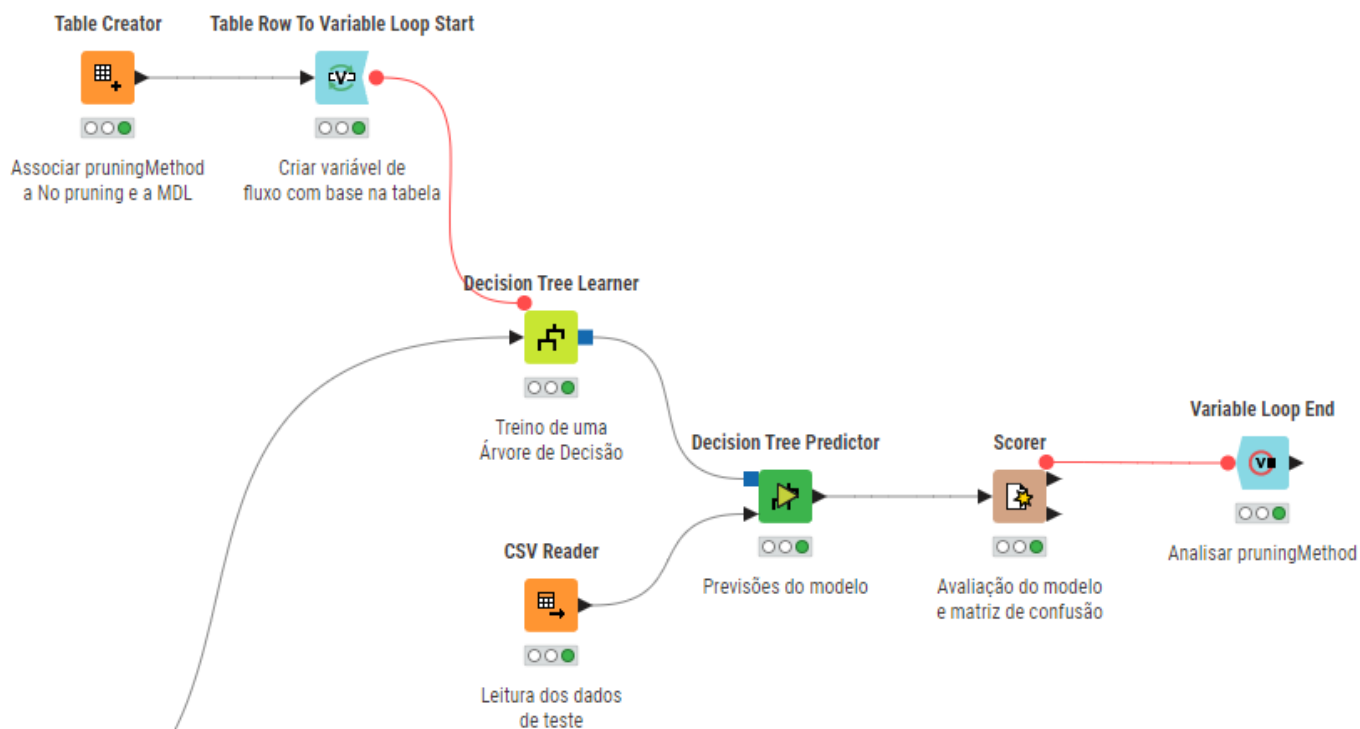


Figura 2.34: *Workflow* relativo à alínea c).

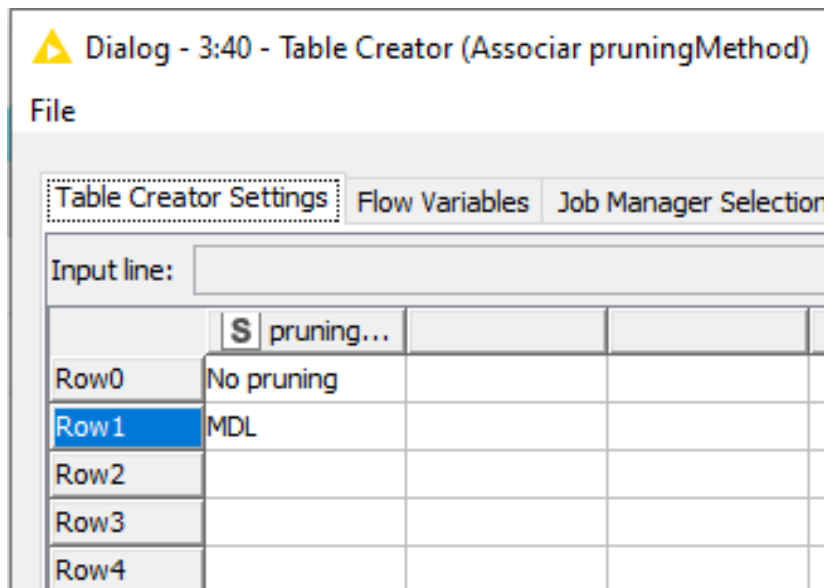


Figura 2.35: Configuração do nodo *Table Creator*.

Accuracy ↓ Number (double)	pruningMethod String
0.706	MDL
0.682	No pruning

Figura 2.36: *Accuracy* para cada valor do parâmetro.

2.4.4 Alínea d)

- Fazer o *tuning* dos parâmetros anteriores num único *workflow*. Guardar e analisar todos os resultados obtidos para cada combinação de hiper-parâmetros. Qual a combinação que oferece melhor performance? Existem grandes discrepâncias?

Para fazer o *tuning* completo dos parâmetros anteriores num único *workflow*, utilizei o nodo **Table Creator** onde expus em cada linha da tabela uma combinação possível dos parâmetros, onde, no final, obtive todas as 36 combinações possíveis dos 3 parâmetros anteriores. De lembrar que isto apenas é possível, sem grande trabalho, visto que não temos um número muito grande de combinações possíveis, caso contrário, seria boa ideia utilizar antes *loops* de *loops*. Após isso, criei 3 variáveis de fluxo com base nessa tabela que vão alterando o seu valor a cada iteração do *loop* e associei-as aos respectivos parâmetros no *Decision Tree Learner*. No final, é possível observar a *accuracy* de todas as combinações de hiper-parâmetros, onde, neste caso, o maior valor para a *accuracy* foi **0.706** que aparece em 6 combinações diferentes, todas com um denominador comum: método de *pruning* **MDL** e número mínimo de registos por nodo entre **2** e **4**. A combinação com pior *accuracy* registou um valor **0.647**, cerca de 6 pontos percentuais abaixo da combinação com melhor *accuracy*, havendo uma pequena variação mas não extremamente grande.

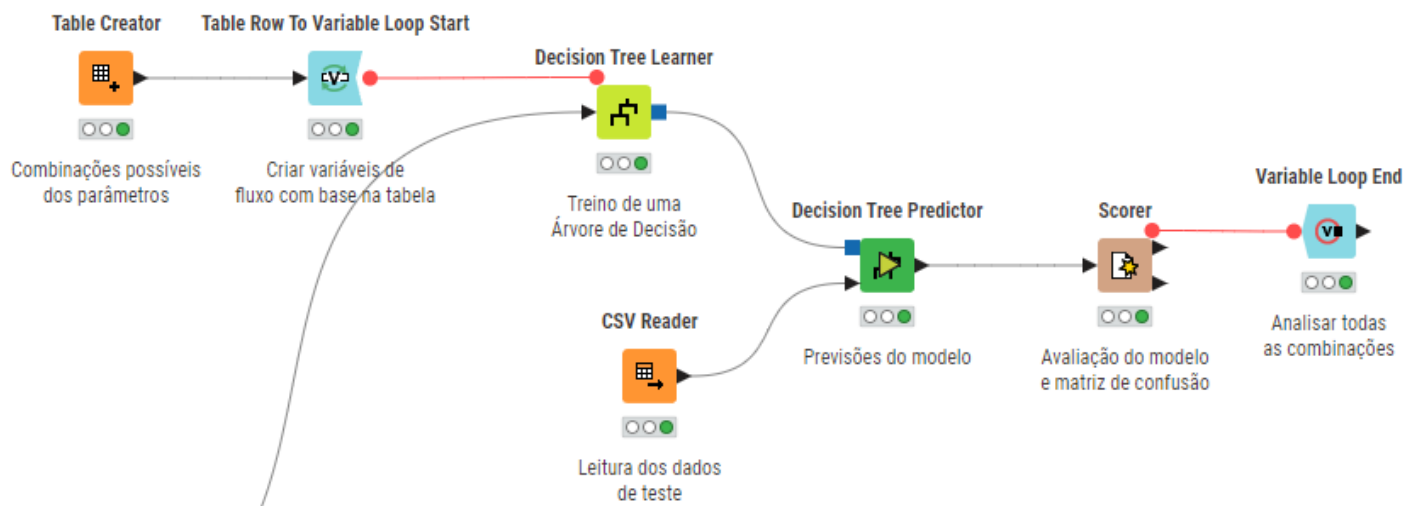


Figura 2.37: *Workflow* relativo à alínea d).

	I	minRegis...	S	qualityM...	S	pruning...
Row0		2		Gain Ratio		No pruning
Row1		2		Gain Ratio		MDL
Row2		2		Gini Index		No pruning
Row3		2		Gini Index		MDL
Row4		3		Gain Ratio		No pruning
Row5		3		Gain Ratio		MDL
Row6		3		Gini Index		No pruning
Row7		3		Gini Index		MDL
Row8		4		Gain Ratio		No pruning
Row9		4		Gain Ratio		MDL
Row10		4		Gini Index		No pruning
Row11		4		Gini Index		MDL
Row12		5		Gain Ratio		No pruning
Row13		5		Gain Ratio		MDL
Row14		5		Gini Index		No pruning
Row15		5		Gini Index		MDL
Row16		6		Gain Ratio		No pruning
Row17		6		Gain Ratio		MDL
Row18		6		Gini Index		No pruning
Row19		6		Gini Index		MDL
Row20		7		Gain Ratio		No pruning
Row21		7		Gain Ratio		MDL
Row22		7		Gini Index		No pruning

Figura 2.38: Algumas das combinações no nodo *Table Creator*.

Accuracy ↓ Number (double)	minRegistos Number (integer)	qualityMeasure String	pruningMethod String
0.706	2	Gain Ratio	MDL
0.706	2	Gini Index	MDL
0.706	3	Gain Ratio	MDL
0.706	3	Gini Index	MDL
0.706	4	Gain Ratio	MDL
0.706	4	Gini Index	MDL
0.682	3	Gain Ratio	No pruning

Figura 2.39: Combinações com melhor *accuracy*.

2.5 Tarefa 5

- Treinar e fazer o *tuning* de uma *Random Forest*. Guardar e analisar todos os resultados obtidos para cada combinação de hiper-parâmetros;

No que diz respeito a treinar uma *Random Forest*, utilizei o nodo *Random Forest Learner* nos dados já tratados e conectei o modelo desenvolvido ao nodo *Random Forest Predictor* que por sua vez está associado aos dados de teste, que são os dados que queremos prever. Para fazer o *tuning* da *Random Forest*, utilizei o nodo *Table Creator* onde criei uma coluna que associa a *string splitCriterion* aos 3 valores possíveis que este parâmetro pode ter na configuração da *Random Forest* (*Information Gain*, *Information Gain Ratio* e *Gini Index*), onde, neste caso, coloquei os valores sem espaços e relativamente ao valor *Gini Index*, coloquei apenas *Gini*. Depois, através do nodo *Table Row to Variable Loop Start*, criei uma variável de fluxo que percorrerá estes 3 possíveis valores. Relativamente aos parâmetros numéricos, utilizei o *loop Parameter Optimization* composto pelos nodos *Parameter Optimization Loop Start* e *Parameter Optimization Loop End*, onde, no primeiro, adicionei 3 variáveis de fluxo: *nTrees* (inteiro que varia de 1 a 20), *nLevels* (inteiro que varia de 1 a 20) e *minNodeSize* (inteiro que varia de 2 a 10) que controlam os parâmetros da *Random Forest*: *Number of models*, *Limit number of levels (tree depth)* e *Minimum node size*, respetivamente. Depois, no nodo *Parameter Optimization Loop End*, é possível observar a *accuracy* que teve qualquer combinação dos parâmetros numéricos e através do nodo *Table Row to Variable* transformamos a combinação de parâmetros numéricos que teve melhor *accuracy* em variáveis de fluxo que serão depois iteradas com os diferentes valores do parâmetro nominal. Após isso, no nodo *Variable Loop End*, conseguimos ver qual a combinação hiper-parâmetro que obteve melhor *accuracy* que neste caso foi de **0.718**.

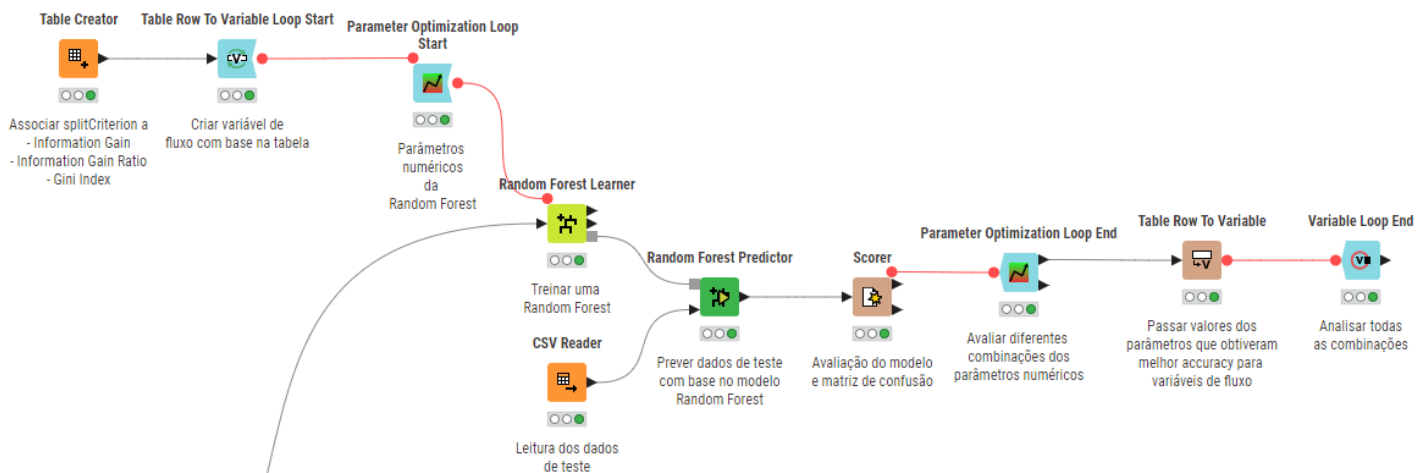


Figura 2.40: *Workflow* relativo à Tarefa 5.

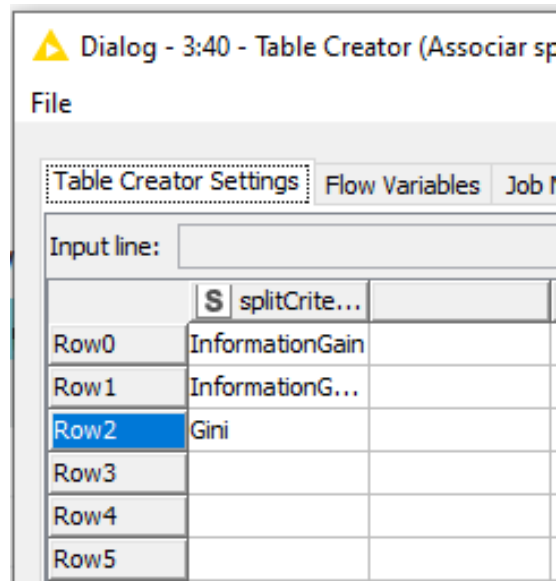


Figura 2.41: Configuração do nodo *Table Creator*.

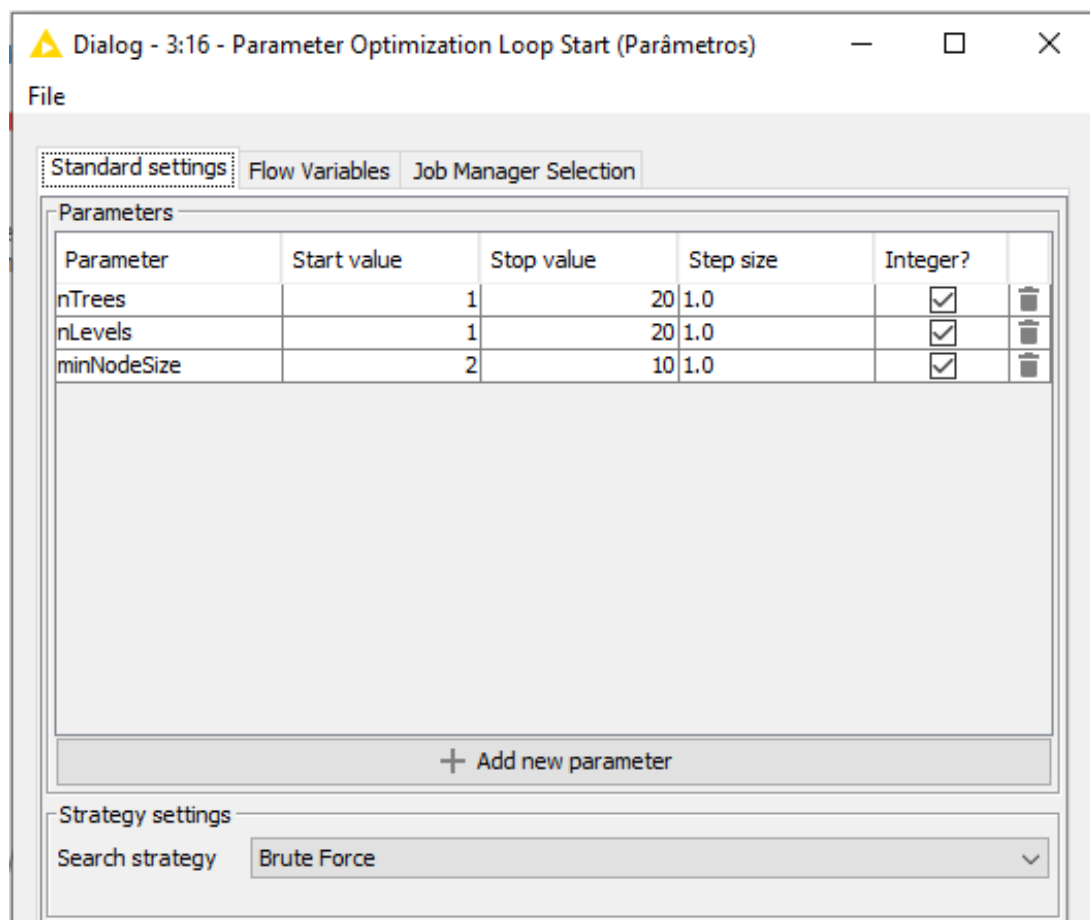


Figura 2.42: Configuração do nodo *Parameter Optimization Loop Start*.

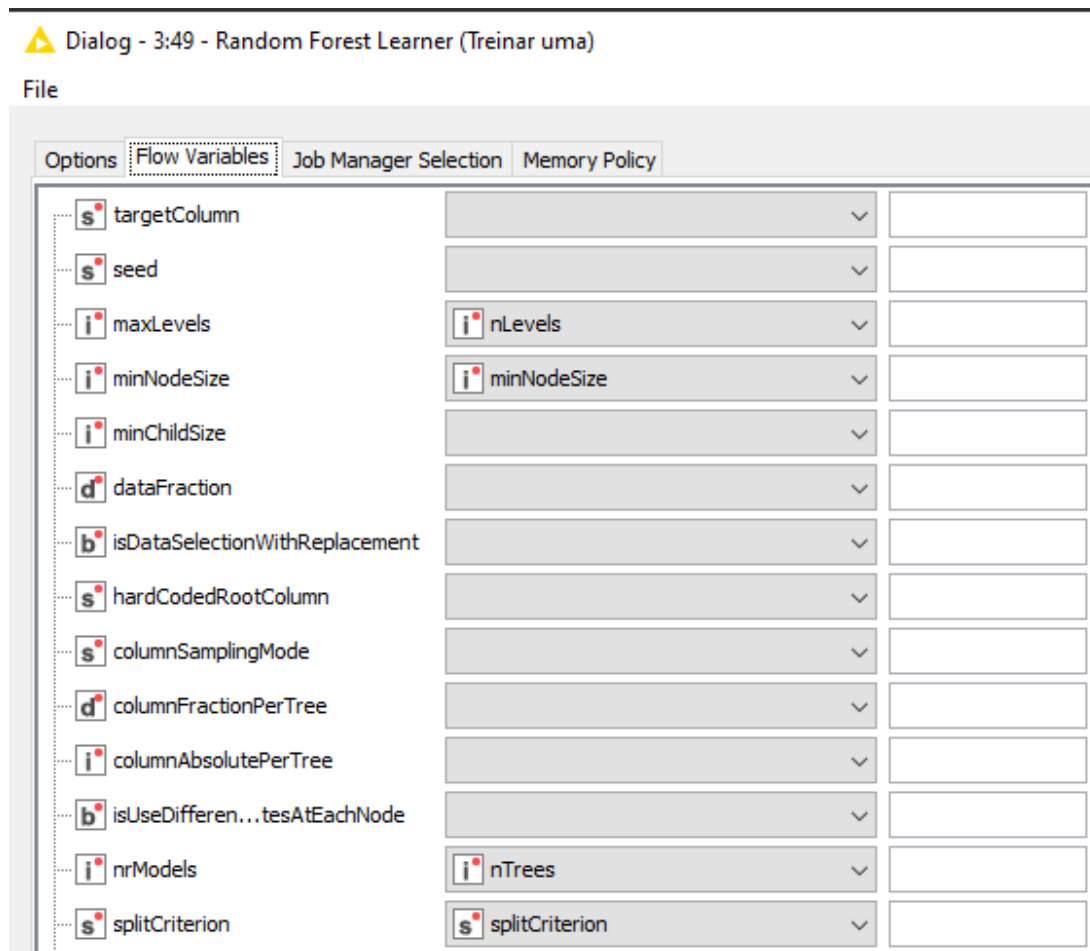



Figura 2.43: Uso das variáveis de fluxo no nodo *Random Forest Learner*.

► 1: Best parameters ► 2: All parameters  Flow Variables

Rows: 1 | Columns: 4

Table Statistics

#	Row...	nTrees Number (integer)	nLevels Number (integer)	minNodeSize Number (integer)	Objective value Number (double)
1	Best ...	15	9	2	0.718

Figura 2.44: Melhor combinação de parâmetros numéricos.

nTrees <small>Number (integer)</small>	▼	nLevels <small>Number (integer)</small>	▼	minNodeSize <small>Number (integer)</small>	▼	Objective value ↓ <small>Number (double)</small>	▼	splitCriterion <small>String</small>
8		7		2		0.718		InformationGainRatio
15		9		2		0.718		Gini
8		7		2		0.706		InformationGain

Figura 2.45: Melhores combinações de todos os parâmetros.

2.6 Tarefa 6

- Analisar e comparar as performances dos modelos treinados na Tarefa 4 e na Tarefa 5. Que conclusões se podem tirar?

Comparando as duas performances dos dois modelos, percebemos que há um ganho de *accuracy* muitíssimo pequeno no modelo de **Random Forest** comparativamente ao modelo de **Decision Tree**. No primeiro, uma das melhores combinações de valores hiper-parâmetro foi:

- Número mínimo de registos por nodo: **7**;
- Medida de qualidade: **Gain Ratio**;
- Método de *pruning*: **MDL**;
- **Accuracy**: **70,6%**.

Já no modelo de *Random Forest*, uma das melhores combinações de valores hiper-parâmetro foi:

- Número de modelos: **15**;
- Número máximo de níveis (*tree depth*): **9**;
- Tamanho mínimo do nodo: **2**;
- Critério de divisão: **Gini Index**;
- **Accuracy**: **71,8%**.

Como frisado anteriormente, a *accuracy* obtida no modelo de *Random Forest* é superior, mas o ganho é muito ligeiro.

2.7 Workflow completo para o caso de uma *Random Forest* (com uso de meta-nodos)

De notar que, no meta-nodo **Leitura e tratamento de dados**, estão presentes todas as tarefas até à Tarefa 2 (inclusivamente), exceto os nodos referentes às vistas gráficas para análise dos dados, como pedido na Tarefa 1, que estão no meta-nodo **Exploração dos dados**. No que toca a treino e *tuning* de um modelo, está ilustrada a parte relativa à *Random Forest*, visto ter sido realizada com ciclos de ciclos (ao contrário da *Decision Tree*) e de ter registado uma *accuracy*, após o *tuning* hiper-parâmetro, ligeiramente superior.

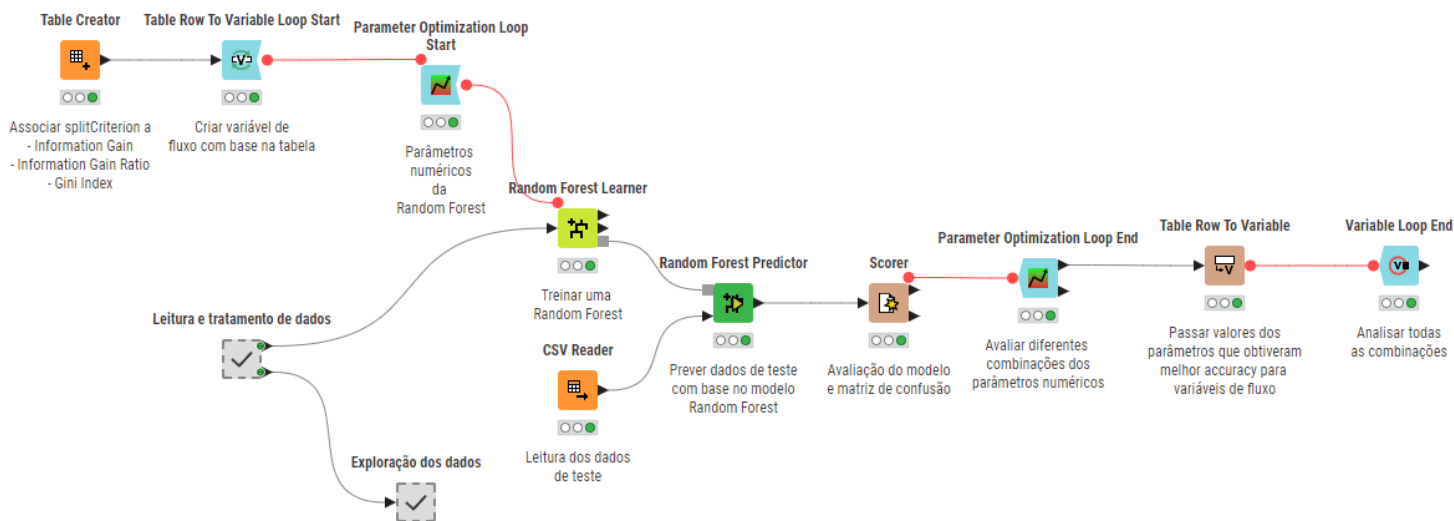


Figura 2.46: Workflow completo com *tuning* de uma *Random Forest*.