Sistemas Baseados em Similaridade

Enunciado Prático Individual 5

José Virgílio Silva Loureiro (PG52252)

T1. Nesta tarefa carreguei no Knime os respetivos datasets. Utilizei o nodo “Table Reader” para ler o conjunto de dados que contém informação sobre cada uma das lojas, incluindo o seu tipo e tamanho. E para o segundo dataset utilizei o nodo “CSV Reader”. De seguida apliquei um nodo “Joiner” que juntou ambos os datasets pelo atributo “Store”.

Para a exploração dos dados criei um metanode com o nome “Dados visualmente”, onde primeiramente utilizo o nodo “Data Explorer”, de onde tirei as seguintes conclusões:

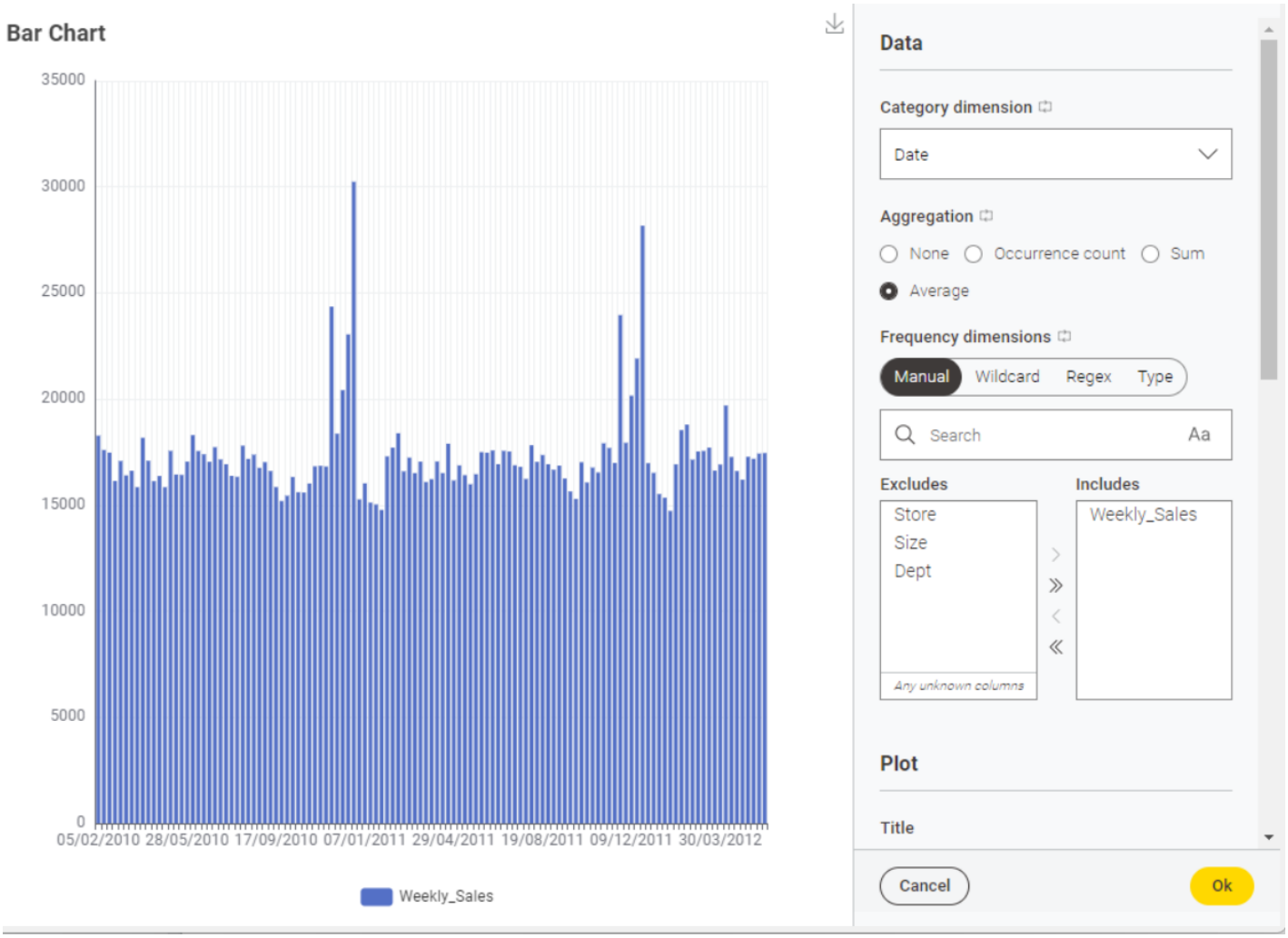
•A feature que apresenta um desvio-padrão maior é o “Size”;

•As features que apresentam maior dispersão em torno da média (Variância) são o “Size” e a “Weekly_Sales”;

•A feature que apresenta maior Skewness e maior Kurtosis é a “Weekly_Sales”;

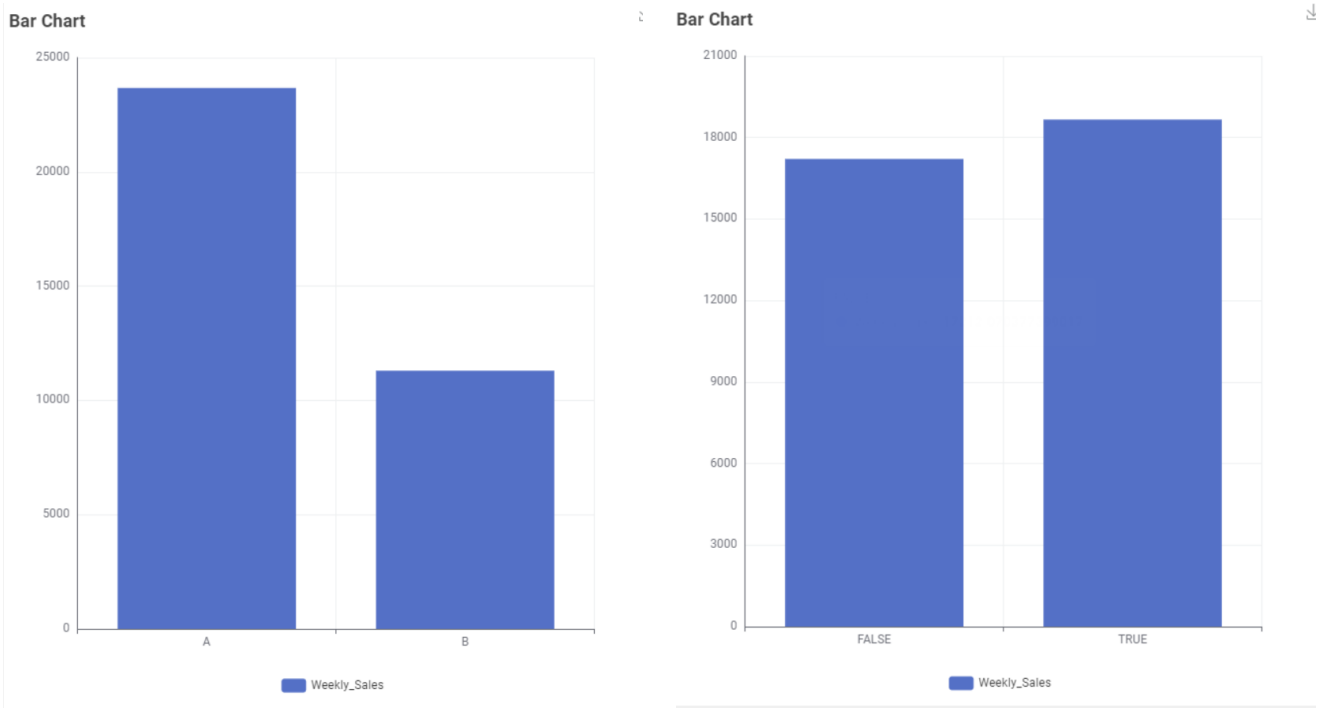
•Não existem “Missing Values”.

Depois usei o nodo Bar Chart para visualizar a média de vendas por data, onde escolhi a “Date” vs “Weekly_Sale”, assim obtive o seguinte gráfico:



Concluindo que nos dias anteriores ao Natal é onde se verifica o maior número de vendas.

Usando esse node comparei também o “Type” com a “Weekly_Sale “ e o “IsHoliday” com a “Weekly_Sale” onde obtive os seguintes gráficos:



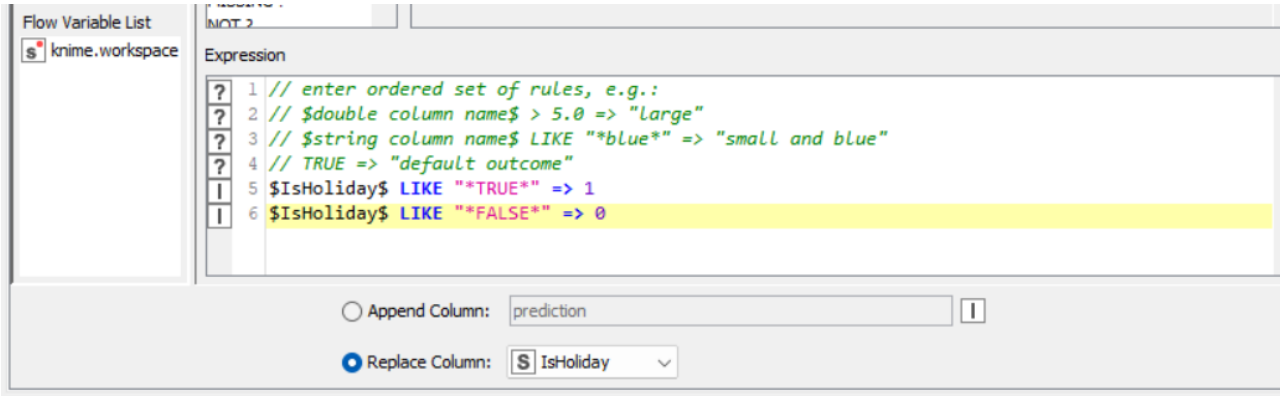
Que me levam a concluir que o “Type” A tem médias de venda mais altas que o B e que existe também uma média de venda maiores quando existe feriados.

Para analisar os dados relativamente à correlação entre features, usei o nodo “Rank Correlation” onde obtive a seguinte tabela de correlação:

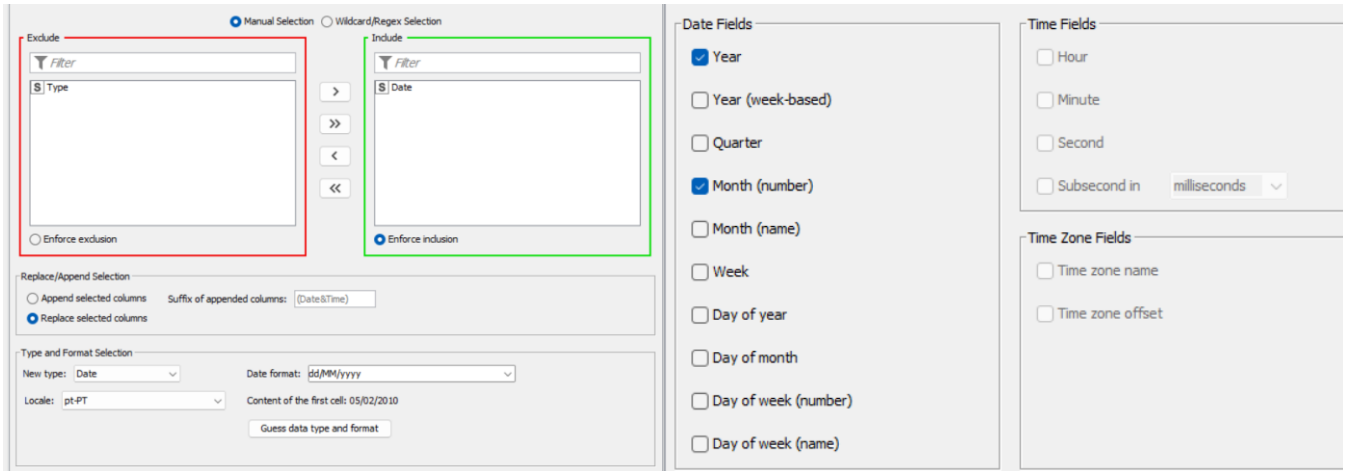
corr = -1 / corr = +1 / corr = n/a	Store	Type	Size	Dept	Date	W...	Is...
Store							
Type							
Size							
Dept							
Date							
Weekly_Sales							
IsHoliday							

Onde observei que existe uma clara correlação negativa entre o “Type” e o “Size”, e uma pequena correlação negativa entre o “Type” e a “Weekly_Sales”.

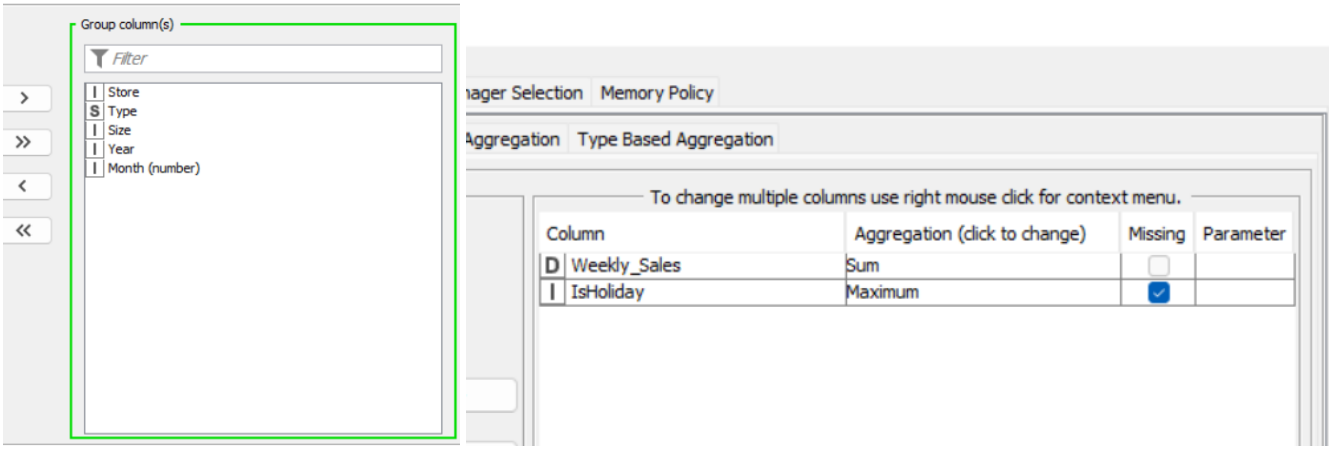
T2. A) Nesta tarefa transformei o atributo *isHoliday* de modo que este passasse de dar valores TRUE e FALSE e para apresentar valores 1 e 0, respetivamente para isso utilizei o nodo “Rule Engine” com as seguintes configurações:



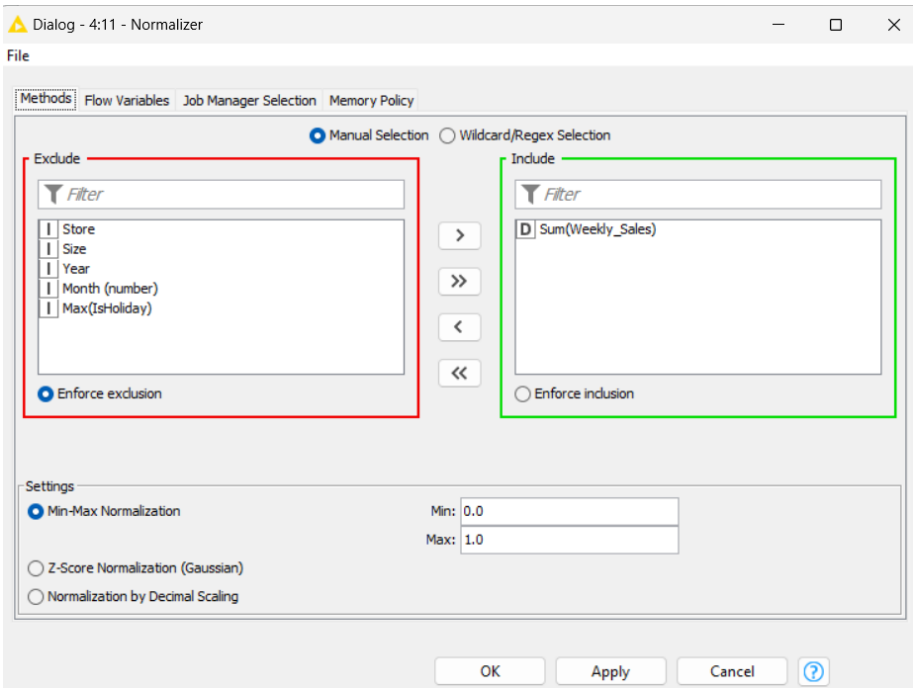
T2. B) Para adicionar a cada registo as features ano e mês utilizei um nodo “String to Date&Time” aplicado ao atributo “Date” para o transformar num formato que seja reconhecido pelo nodo seguinte “Extract Date&Time Fields” que adiciona as colunas indicadas na configuração, Year e Month (number):



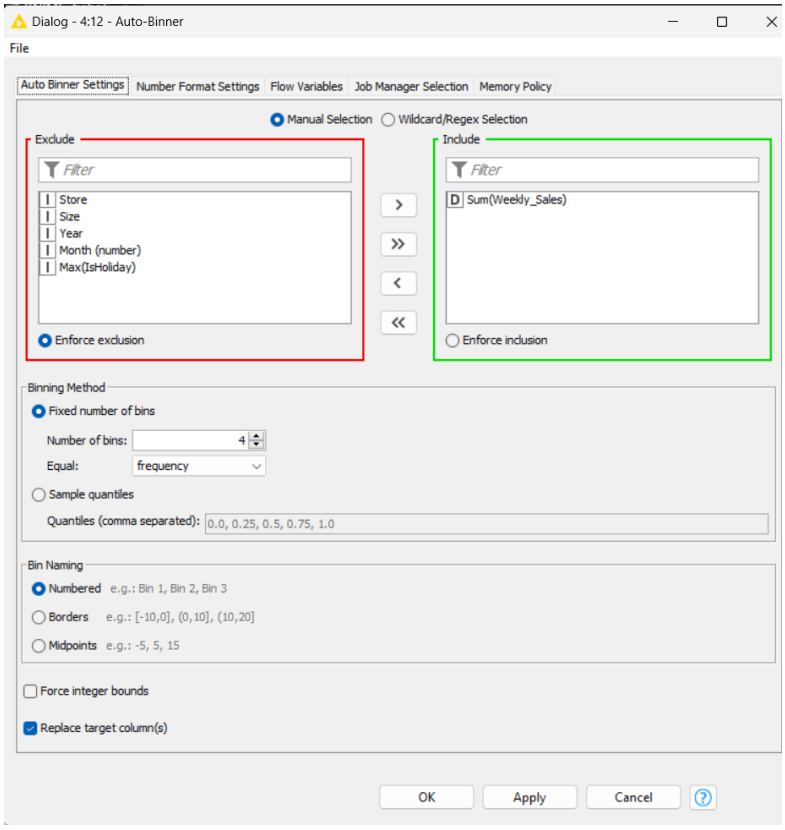
T2. C) Depois utilizei o nodo “GroupBy”, onde agrupei os registos por loja, tipo, tamanho, ano e mês, agregando assim de forma a obter o somatório das vendas semanais de cada loja “(Sum (Weekly_Sales))” e a indicação da existência de feriados nesse mês(Max(IsHoliday)).



T2. D) Na realização desta alínea, utilizei o nodo “Normalizer” de modo a normalizar o somatório das vendas semanais neste nodo utilizei a configuração da transformação linear Min-Max entre 0 e 1.

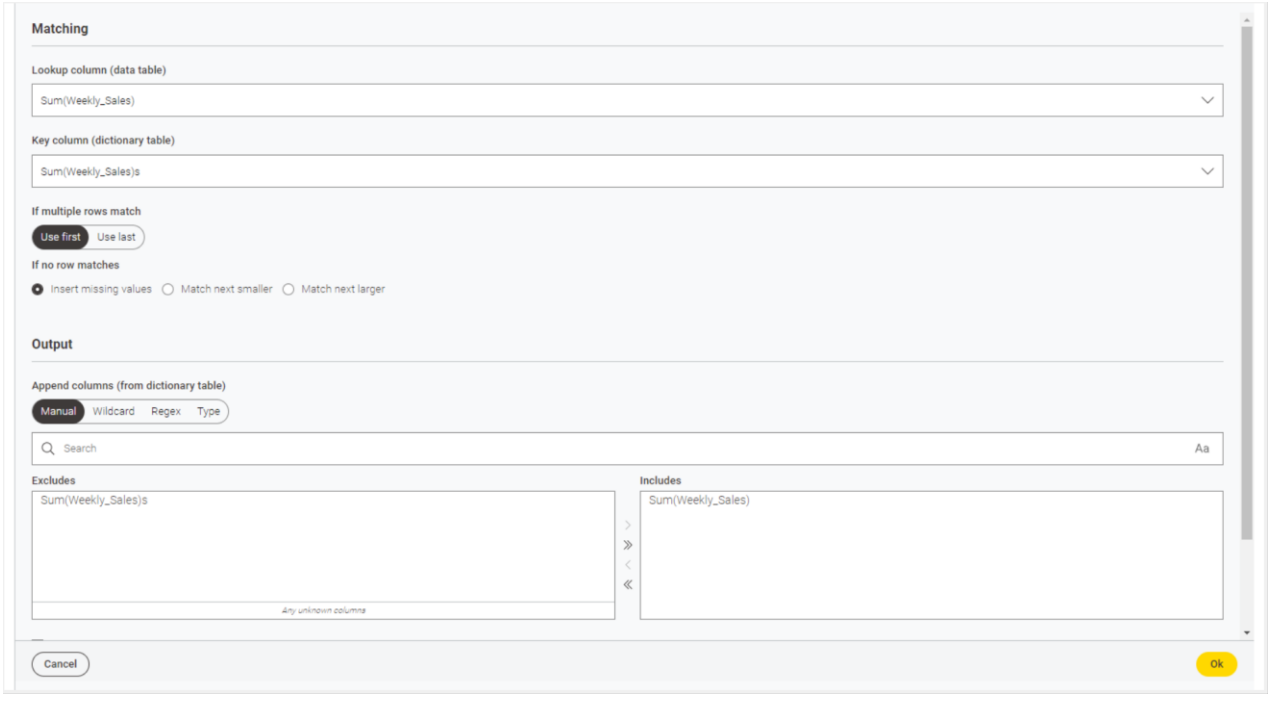


T2. E) e F) Aqui comecei por utilizar o nodo “Auto-Binner” para criar 4 bins de igual frequência sobre o valor normalizado da feature do somatório das vendas semanais (deixando ligada a opção “replace target column(s)”).

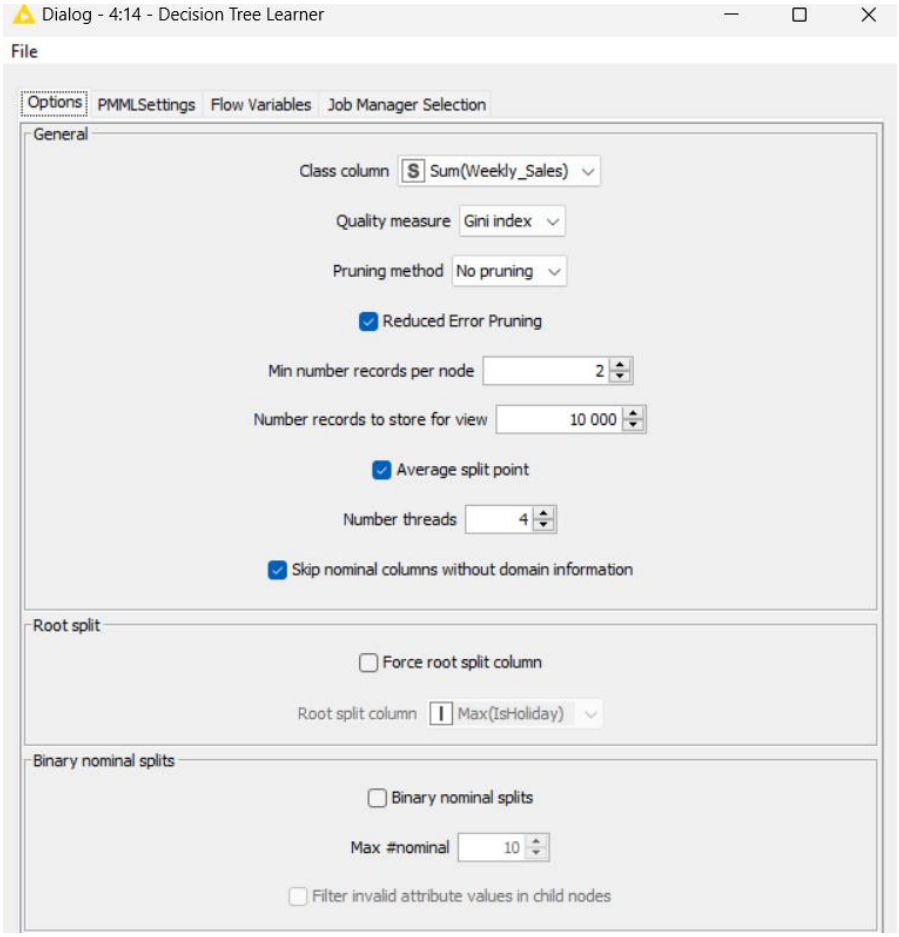


Depois criei a tabela seguinte com o objetivo de renomear cada Bin, ou seja, o primeiro corresponde a Low, o segundo a Medium, o terceiro a High e o quarto a Very High. Para efetivamente substituir esses bins pelos novos, utilizei o nodo “Value lockup”:

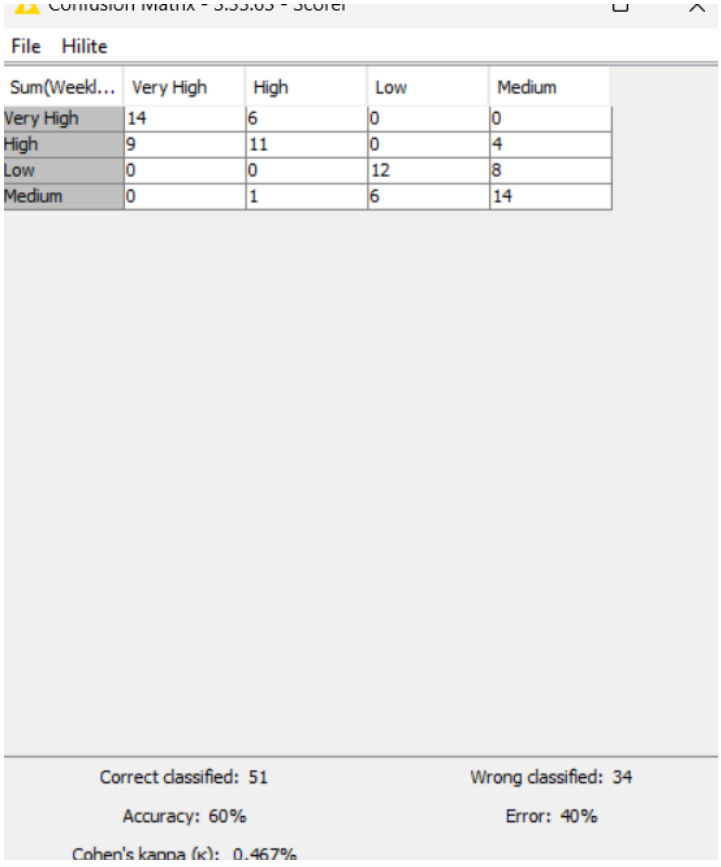
Row0	Bin 1	Low
Row1	Bin 2	Medium
Row2	Bin 3	High
Row3	Bin 4	Very High



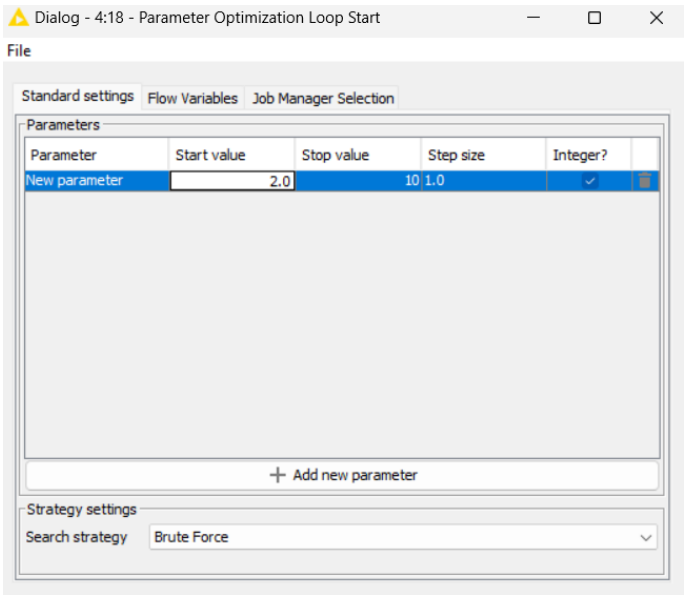
T3. A) e B) e C) Nesta tarefa usei um nodo “Decision Tree Learner” que recebe os dados já processados do dataset para treinar o modelo de modo a estimar as vendas mensais de cada uma das suas lojas.

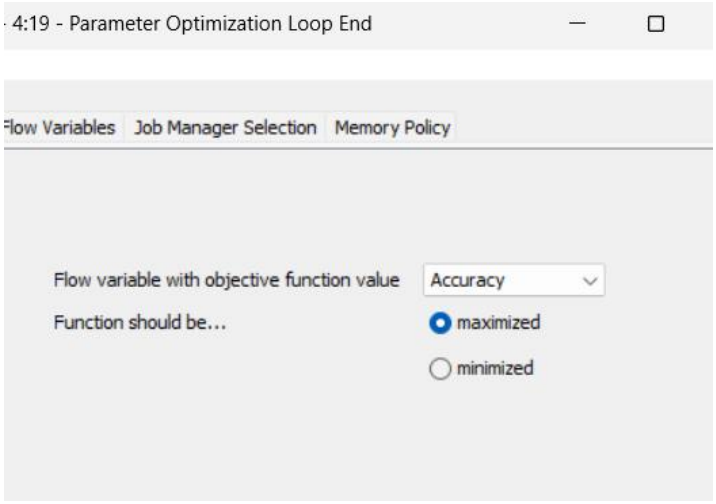


Depois, apliquei um nodo “Decision Tree Predictor” que recebe os dados de teste de um nodo “CSV Reader” e também recebe o modelo de árvore de decisão treinado e prevê o valor de vendas de cada mês. Depois usei o nodo “Scorer” para poder visualizar a accuracy e a matriz de confusão:

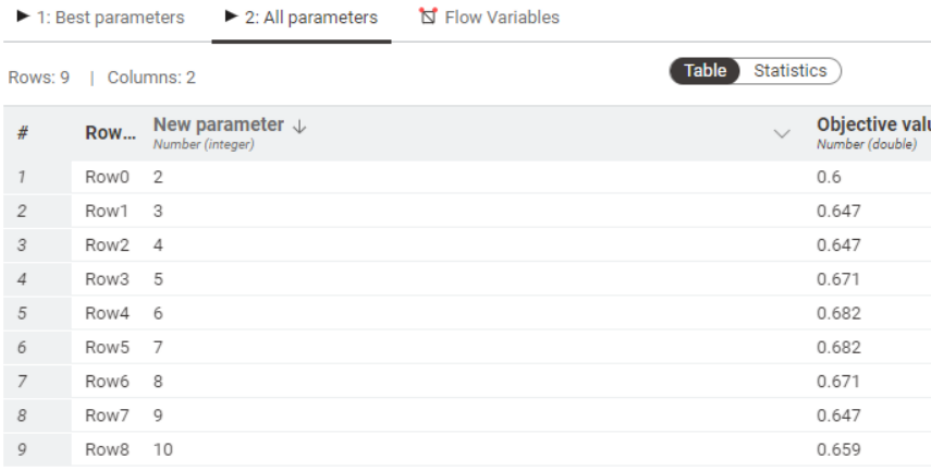


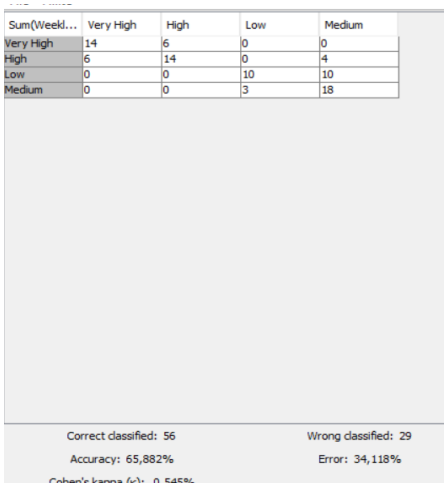
T4. A) Para a realização desta tarefa utilizei o nodo “Parameter Optimization Loop Start” que cria um ciclo de testes com uma flow variável “New parameter” e depois usei o nodo “Parameter Optimization Loop End” que fecha o ciclo, para fazer o tunning do modelo, experimentando todos os valores, entre 2 e 10, para o número mínimo de registo por nodo, assim usei as seguintes configurações, respetivamente





E obtive que o melhor número mínimo de registo foi 6 com uma accuracy de 65.882%

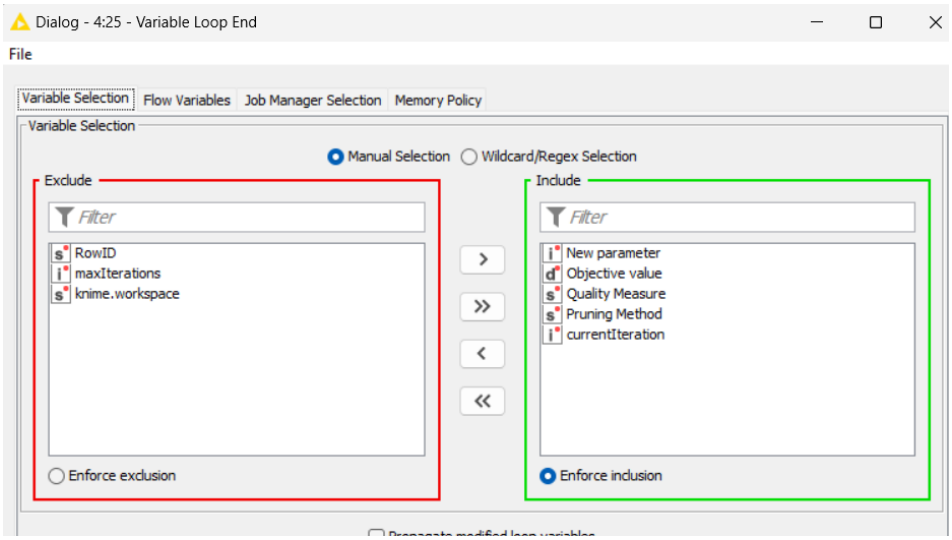




T4. B) e C) e D) Nesta tarefa comecei por usar um outro loop para testar as possibilidades de medida de qualidade e do método de pruning em simultâneo. Para isso criei um nodo “Table Creator”, com os registos seguintes, e liguei a um nodo “Table Row To Variable Loop Start”.

	S Pruning ...	S Quality ...
Row0	No pruning	Gini index
Row1	No pruning	Gain radio
Row2	MDL	Gini index
Row3	MDL	Gain radio

Para fechar estes ciclos, primeiramente utilizei o nodo “Table Row To Variable”, com os inputs “Objetive Value” e “New parameter”, depois usei o nodo “Variable Loop End”, com as seguintes configurações:



Com isso obtive os seguintes resultados:

#	Row...	New parameter *Number (integer)*	Objective value *Number (double)*	Quality Measure *String*	Pruning Method *String*	currentIteration *Number (integer)*
1	Row0	6	0.682	Gini index	No pruning	0
2	Row1	2	0.694	Gini index	MDL	1
3	Row2	3	0.682	Gain ratio	No pruning	2
4	Row3	2	0.706	Gain ratio	MDL	3

Onde podemos ver que a combinação que oferece melhor performance é utilizando 2 registos por nodo com a medida de qualidade Gain Ratio e o método de pruning MDL. Apesar de haver um aumento de performance não é um aumento muito significativo e não existe muita discrepância. Mesmo usando a melhor combinação de fatores a nossa accuracy continua a ser apenas 68%.

Sum(Weekl...	Very High	High	Low	Medium
Very High	16	4	0	0
High	9	11	0	4
Low	0	0	15	5
Medium	0	0	5	16

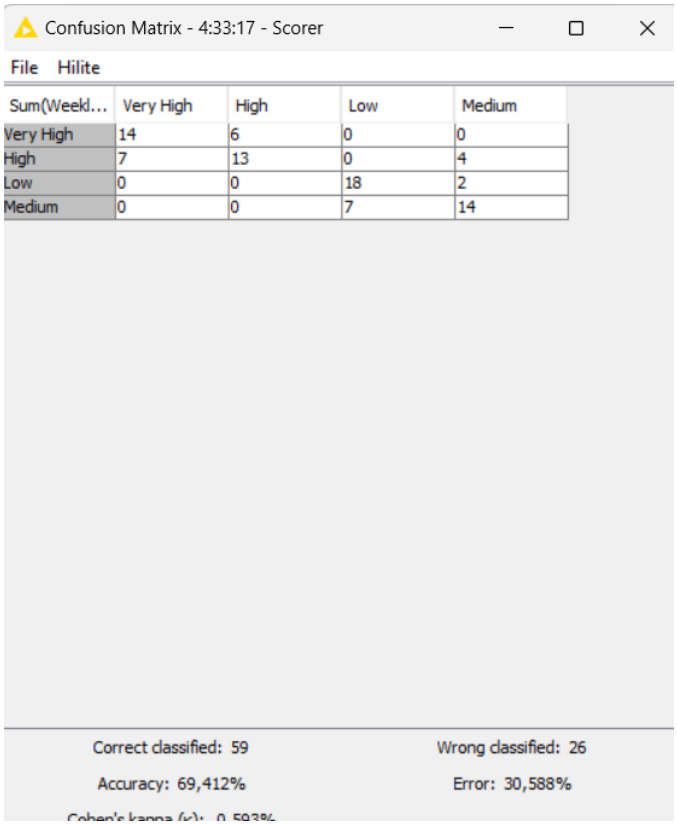
Correct classified: 58 Wrong classified: 27

Accuracy: 68,235% Error: 31,765%

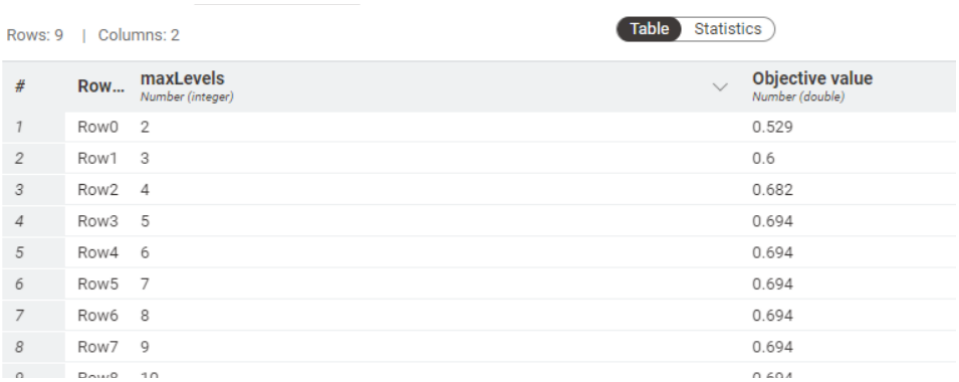
Cohen's kappa (κ): 0,578%

Para guardar a tabela de performance da melhor combinação dos hiper parâmetros usei o nodo”Csv Writer”.

T5. Em primeiro lugar, utilizei o mesmo procedimento que para o modelo da Decision Tree mas desta vez para uma Random Forest. Utilizei o nodo “Random Forest Learner” ao qual fiz chegar os dados preparados anteriormente para treino, e liguei-o ao nodo “Random Forest Predictor” que recebe também os dados para teste. Para medir a precisão utilizei um nodo “Scorer” que avaliou este modelo com uma accuracy superior à accuracy do modelo inicial do Decision Tree.

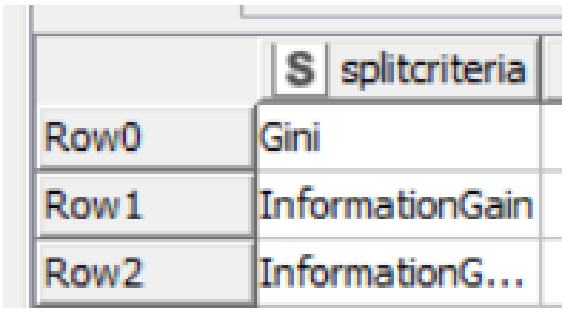


Depois, usei um loop para experimentar valores para o número máximo de níveis. Para isso apliquei o nodo “Parameter Optimization Loop Start” que passa uma variável “maxlevels” que varia nos inteiros entre 2 e 10 para o Forest Tree Learner e fechei o ciclo com o nodo “Parameter Optimization Loop End”. Obtive o seguinte resultado:



#	Row...	maxLevels (Number (integer))	Objective value (Number (double))
1	Row0	2	0.529
2	Row1	3	0.6
3	Row2	4	0.682
4	Row3	5	0.694
5	Row4	6	0.694
6	Row5	7	0.694
7	Row6	8	0.694
8	Row7	9	0.694

Depois utilizei outro loop para testar as possibilidades de split criterion. Para isso associei um nodo “Table Creator”, com os registos seguintes, a um nodo “Table Row To Variable Loop Start”.

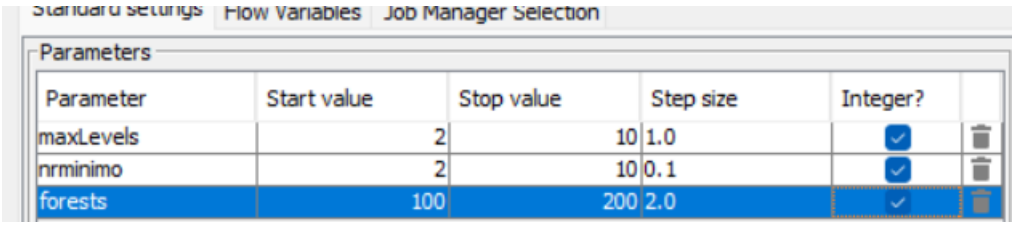


Para terminar os ciclos, novamente utilizei o nodo “Table Row To Variable”, com os inputs “Objetive Value” e “maxLevels”, depois usei o nodo “Variable Loop End”, obtendo a seguinte tabela:

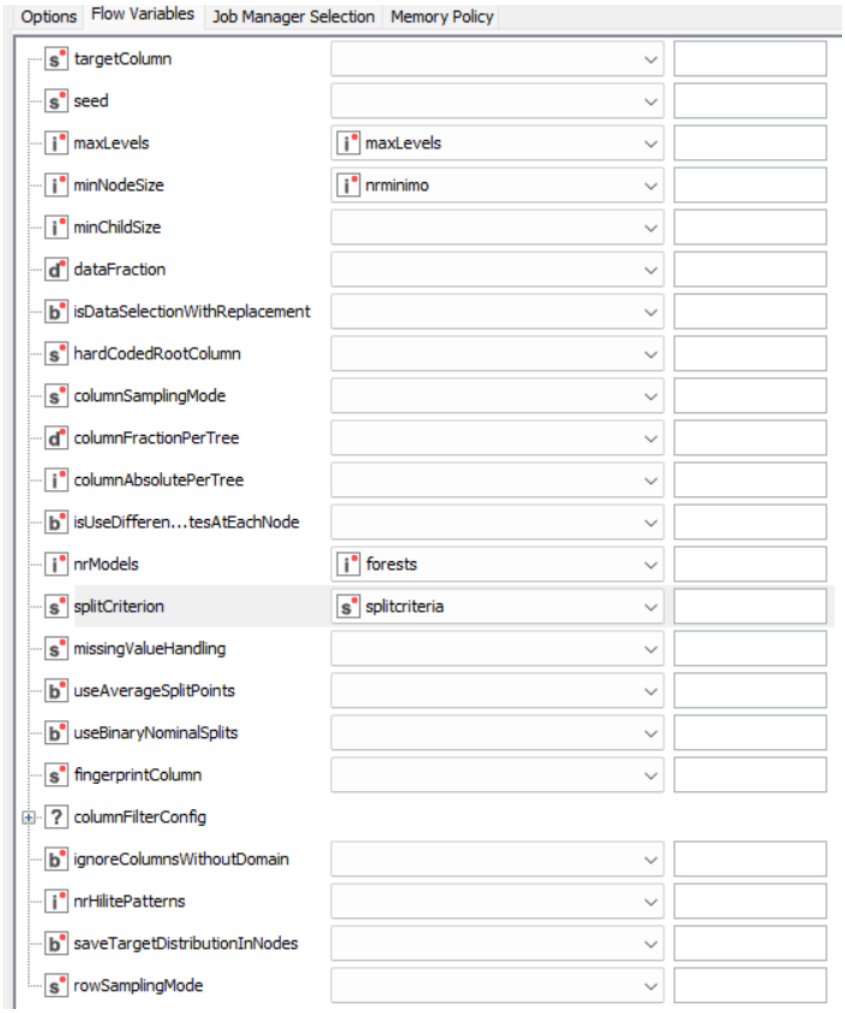
Rows: 3 | Columns: 4 — Table | Statistics

#	Row...	maxLevels Number (integer)	Objective value Number (double)	splitcriteria String	currentIteration Number (integer)
1	Row0	5	0.694	Gini	0
2	Row1	5	0.694	InformationGain	1
3	Row2	5	0.694	InformationGainRatio	2

Por fim para testar se diferentes números de modelos fariam o Forest Tree Learner prever com mais precisão adicionei duas variáveis no node “Parameter Optimization Loop Start”:



E configurei o node “Forest Tree Learner”:



Obtive por fim os seguintes resultados:

#	Row...	maxLevels Number (integer)	nrminimo Number (integer)	forests Number (integer)	Objective value Number (double)	splitcriteria String	currentIteration Number (integer)
1	Row0	10	2	104	0.718	Gini	0
2	Row1	10	4	100	0.706	InformationGain	1
3	Row2	10	2	102	0.706	InformationGainRatio	2

Fazendo a tabela de confusão da melhor combinação dos hiperparametros obtenho :

Sum(Weekl...	Very High	High	Low	Medium
Very High	14	6	0	0
High	7	13	0	4
Low	0	0	18	2
Medium	0	0	6	15

Correct classified: 60 Wrong classified: 25

Accuracy: 70,588% Error: 29,412%

Cohen's kappa (κ): 0,609%

Ou seja, apesar do tuning do modelo ele não conseguiu melhorar assim muito a sua accuracy. O que considero ter haver com o dataset sobre o qual estamos a treinar o modelo. Com mais data acredito que o modelo tivesse maior previsão de acuracy.

T6. Olhando agora para as performances dos modelos treinados em T4 e em T5, posso concluir que o modelo de tunning de uma Random Forest foi me mais lento a fazer tuning do que o modelo de tunning de uma Decision Tree no entanto o modelo que apresenta melhor Accuracy é o modelo da Random Forest uma vez que tem Accuracy de 70,5% enquanto o modelo da Tuned da Decision Tree tem de Accuracy de 68%. Acredito que com mais dados a diferença pudesse ser maior visto que fazer o tuning do modelo da Random Forest não alterou o seu nível de previsão significativamente.