Universidade do Minho
Escola de Engenharia

Perfil de Machine Learning: Fundamentos e Aplicações

Sistemas Baseados em Semelhança

1º/2º Ano, 1º Semestre

Ano letivo 2022/2023

Enunciado Prático nº 5

Tema: Tuning de Modelos Baseados em Árvores

Joana Mota PG45528

Tarefas:

T1. Descarregou-se dois *datasets*, em que um contem a informação sobre cada loja, o seu tipo e tamanho e outro que contem a informação sobre as vendas semanais de cada departamento, de cada loja, a data e se houve feriado. Juntou-se os dois através do nodo *joiner e* verifica-se que há 17 lojas com dois tipos, A e B, com vários departamentos, do 1 ao 99.

Através de um gráfico pie, consegue-se analisar a soma das vendas semanais por tipo de loja, sendo que o tipo A apresenta 67% e o tipo B apresenta 33% das vendas semanais. Assim, as lojas com tipo A apresentam maior receita do que as lojas do tipo B.

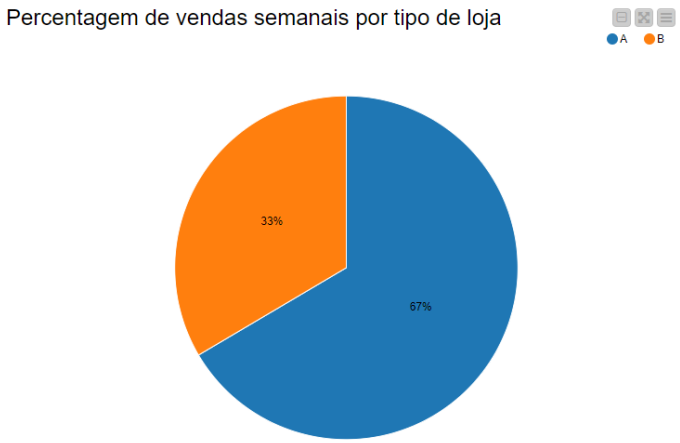


Figura 1. Percentagem de vendas semanais por tipo de loja

No entanto, verifica-se que existe menos lojas do tipo A, nomeadamente, 8 lojas do tipo A e 9 lojas do tipo B.

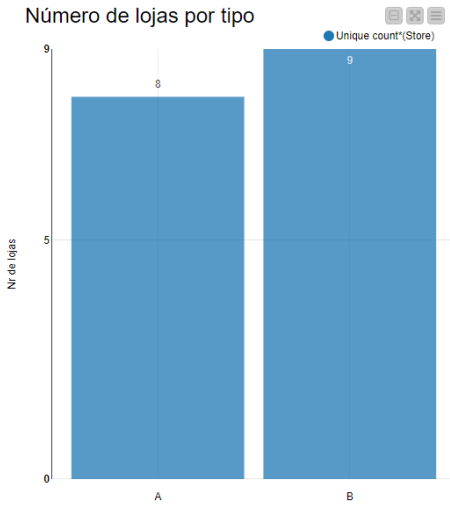


Figura 2. Número de lojas por tipo

Também se verifica que as vendas semanais aumentam a sua receita quando há feriados durante a semana, apresentando, assim, 52% de receita com feriados e 48% de receita sem feriados.

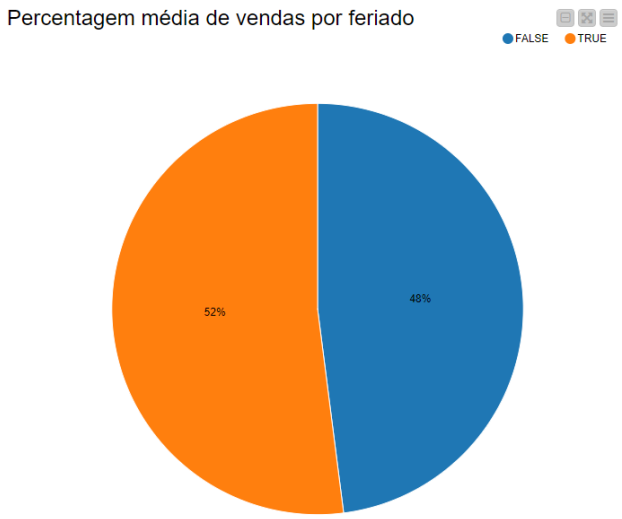


Figura 3. Percentagem média de vendas por feriado

T2. Primeiramente, começou-se por passar a *feature Isholiday* para *integer* de modo que o valor *True* correspondesse a 1, assim utilizou-se o *nodo category to number*.

De seguida, a partir da data extraiu-se as *features* ano e mês, através do nodo *Extract data&Fields*, tendo sido passado para *string* a *feature date*, primeiramente. Posteriormente, utilizou-se o nodo *Group By* para agrupar os registos por loja, tipo, tamanho, ano e mês, de modo a obter o somatório das vendas semanais por loja e a indicação da existência de feriados nesse mês. Utilizou-se um SUM para as vendas e um Max nos feriados, o máximo será o valor 1, ao qual corresponde a existência de feriados. Depois normalizou-se as vendas semanais entre 0 e 1, através do nodo *normalizer* e, com o nodo *auto binner*, criou-se 4 bins para as vendas fazendo um *replace target column*, de modo a substituir os valores.

No final, renomeou-se os bins através do nodo *rule engine*, em que o bin 1, 2, 3, e 4 que corresponde a low, medium, high e very high, respetivamente.

T3. Para treinar e testar o modelo, descarregou-se o *dataset* de teste e utilizou-se o algoritmo Árvore de Decisão para prever o valor das vendas de cada mês para cada loja. Utilizou-se o nodo scorer (Javascript) tendo obtido uma *accuracy* de 0.6. A matriz de confusão está representada na tabela seguinte.

Scorer View

Confusion Matrix

	High (Predicted)	Low (Predicted)	Medium (Predicted)	Very High (Predic...	
High (Actual)	11	0	4	9	45.83%
Low (Actual)	0	12	8	0	60.00%
Medium (Actual)	1	6	14	0	66.67%
Very High (Actual)	6	0	0	14	70.00%
	61.11%	66.67%	53.85%	60.87%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
60.00%	40.00%	0.467	51	34

Tabela 1. Matriz de confusão

T4. Para otimizar o modelo, começou-se por utilizar o nodo *parameter optimization loop start*, para saber o número mínimo registos por nodo, entre os valores de 2 e 10. Criou-se uma variável chamada "MinRegistosPorNodo", ao qual configurou-se no nodo *decision tree learner*.

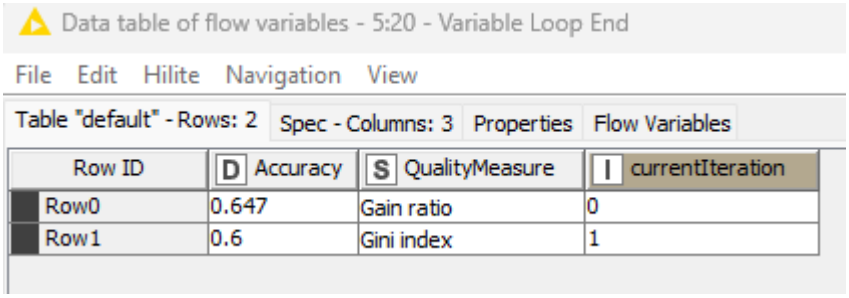
Depois, no final do nodo *scorer*, aplicou-se o nodo p*arameter optimization loop end* para obter os resultados, verificando assim que o melhor número é 6 registos por nodo para obter o máximo de score.

Row ID	MinRegi...	Objecti...
Best parameters	6	0.682

Row ID	MinRegi...	Objecti...
Row0	2	0.6
Row1	3	0.647
Row2	4	0.647
Row3	5	0.671
Row4	6	0.682
Row5	7	0.682
Row6	8	0.671
Row7	9	0.647
Row8	10	0.659

Tabela 2. Resultado da variável min registos por nodo

De seguida, para saber todas as possibilidades para a medida de qualidade utilizou-se os nodos *table creator* e *table row to variable loop start.* O nodo *table creator* serviu para definir uma coluna com as medidas de qualidade, nomeadamente, "Gain ratio" e "Gini index", ligando, posteriormente, ao nodo *table row to variable loop start* para incluir a coluna definida anteriormente. Depois, obteve-se os resultados através do nodo *variable loop end*, onde a medida de qualidade "Gain ratio" é a que apresenta maior *accuracy*. Ou seja, de forma a poder ter o máximo de score, os dados de treino têm de ser divididos através da medida de "Gain ratio".



Row ID	Accuracy	QualityMeasure	currentIteration
Row0	0.647	Gain ratio	0
Row1	0.6	Gini index	1

Tabela 3. Resultado da variável Quality Measure

Por último, utilizando os mesmos nodos, otimizou-se o modelo de forma a poder ver todas as possibilidades para o método *pruning*, isto é, se é necessário ou não fazer *pruning*. Assim, através dos resultados, verifica-se que é necessário fazer *pruning* ao modelo para maximizar a sua *accuracy*, quer seja com a medida de qualidade Gain ratio ou Gini Index. O objetivo do método *pruning* é reduzir a complexidade, evitando o *overfitting* dos dados, e melhorar a taxa de acerto.

Row ID	MinRegistosPorN...	Objective value	Pruning...	QualityMeasure
Row0	3	0.682	No pruning	Gain index
Row1	2	0.706	MDL	Gain index

Tabela 4. Resultado da variável pruning

De forma a colocar todos os parâmetros anteriores num único *workflow*, guardou-se os resultados todos no nodo *Loop End*. Verifica-se, assim, que a combinação que aumenta o score ao modelo é ter o mínimo de 2 registos por nodo, com uma medida de qualidade "Gain ratio" e utilizar *pruning*, isto é, MDL. No entanto, comparando às outras combinações, não existe grandes discrepâncias.

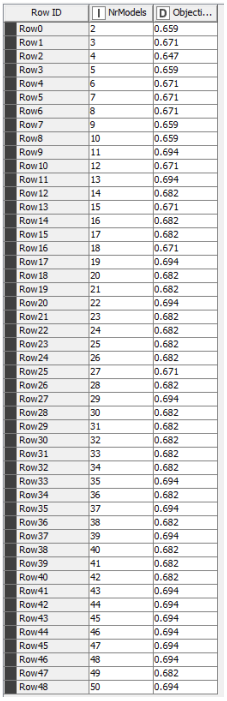
Row ID	MinRegistos...	Objective value	PruningM...	Quality...
Row0#0	3	0.682	No pruning	Gain ratio
Row1#0	2	0.706	MDL	Gain ratio
Row0#1	3	0.682	No pruning	Gain index
Row1#1	2	0.706	MDL	Gain index

Tabela 5. Resultados de todas as variáveis

T5. Nesta fase, foi utilizado outro tipo de algoritmo para modelos de *machine learning*, nomeadamente, o *random forest* para treinar e fazer o *tuning*. Aplicou-se então o nodo *random forest learner* para treinar e o nodo *random forest predictor* para testá-lo. Para além disto, fez-se a otimização do modelo. Primeiramente, criou-se uma variável "NrModels" no nodo *parameter*

optimization loop start, com os valores entre 1 e 50. Obtendo, depois, 11 árvores necessárias para maximizar a *accuracy* do modelo.

Row ID	NrModels	Objecti...
Best parameters	11	0.694



Row ID	NrModels	Objecti...
Row0	2	0.659
Row1	3	0.671
Row2	4	0.647
Row3	5	0.659
Row4	6	0.671
Row5	7	0.671
Row6	8	0.671
Row7	9	0.659
Row8	10	0.659
Row9	11	0.694
Row10	12	0.671
Row11	13	0.694
Row12	14	0.682
Row13	15	0.671
Row14	16	0.682
Row15	17	0.682
Row16	18	0.671
Row17	19	0.694
Row18	20	0.682
Row19	21	0.682
Row20	22	0.694
Row21	23	0.682
Row22	24	0.682
Row23	25	0.682
Row24	26	0.682
Row25	27	0.671
Row26	28	0.682
Row27	29	0.694
Row28	30	0.682
Row29	31	0.682
Row30	32	0.682
Row31	33	0.682
Row32	34	0.682
Row33	35	0.694
Row34	36	0.682
Row35	37	0.694
Row36	38	0.682
Row37	39	0.694
Row38	40	0.682
Row39	41	0.682
Row40	42	0.682
Row41	43	0.694
Row42	44	0.694
Row43	45	0.694
Row44	46	0.694
Row45	47	0.694
Row46	48	0.694
Row47	49	0.682
Row48	50	0.694

Tabela 6. Resultado da variável Nr de Models

De seguida, otimizou-se o modelo criando a variável "MaxTree" no nodo *parameter optimization loop start* para verificar o máximo de profundidade em cada árvore, entre os valores de 2 e 10. Obteve-se, assim, como resultado o máximo de 4 de profundidade. Quanto mais profunda a árvore, mais pesado e mais lento fica o modelo.

Row ID	MaxTree	Objecti...
Best parameters	4	0.694

Row ID	MaxTree	Objecti...
Row0	2	0.553
Row1	3	0.6
Row2	4	0.694
Row3	5	0.694
Row4	6	0.694
Row5	7	0.694
Row6	8	0.694
Row7	9	0.682
Row8	10	0.694

Tabela 7. Resultado da variável Max Tree

Depois, utilizando o mesmo nodo, criou-se uma variável "NodeSize", com os valores entre 2 e 25, obtendo no final como melhor resultado o tamanho de 6.

Row ID	NodeSize	Objecti...
Best parameters	6	0.706

Row ID	NodeSize	Objecti...
Row0	2	0.694
Row1	3	0.694
Row2	4	0.694
Row3	5	0.694
Row4	6	0.706
Row5	7	0.694
Row6	8	0.694
Row7	9	0.694
Row8	10	0.694
Row9	11	0.694
Row10	12	0.694
Row11	13	0.694
Row12	14	0.694
Row13	15	0.694
Row14	16	0.694
Row15	17	0.694
Row16	18	0.694
Row17	19	0.694
Row18	20	0.694
Row19	21	0.694
Row20	22	0.694
Row21	23	0.694
Row22	24	0.694
Row23	25	0.694

Tabela 8. Resultado da variávrel Node Size

Posteriormente, criou-se e acrescentou-se a variável "*SplitCriterion*" no nodo *table creator* com "InformationGain", "InformationGainRatio" e "Gini" e, introduzindo depois, no nodo *table row to variable loop sart*. Juntou-se as variáveis anteriores no mesmo *loop*, bem como esta última às mesmas. No final, obteve-se os resultados de todos os parâmetros no nodo *variable loop end*. A melhor combinação é haver 29 árvores, com o máximo de 9 de profundidade, tamanho de 12 nos nós e com um *split* de Gini index. Mais uma vez, não existe grandes discrepâncias entre as combinações.

Row ID	MaxTre...	NrModels	NodeSize	Objecti...	SplitCrite...
Row0	7	15	11	0.694	InformationGain
Row1	9	33	21	0.694	InformationGa...
Row2	9	29	12	0.706	Gini

Tabela 9. Resultado de todas as variáveis

T6. Para analisar e comparar as performances dos modelos treinados, voltou-se a treinar cada um deles, mas desta vez com os melhores resultados dos parâmetros, inserindo no nodo learner dos algoritmos. Verifica-se que o melhor modelo para estes dados é com a *decision tree*, pois apresenta uma *accuracy* maior que a *random forest*, 70% e 63%, respetivamente. Por norma, a *decision tree* é um bom algoritmo para dados mais simples, por isso talvez seja esse o motivo para, neste caso, ser a melhor opção para o modelo. E verifica-se, através da matriz de confusão, que é pouca diferença de acertos entre um modelo e outro.

Confusion Matrix

	High (Predicted)	Low (Predicted)	Medium (Predicted)	Very High (Predic...	
High (Actual)	14	0	4	6	58.33%
Low (Actual)	0	18	2	0	90.00%
Medium (Actual)	0	7	14	0	66.67%
Very High (Actual)	6	0	0	14	70.00%
	70.00%	72.00%	70.00%	70.00%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
70.59%	29.41%	0.608	60	25

Decision Tree

Tabela 10. Matriz de confusão da Decision Tree

Confusion Matrix

	High (Predicted)	Low (Predicted)	Medium (Predicted)	Very High (Predic...	
High (Actual)	14	0	3	7	58.33%
Low (Actual)	0	15	5	0	75.00%
Medium (Actual)	4	6	11	0	52.38%
Very High (Actual)	6	0	0	14	70.00%
	58.33%	71.43%	57.89%	66.67%	

Overall Statistics

Overall Accuracy	Overall Error	Cohen's kappa (κ)	Correctly Classified	Incorrectly Classified
63.53%	36.47%	0.513	54	31

Random Forest

Tabela 11. Matriz de confusão da Random Forest