Mini-teste 1 **Vitor Peixoto - A79175**
Mineração de Dados 14/10/2018
Mestrado em Engenharia Informática Universidade do Minho

Introdução

Neste mini-teste foi atribuído um conjunto de três questões baseados na análise de conjuntos de dados utilizando diferentes classificadores. Os exercícios baseiam-se na análise e comparação da *performance* dos diversos classificadores utilizados e quando sujeitos a alterações dos parâmetros e algoritmos que os compõem.

Exercício 1

Este primeiro exercício pede para observar o *dataset* ***segment***, em conjunto com o seu ficheiro de teste e de dados de treino, alternando as várias opções de *pruning* existentes.

Primeiro temos de entender a importância do *pruning* numa árvore. Esta técnica permite simplificar o modelo de previsão de forma a combater o *overfitting* dos dados de treino.

Como **primeiro caso**, vamos correr o algoritmo *J48* utilizando a técnica de *pruning* com *subtree raising*. Esta técnica elimina um nó interior da árvore e sobe a subárvore abaixo desse nó, um nível. Utilizando os valores por defeito do *Weka*, obtemos os seguintes resultados:

```
Correctly Classified Instances         779               96.1728 %
Incorrectly Classified Instances        31                3.8272 %
Kappa statistic                          0.9553
Mean absolute error                      0.0127
Root mean squared error                  0.1005
Relative absolute error                  5.1771 %
Root relative squared error             28.6807 %
Total Number of Instances              810
```

Figura 1: Resultados obtidos recorrendo à técnica de *subtree raising* com os dados de teste.

```
Correctly Classified Instances        1485               99      %
Incorrectly Classified Instances        15                1      %
Kappa statistic                          0.9883
Mean absolute error                      0.0048
Root mean squared error                  0.0488
Relative absolute error                  1.9473 %
Root relative squared error             13.9545 %
Total Number of Instances              1500
```

Figura 2: Resultados obtidos recorrendo à técnica de *subtree raising* com os dados de treino.

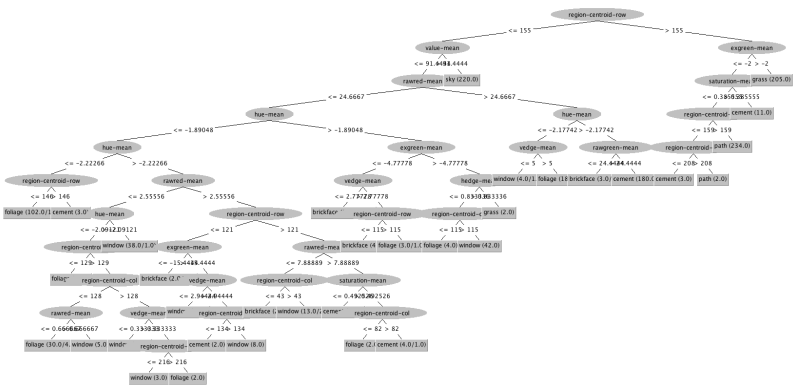

Figura 3: Árvore obtida com os dados de teste e de treino, com 67 nós e 34 folhas.

Usando os dados de treino, obtemos melhores resultados, mas são **resultados enganadores**, visto que o algoritmo memoriza as respostas. O método mais correto para testar o classificador no *dataset* será através da utilização de um conjunto de dados de teste independentes.

No **segundo caso**, vamos correr o mesmo algoritmo *J48* optando pela alteração do valor do fator de confiança. Este parâmetro testa a eficiência do *post-pruning*. Ao diminuir este valor, diminuimos também a quantidade de *post-pruning* à qual a árvore é sujeita.
Neste caso, apresentamos resultados para valores do fator de confiança que variam de valores próximos de 0 até 0.5, em intervalos de 0.1. O motivo pelo qual paramos em 0.5 é porque a partir deste valor não ocorre *pruning* dos dados.

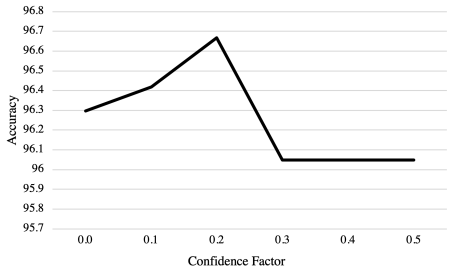

Figura 4: Resultados obtidos com a variação do fator de confiança para os dados de teste.

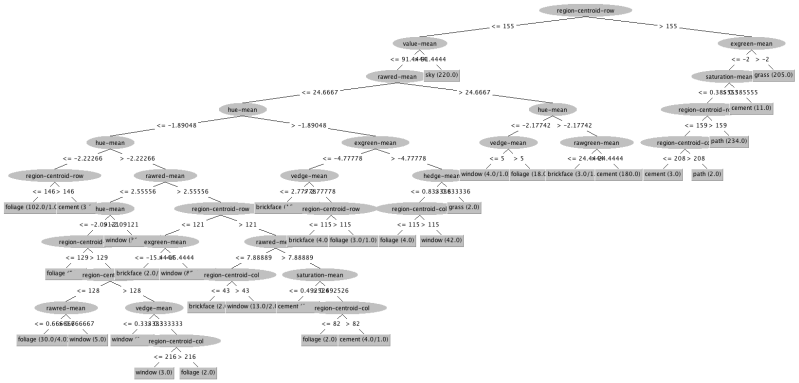

Figura 5: Árvore obtida com o melhor fator de confiança (0.2).

De facto, nos conjunto de dados de teste, verifica-se um pico na percentagem de instâncias classificadas corretamente quando o fator de confiança é 0.2 (96.667%), após o qual os restantes resultados obtidos demonstram efeitos de *overtraining*, através do decréscimo na precisão de classificação das instâncias. O melhor caso, apresenta uma percentagem de 3.333% de instâncias classificadas incorretamente, numa árvore com 63 nós e 32 folhas.

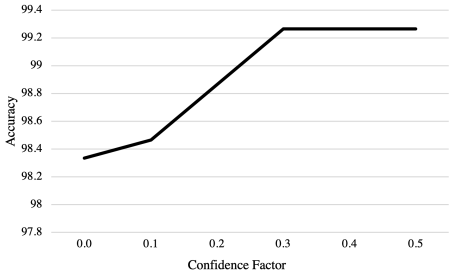

Figura 6: Resultados obtidos com a variação do fator de confiança para os dados de treino.

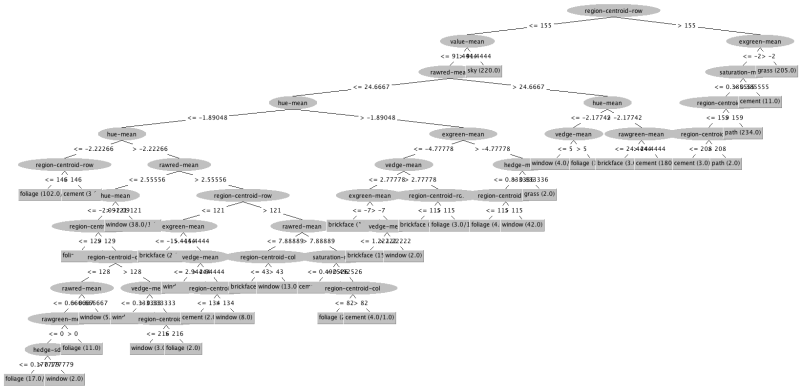

Figura 7: Árvore obtida com o melhor fator de confiança (0.3).

No conjunto de dados de treino, verifica-se um pico a partir de um fator de confiança de 0.3 (99.267%). Para além disso, verifica-se que a técnica de *pruning* não é vantajosa no conjunto de dados de treino uma vez que generaliza a árvore para casos de teste. De facto, a percentagem de instâncias corretamente classificadas numa árvore *unpruned* estabeleceu-se nos 99.333%. Valor ligeiramente superior ao obtido com *pruning* e melhor fator de confiança.

No **terceiro caso**, vamos aplicar a técnica de *subtree replacement* [2]. Esta técnica consiste em substituir o nó pai pela sua subárvore, reduzindo o seu tamanho. Ao correr o algoritmo, obtemos os seguintes resultados para os dados de teste:

```
Correctly Classified Instances         779               96.1728 %
Incorrectly Classified Instances        31                3.8272 %
Kappa statistic                          0.9553
Mean absolute error                      0.0127
Root mean squared error                  0.1005
Relative absolute error                  5.1771 %
Root relative squared error             28.6807 %
Total Number of Instances              810
```

Figura 8: Resultados obtidos através da técnica de *subtree replacement* para os dados de teste.

```
Correctly Classified Instances        1485               99      %
Incorrectly Classified Instances        15                1      %
Kappa statistic                          0.9883
Mean absolute error                      0.0048
Root mean squared error                  0.0488
Relative absolute error                  1.9473 %
Root relative squared error             13.9545 %
Total Number of Instances             1500
```

Figura 9: Resultados obtidos através da técnica de *subtree replacement* para os dados de treino.

Comparando estes três casos, podemos concluir que, para os dados de teste, o caso com

mais precisão na classificação das instâncias dos dados de teste foi no 2º caso, onde aplicamos a técnica de *subtree raising* variando o fator de confiança, sendo o fator de confiança ideial de 0.2. Isto deve-se ao facto da técnica de *subtree raising* combater o *overfitting* e permitir uma precisão na classificação de instâncias bastante superior a uma árvore *unpruned*. Esta técnica de *pruning* conjugada com a variação do fator de confiança ideal para este caso, permitiu obter uma precisão próxima do máximo possível para este caso.

Concluimos também que não podemos retirar muitas conclusões dos dados de treino, visto ser enganador. É um método onde o classificador funciona decorando os dados, o que acaba por o limitar quando colocado com instâncias diferentes das existentes no *dataset* de treino. Generalizar através dos dados de treino acaba por nos fornecer um classificador com uma aplicabilidade mais ampla, que terá de ser testado com um conjunto de dados de teste independente.

Exercício 2

No segundo exercício é pedido para analisar o *dataset* ***glass*** usando validação cruzada com 10 *folds* (provado ser estatisticamente satisfatório na avaliação da *performance* de um classificador [3]) e com três algoritmos diferentes: *NaiveBayes*, Redes *Bayesianas* e *J48*.

NaiveBayes

O classificador *NaiveBayes* baseia-se no Teorema de Bayes, assumindo forte independência entre atributos. Os resultados obtidos foram os seguintes:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.729	0.424	0.455	0.729	0.560	0.286	0.708	0.459	build wind float
0.171	0.101	0.481	0.171	0.252	0.100	0.717	0.506	build wind non-float
0.235	0.086	0.190	0.235	0.211	0.135	0.723	0.162	vehic wind float
?	0.000	?	?	?	?	?	?	vehic wind non-float
0.308	0.040	0.333	0.308	0.320	0.278	0.841	0.347	containers
0.889	0.029	0.571	0.889	0.696	0.698	0.985	0.720	tableware
0.828	0.022	0.857	0.828	0.842	0.818	0.928	0.789	headlamps
0.486	0.188	0.496	0.486	0.453	0.297	0.762	0.501	

Figura 10: Resultados para *NaiveBayes*.

a	b	c	d	e	f	g	<-- classified as
51	5	11	0	0	2	1	a = build wind float
48	13	6	0	5	3	1	b = build wind non-float
12	0	4	0	0	1	0	c = vehic wind float
0	0	0	0	0	0	0	d = vehic wind non-float
0	8	0	0	4	0	1	e = containers
0	0	0	0	0	8	1	f = tableware
1	1	0	0	3	0	24	g = headlamps

Figura 11: Matriz de confusão.

Devemos notar que no caso da classe *"Vehic wind non-float"* que se encontra com pontos de interrogação. Isso deve-se ao facto de o algoritmo não ter classificado nenhuma instância como sendo dessa classe, nem nenhuma instância ser de facto dessa classe também.

Redes Bayesianas

O classificador de Redes *Bayesianas* baseia-se também no Teorema de Bayes, mas modela as relações entre atributos de uma maneira mais generalizada do que o *NaiveBayes*. Os resultados obtidos foram os seguintes:

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.886	0.229	0.653	0.886	0.752	0.620	0.885	0.765	build wind float
0.605	0.101	0.767	0.605	0.676	0.537	0.836	0.759	build wind non-float
0.059	0.015	0.250	0.059	0.095	0.087	0.753	0.195	vehic wind float
?	0.005	0.000	?	?	?	?	?	vehic wind non-float
0.692	0.015	0.750	0.692	0.720	0.703	0.949	0.636	containers
0.778	0.029	0.538	0.778	0.636	0.629	0.993	0.908	tableware
0.897	0.016	0.897	0.897	0.897	0.880	0.965	0.934	headlamps
0.706	0.117	0.695	0.706	0.686	0.589	0.876	0.738	

Figura 12: Resultados para Redes *Bayesianas*.

```
 a  b  c  d  e  f  g   <-- classified as
62  5  2  0  0  1  0 |  a = build wind float
21 46  1  0  3  4  1 |  b = build wind non-float
11  4  1  0  0  1  0 |  c = vehic wind float
 0  0  0  0  0  0  0 |  d = vehic wind non-float
 0  3  0  0  9  0  1 |  e = containers
 0  1  0  0  0  7  1 |  f = tableware
 1  1  0  1  0  0 26 |  g = headlamps
```

Figura 13: Matriz de confusão.

O problema dos pontos de interrogação surge novamente neste classificador, mas desta vez uma instância foi classificada como "d", mas nenhuma instância era verdadeiramente "d".

J48

O classificador *J48* baseia-se em árvores de decisão, através da escolha dos melhores atributos para cada subárvore correspondente.

TP Rate	FP Rate	Precision	Recall	F-Measure	MCC	ROC Area	PRC Area	Class
0.714	0.174	0.667	0.714	0.690	0.532	0.806	0.667	build wind float
0.618	0.181	0.653	0.618	0.635	0.443	0.768	0.606	build wind non-float
0.353	0.046	0.400	0.353	0.375	0.325	0.766	0.251	vehic wind float
?	0.000	?	?	?	?	?	?	vehic wind non-float
0.769	0.010	0.833	0.769	0.800	0.788	0.872	0.575	containers
0.778	0.029	0.538	0.778	0.636	0.629	0.930	0.527	tableware
0.793	0.022	0.852	0.793	0.821	0.795	0.869	0.738	headlamps
0.668	0.130	0.670	0.668	0.668	0.539	0.807	0.611	

Figura 14: Resultados para Redes *Bayesianas*.

```
 a  b  c  d  e  f  g   <-- classified as
50 15  3  0  0  1  1 |  a = build wind float
16 47  6  0  2  3  2 |  b = build wind non-float
 5  5  6  0  0  1  0 |  c = vehic wind float
 0  0  0  0  0  0  0 |  d = vehic wind non-float
 0  2  0  0 10  0  1 |  e = containers
 1  1  0  0  0  7  0 |  f = tableware
 3  2  0  0  0  1 23 |  g = headlamps
```

Figura 15: Matriz de confusão.

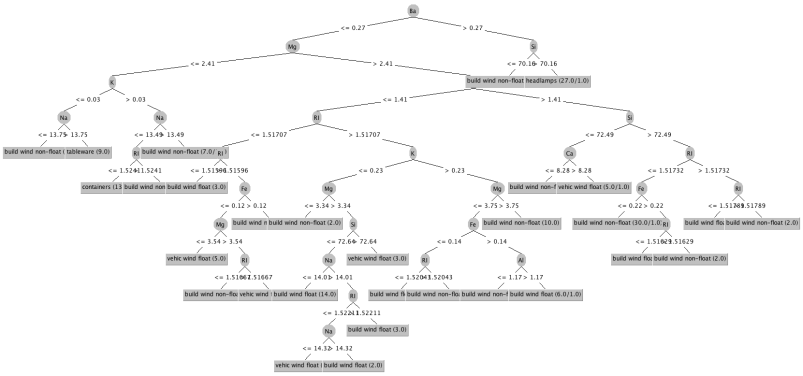

Figura 16: Árvore obtida.

O problema dos pontos de interrogação surge novamente neste classificador, tal como surgiu em *NaiveBayes*.

Tendo agora classificado o *dataset* com os três classificadores pedidos, podemos comparar os resultados. De facto, aquele que apresenta resultados mais satisfatórios, no que toca à correta classificação das instâncias nas respetivas classes, é o algoritmo de Redes *Bayesianas*, pois apresenta uma maior percentagem de instâncias classificadas corretamente, apresenta um maior *rate* de *true positives* e menor de *false positives* e apresenta também uma maior precisão. Uma análise cuidada à Matriz de confusão, permite perceber também que é com esse classificador que há um maior acerto na classificação das instâncias.

O classificador *J48* apresenta também melhores resultados do que *NaiveBayes* para este *dataset*.

De facto, não há um modo de justificar o porquê de um classificador ser melhor que o outro, uma vez que a *performance* de cada classificador varia de acordo com o *dataset*. Neste caso, ficou estabelecido que o melhor classificador de instâncias seriam as Redes *Bayesianas*.

Exercício 3

Neste exercício era pedido para identificar um empate no erro de uma classe. Após efetuar a contagem dos erros totais (na matriz de confusão), apresentados por todas as classes nos três classificadores utilizados, descobrimos um empate em número de erros na classe "*Tableware*"entre os classificadores *J48* e Redes *Bayesianas*, sendo 2 o número de erros total.

```
=== BayesNet ===
     a  b  c  d  e  f  g   <-- classified as
     0  1  0  0  0  7  1 |  f = tableware

=== J48 ===
     a  b  c  d  e  f  g   <-- classified as
     1  1  0  0  0  7  0 |  f = tableware
```

Figura 17: Excerto das matrizes de confusão para os classificadores na classe *"Tableware"*.

No entanto, tal como pede o exercício, conseguimos verificar um fator de desempate na variável AUC (*Area Under Curve*), representada na tabela de precisão como *"ROC Area"*. Esta área representa a probabilidade de um classificador colocar um exemplo positivo mais alto do que um exemplo negativo. Logo, quanto mais alta for esta variável, melhor.

```
=== BayesNet ===
     ROC Area  Class
     0.993     tableware

=== J48 ===
     ROC Area  Class
     0.930     tableware
```

Figura 18: Excerto das tabelas de precisão para os valores da AUC da classe *"Tableware"*.

Como podemos observar pelas tabelas de precisão de ambos os classificadores analisados e para a classe *"Tableware"*, com Redes *Bayesianas* obtemos um AUC superior do que com *J48*, logo este classificador revela-se melhor no que toca à correta classificação de instâncias da classe *"Tableware"* apesar do empate verificado nas matrizes de confusão.

Conclusão

Este mini-teste ajudou a perceber que a precisão de cada classificador em classificar corretamente as instâncias pode variar bastante de acordo com o conjunto de dados fornecido, não havendo um classificador ideal, mas sim um adequado a cada conjunto de dados. Serviu também para perceber que a utilização da técnica de *pruning* e a variação de parâmetros como o fator de confiança permitem aumentar a precisão do classificador. De um modo geral, a realização deste mini-teste foi positiva para consolidar e aprofundar os conhecimentos absorvidos nas aulas da UC.

Referências

[1] Sam Drazin and Matt Montag. *Decision Tree Analysis using Weka*. Universidade de Miami.

[2] *Details of J48 Pruning Parameters* [online]. WEKA Forum, [citado a 15/10/2018] (http://weka.8497.n7.nabble.com/Details-of-J48-pruning-parameters-td42456.html).

[3] lan Witten, Eibe Frank. *Data Mining – Practical Machine Learning Tools and Techniques*, 2ª edição, Elsevier, 2005.

[4] Paulo Azevedo. *Slides de Mineração de Dados*. Universidade do Minho.