Mini-teste 2 **Vitor Peixoto - A79175**
Mineração de Dados 10/11/2018
Mestrado em Engenharia Informática Universidade do Minho

Introdução

Neste mini-teste foi atribuído um conjunto de quatro questões baseados na análise de conjuntos de dados. Essa análise é feita pela 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* ***labor***, com o algoritmo de classificação *J48* recorrendo a validação cruzada de 10 *folds*.

Este método consiste em 10 divisões das instâncias do *dataset*, sendo que a cada iteração, uma parte será usada como dados de treino e as restantes como dados de teste. Este procedimento é repetido para cada parte dividida, sendo que cada porção do *dataset* são dados de treino uma vez e dados de teste 9 (k-1) vezes.

Os resultados obtidos com estes parâmetros revelaram uma percentagem de cerca de 73.68% de instâncias corretamente classificadas e de 0.695 de AUC (*Area under curve*).

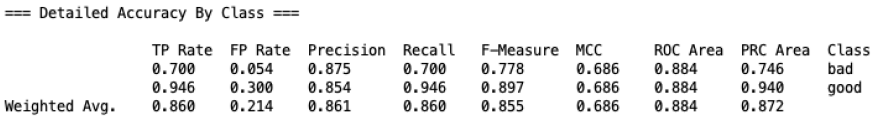
```
Correctly Classified Instances          42               73.6842 %
Incorrectly Classified Instances        15               26.3158 %
Kappa statistic                          0.4415
Mean absolute error                      0.3192
Root mean squared error                  0.4669
Relative absolute error                 69.7715 %
Root relative squared error             97.7888 %
Total Number of Instances               57


=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure  MCC      ROC Area  PRC Area  Class
               0.700     0.243     0.609      0.700     0.651      0.444     0.695     0.559     bad
               0.757     0.300     0.824      0.757     0.789      0.444     0.695     0.738     good
Weighted Avg.  0.737     0.280     0.748      0.737     0.740      0.444     0.695     0.675
```

Figura 1: Resultados obtidos através de *J48* com validação cruzada.

Este exercício pedia também para correr o mesmo algoritmo melhorado com o método de *Bagging*. *Bagging* é uma das técnicas de composição de modelos onde um conjunto de classificadores são combinados para obter um classificador mais forte e com melhores resultados.

Os resultados obtidos com a composição de *Bagging* sobre *J48* foram muito superiores aos obtidos apenas com *J48*. Obtivemos uma percentagem perto dos 86% de instâncias corretamente classificadas e um AUC de 0.884.

```
Correctly Classified Instances          49               85.9649 %
Incorrectly Classified Instances         8               14.0351 %
Kappa statistic                          0.6771
Mean absolute error                      0.2588
Root mean squared error                  0.3533
Relative absolute error                 56.5714 %
Root relative squared error             73.985  %
Total Number of Instances               57
```



```
=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.700    0.054    0.875      0.700    0.778      0.686    0.884     0.746     bad
                 0.946    0.300    0.854      0.946    0.897      0.686    0.884     0.940     good
Weighted Avg.    0.860    0.214    0.861      0.860    0.855      0.686    0.884     0.872
```

Figura 2: Resultados obtidos através de *Bagging* sobre *J48* com validação cruzada.

No exemplo do *Netflix Prize* apresentado na aula, a união entre as equipas e os seus classificadores permitiu um *score* superior. Esse caso verifica-se também na composição de modelos para este *dataset* onde a sobreposição entre *Bagging* e *J48* foi benéfico na classificação das instâncias e no valor da AUC. Esta melhoria deve-se ao facto de o *Bagging* provocar uma melhoria em algoritmos instáveis, como é o caso do *J48*, uma vez que enriquece o conjunto de dados.

Exercício 2

Este exercício pede para analisar e especificar os benefícios que a aplicação de *Bagging* traria num *dataset* onde foi aplicado o classificador de *Naive Bayes* com uma taxa de erro de 0.035.

A técnica de *Bagging* reduz a variância do erro do classificador. Isto funciona para algoritmos instáveis, como é o caso de algoritmos que envolvem árvores de decisão (i.e. *J48*). No entanto, para algoritmos mais estáveis, como é o caso de *Naive Bayes*, esta técnica pode mesmo aumentar a variância, uma vez que este algoritmo apresenta uma variância por si só já baixa.

Tendo então em conta a já insignificativa taxa de erro apresentada pelo algoritmo *Naive Bayes unbagged*, pode-se assumir que a aplicação de *Bagging* neste caso poderá não ser benéfica e aumentar a taxa de erro nas instâncias corretamente classificadas.

Exercício 3

Neste exercício era pedido um estudo sobre o custo de erros no *dataset* ***vote*** usando os classificadores *J48* e *Naive Bayes*.

Antes de iniciarmos o estudo, temos de entender que há duas maneiras de tornar um classificador sensível ao custo: por *Classification* ou por *Learning*. O método ***Classification*** usa um classificador *standard*, ajustando o *output* à matriz de custo. O método ***Learning*** aprende um novo classificador ótimo, reajustando os exemplos mal classificados por duplicação de exemplos ou por reajustamento dos pesos.

Para conseguir estudar o custo dos erros, foram definidas 3 matrizes de custo:

$$M1 = \begin{bmatrix} 0 & 2 \\ 1 & 0 \end{bmatrix}$$

$$M2 = \begin{bmatrix} 0 & 1 \\ 2 & 0 \end{bmatrix}$$

$$M3 = \begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix}$$

Vamos agora passar ao estudo dos custos neste *dataset*:

Usamos o classificador ***J48*** aplicado num modelo sensível ao custo por ***Classification***. Obtivemos os seguintes resultados para as respetivas matrizes de custo:

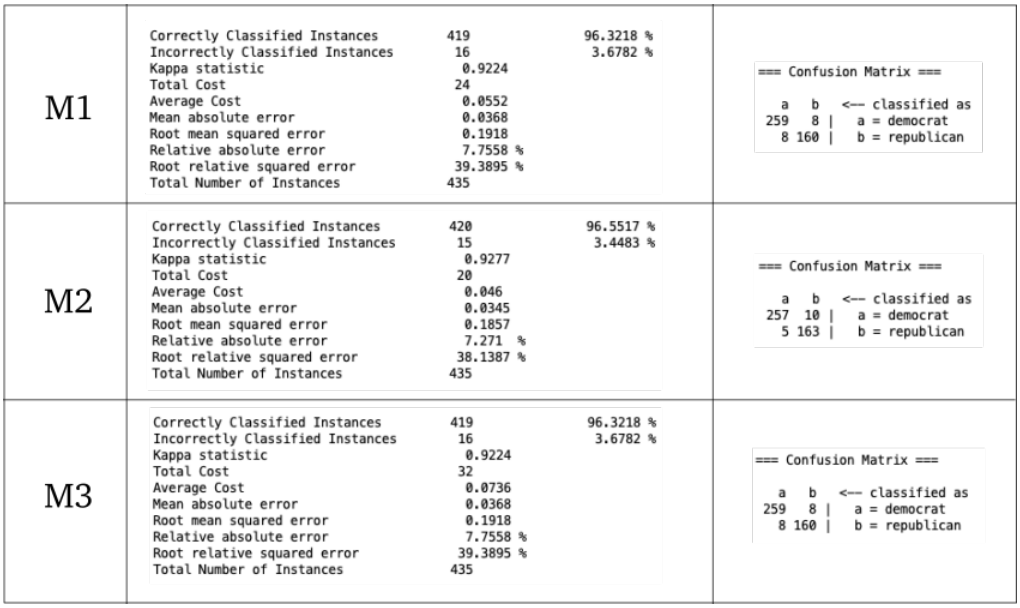


Figura 3: Resultados obtidos através de *J48* com *Cost-Sensitive Classification* aplicado nas 3 matrizes.

Apesar de o algoritmo aplicado sobre as três matrizes ser exatamente o mesmo, o número de instâncias corretamente classificadas em M2 é diferente, pois a própria matriz de custo influencia a aprendizagem do classificador, podendo afetar a sua performance. Assim sendo, neste caso, para as matrizes M1, M2 e M3, obtivemos um custo de 24, 20 e 32, respetivamente.

Usamos agora o mesmo classificador ***J48*** mas com um modelo *Cost-Sensitive* por ***Learning***. Os resultados obtidos foram os seguintes:

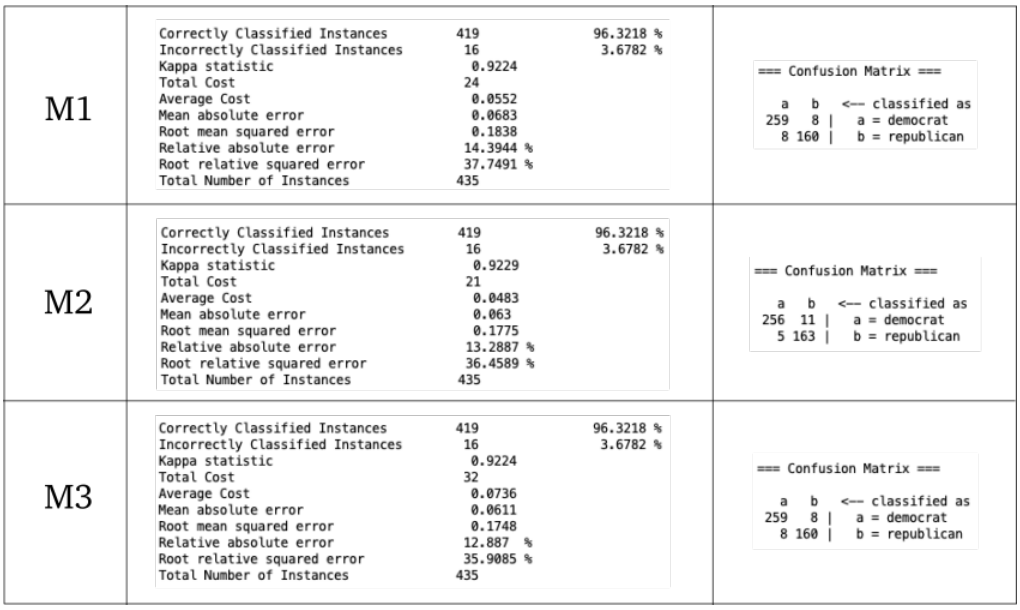


Figura 4: Resultados obtidos através de *J48* com *Cost-Sensitive Learning* aplicado nas 3 matrizes.

Neste caso, obtivemos um custo de 24, 21 e 32 para cada uma das matrizes de custo. Estes resultados obtidos com *Learning* foram muito semelhantes aos obtidos com *Classification*.

Passamos agora para ***Naive Bayes*** aplicado num modelo sensível ao custo por ***Classification***. Os resultados obtidos foram os seguintes:

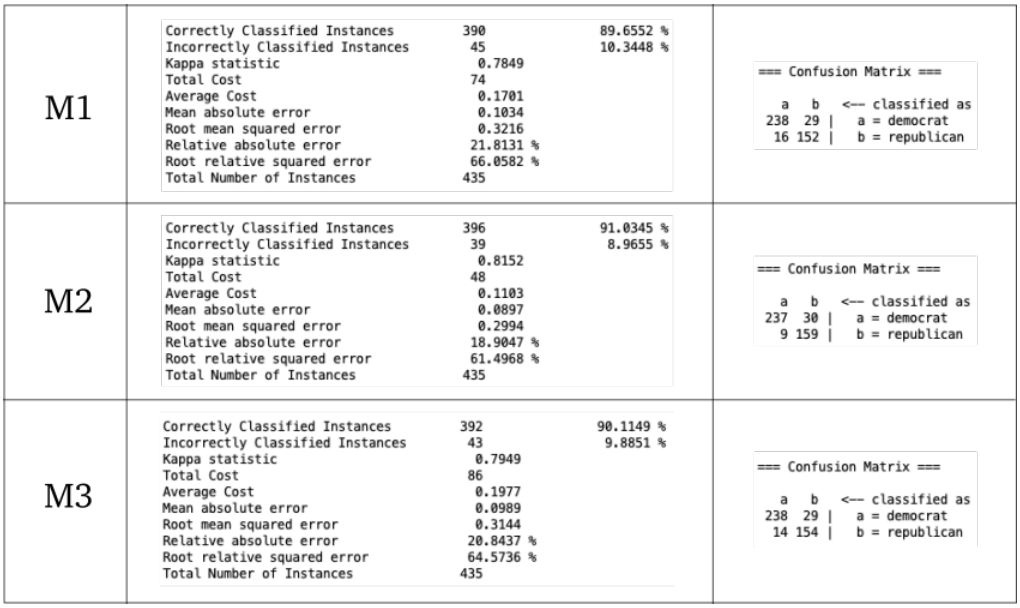


Figura 5: Resultados obtidos através de *Naive Bayes* com *Cost-Sensitive Classification* aplicado nas 3 matrizes.

Neste caso, obtivemos um custo de 74, 48 e 86 para cada uma das matrizes de custo M1, M2 e M3, respetivamente.

Usamos novamente o classificador ***Naive Bayes*** aplicado agora em num modelo sensível ao custo por ***Learning***. Os resultados obtidos foram os seguintes:

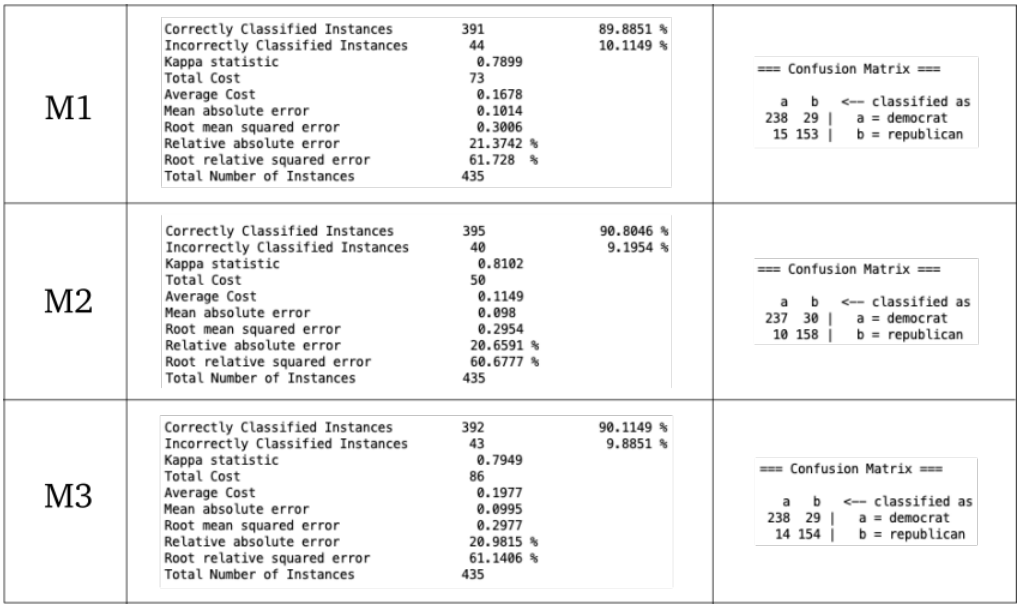


Figura 6: Resultados obtidos através de *Naive Bayes* com *Cost-Sensitive Learning* aplicado nas 3 matrizes.

Neste caso, obtivemos um custo de 73, 50 e 86 para cada uma das matrizes de custo M1, M2 e M3, respetivamente. Os resultados obtidos neste caso, foram novamente semelhantes entre *Learning* e *Classification*.

No cômputo geral, os custos obtidos com *J48* foram melhores aos obtidos com *Naive Bayes*. Isto deve-se às matrizes de confusão obtidas com NB terem mais instâncias como FP e FN. Multiplicando esses valores pelo respetivo valor na matriz de custo, obtém-se o custo total dos

erros, tendo neste caso o custo sido maior com o algoritmo NB.

Exercício 4

Este exercício pedia para considerar o *dataset* ***unbalanced*** e comparar o desempenho dos algoritmos de *clustering k-means* e *EM*.

Inicialmente corremos o *dataset* com o algoritmo *EM*, ignorando a classe *'outcome'* visto ser a classe do resultado. Quando o número de *clusters* como -1 para este algoritmo, ele irá determinar o número de *clusters*.

Correndo então este algoritmo, ele indica 18 como o número de *clusters* através de validação cruzada. Estes 18 *clusters* devem-se ao facto de o algoritmo dividir as instâncias *'active'* da classe *'outcome'* em vários *clusters* em vez de as por num só, ao passo que as instâncias de *'inactive'* estão bem distribuídas. A distribuição das instâncias do *'outcome'* pode ser vista no seguinte gráfico:

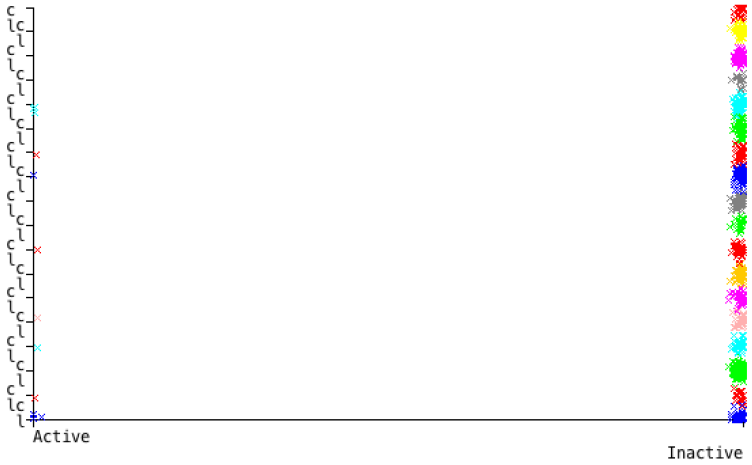


Figura 7: Distribuição de *'outcome'* pelos *clusters*.

A execução do algoritmo *EM* é extremamente lenta (cerca de 48 segundos) dada a necessidade de cálculo do número de *clusters*, verificando-se uma descida significativa (para 0.4 segundos) quando esse número é fornecido.

O algoritmo *EM* foi executado com 2 iterações e apresentou a seguinte colocação das instâncias do *dataset* nos respetivos *clusters*:

Clustered Instances

0	45	(5%)
1	25	(3%)
2	133	(16%)
3	42	(5%)
4	26	(3%)
5	56	(7%)
6	46	(5%)
7	38	(4%)
8	22	(3%)
9	54	(6%)
10	69	(8%)
11	37	(4%)
12	55	(6%)
13	58	(7%)
14	13	(2%)
15	61	(7%)
16	48	(6%)
17	28	(3%)

Figura 8: Distribuição das instâncias pelo algoritmo *EM*.

Tendo agora um número de *clusters* definido, podemos executar também o algoritmo *k-means*. *K-means* é um algoritmo simples, iterativo e baseado nas distâncias entre instâncias (distância euclidiana, neste caso). Difere do algoritmo *EM* na medida em que o *EM* se baseia na probabilidade (expetativa) de um ponto pertencer a um *cluster* específico.

Este algoritmo correu em apenas 0.05 segundos, com 20 iterações, com uma soma dos erros quadrados de cerca de 321.32 e apresentou a seguinte colocação das instâncias do *dataset* nos respetivos *clusters*:

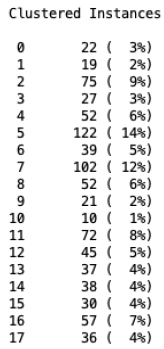


Figura 9: Distribuição das instâncias pelo algoritmo *k-means*.

Conclusão

Com este mini-teste, foi possível tirar algumas conclusões ao longo do trabalho desenvolvido:

- A técnica de *Bagging* é muito eficiente sobre o algoritmo *J48*, conseguindo melhorar significativamente a *performance* deste algoritmo.
- Nem sempre esta técnica é benéfica na classificação de instâncias de um *dataset*. Em algoritmos estáveis, como o *Naive Bayes*, a técnica de *Bagging* pode até diminuir a *performance* do algoritmo.
- O estudo sobre o custo dos erros é essencial para diminuir os casos de FN e FP numa matriz de confusão.
- Há dois métodos de modelos *Cost-sensitive*, podendo ser aplicados sob um algoritmo (*J48*, *Naive Bayes*, etc.) e a sua performance pode variar de acordo com o *dataset*, sendo importante testar todas as possibilidades e escolher a que minimiza o custo dos erros.
- As técnicas de *clustering* são importantes no agrupamento de instâncias similares entre si.

Resumindo, este trabalho assumiu um papel importante e positivo no acompanhamento e aprofundamento das matérias lecionadas nas aulas da Unidade Curricular de Mineração de Dados.