UNIVERSIDADE DO MINHO

Mini-teste 1

Mestrado Integrado em Engenharia Informática

Mineração de Dados
(1.º Semestre / 2020-2021)

A84003 Beatriz Rocha

Braga,
Novembro 2020

1

Considere o *dataset* "CBrasil.csv". Pretende-se elaborar um estudo para o desenvolvimento de um modelo de previsão para analisar as faltas às consultas nos vários centros de saúde do Rio de Janeiro. O objetivo é ter um estudo sobre possíveis modelos pra prever/identificar os pacientes que têm tendência a faltar às consultas marcadas. Temos de pré-processar os dados por forma a:

- **Eliminar atributos redundantes (ou com pouco valor informativo)**

  Em primeiro lugar, comecei por mudar o tipo dos atributos "ScheduledDay"e "AppointmentDay"para "date", com o auxílio de um editor de texto, pois o desempenho do modelo melhorou depois desta alteração. De seguida, discretizei ambas as datas, recorendo ao filtro supervisionado "Discretize"do *Weka* que resultou nos gráficos que podemos ver na Figura 1 e na Figura 2.

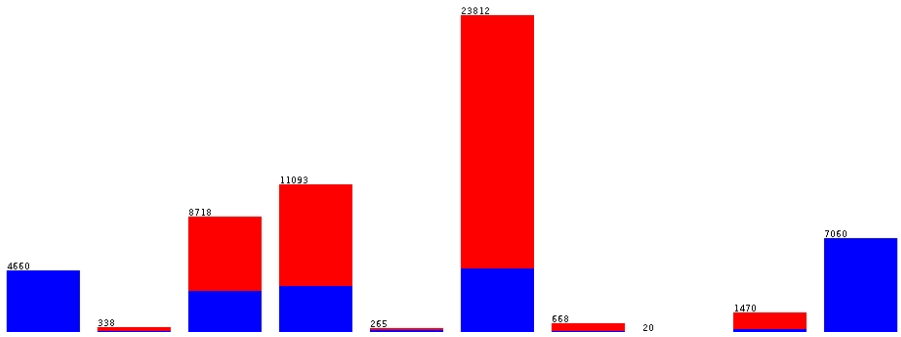

Figura 1: "ScheduledDay"com discretização

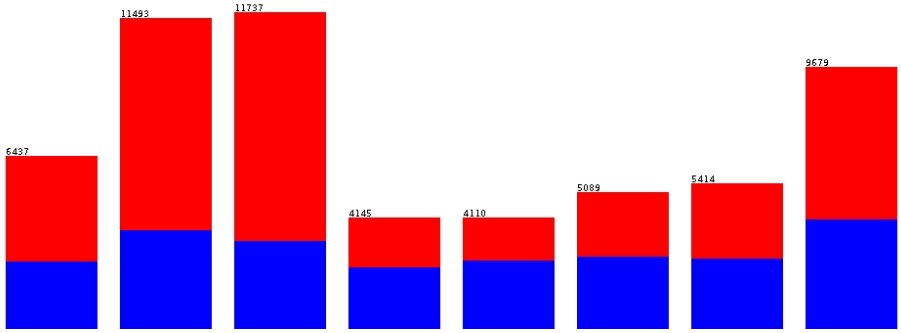

Figura 2: "AppointmentDay"com discretização

  Em segundo lugar, eliminei o "PatientId"(tal como podemos ver na Figura 3 e na Figura 4), pois, do meu ponto de vista, este atributo tem pouco valor informativo, uma vez que o facto de uma pessoa faltar a uma consulta em nada tem que ver com o seu identificador.

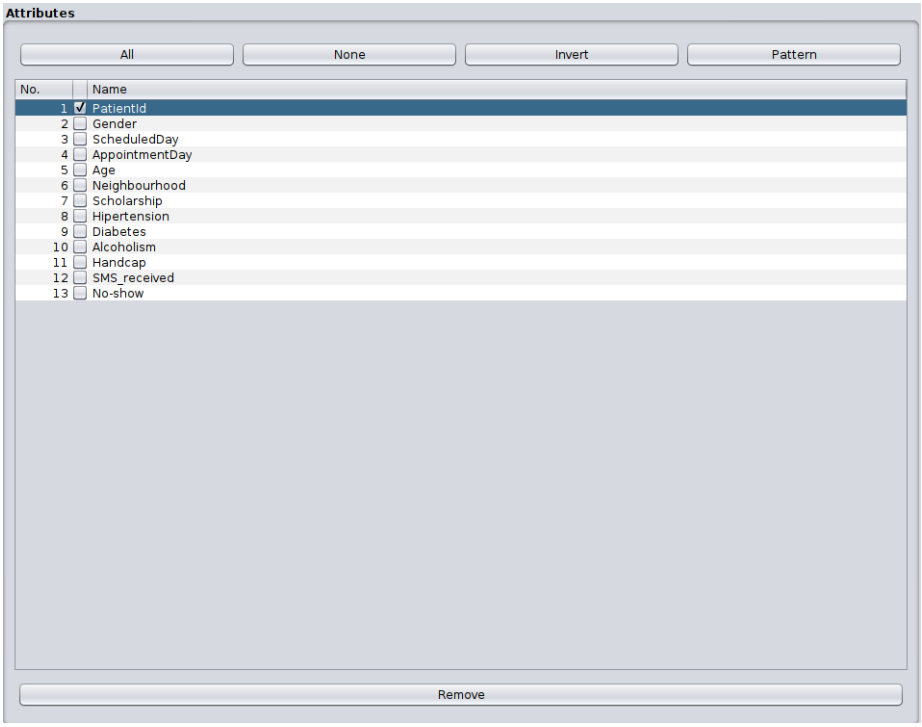

Figura 3: Remoção do atributo "PatientId"

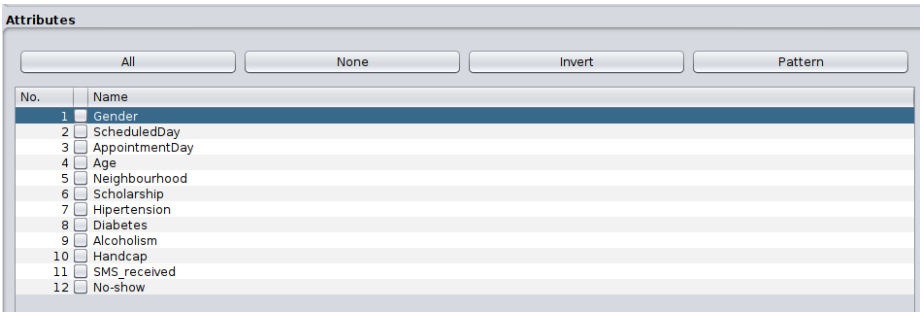

Figura 4: Atributos com valor informativo

- **Forçar atributos booleanos ou categóricos a serem mesmo booleanos ou categóricos (e não interpretar como numéricos como estão nos dados!)**

  Para resolver esta questão, recorri aos filtros "NumericToBinary"e "NumericToNominal"do *Weka*. Comecei por selecionar os atributos numéricos "SMS_received", "Alcoholism", "Diabetes", "Hipertension"e "Scholarship"e apliquei o primeiro filtro aos mesmos. Por fim, selecionei o atributo numérico "Handcap"e apliquei o segundo filtro ao mesmo (uma vez que este apresenta quatro valores possíveis em vez de dois como os restantes).

Depois de atingir um *dataset* com os elementos relevantes apresente

resultados que permitam responder às seguintes questões:

1.1 Quais os atributos a considerar para construir o modelo de previsão? Justifique.

Posto isto, os atributos a considerar são "Gender", "ScheduledDay", "AppointmentDay", "Age", "Neighbourhood", "Scholarship_binarized", "Hipertension_binarized", "Diabetes_binarized", "Alcoholism_binarized", "HandCap", "SMS_received_binarized"e "No-show"(sendo que este último é aquele que queremos prever), visto que, para além de achar que se correlacionam com o facto de uma pessoa faltar a uma consulta, o desempenho do modelo piora se os retirar.

Os seus respetivos tipos apresentam-se em baixo:

- "Gender" - booleano
- "ScheduledDay" - categórico
- "AppointmentDay" - categórico
- "Age" - numérico
- "Neighbourhood" - categórico
- "Scholarship_binarized" - booleano
- "Hipertension_binarized" - booleano
- "Diabetes_binarized" - booleano
- "Alcoholism_binarized" - booleano
- "HandCap" - categórico
- "SMS_received_binarized" - booleano
- "No-show" - booleano

1.2 Qual é o atributo com maior valor informativo?

Nas aulas teóricas foi-nos explicado que o atributo com maior ganho informativo é aquele que deve estar na raíz da árvore. Assim sendo, o atributo com maior valor informativo é "ScheduledDay", visto que a raíz da árvore gerada pelo algoritmo J48 é esse mesmo atributo, tal como podemos ver na Figura 5. Para além disso, a minha resposta também pode ser apoiada na Figura 6, onde se pode observar que a distribuição da classe "No-show"não está equilibrada para este atributo em causa, ao contrário dos restantes atributos.

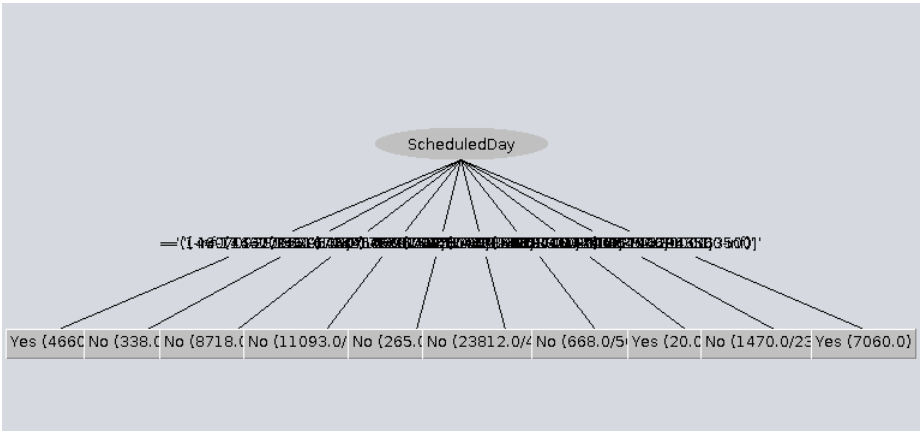

Figura 5: Atributo com maior valor informativo

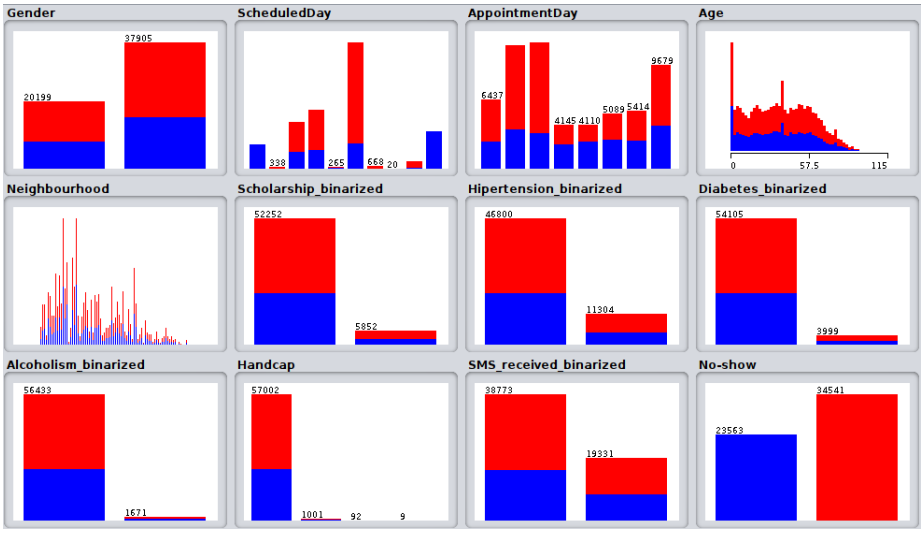

Figura 6: Gráfico com os vários atributos

2

Use o *WEKA* e as suas implementações de *NaiveBayes*, *BayesNet* e J48 para responder às seguintes perguntas (apresente resultados obtidos por validação cruzada e variação de hiper-parâmetros dos três algoritmos):

2.1 Qual o modelo que escolhia para implementar em tempo real dentro destes 3 (e suas variantes)? Justifique.

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

```
Time taken to build model: 0.09 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       45940               79.0651 %
Incorrectly Classified Instances     12164               20.9349 %
Kappa statistic                          0.5323
Mean absolute error                      0.2907
Root mean squared error                  0.391
Relative absolute error                 60.2935 %
Root relative squared error             79.6318 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.516    0.022    0.941      0.516    0.667      0.583    0.788     0.803     Yes
                 0.978    0.484    0.748      0.978    0.847      0.583    0.788     0.791     No
Weighted Avg.    0.791    0.297    0.826      0.791    0.774      0.583    0.788     0.796

=== Confusion Matrix ===

     a     b   <-- classified as
 12166 11397 |     a = Yes
   767 33774 |     b = No
```

Figura 7: NaiveBayes sem discretização

```
Time taken to build model: 0.06 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       45960               79.0995 %
Incorrectly Classified Instances     12144               20.9005 %
Kappa statistic                          0.533
Mean absolute error                      0.2907
Root mean squared error                  0.3909
Relative absolute error                 60.2886 %
Root relative squared error             79.6216 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.517    0.022    0.942      0.517    0.667      0.584    0.788     0.803     Yes
                 0.978    0.483    0.748      0.978    0.848      0.584    0.788     0.791     No
Weighted Avg.    0.791    0.296    0.826      0.791    0.774      0.584    0.788     0.796

=== Confusion Matrix ===

     a     b   <-- classified as
 12172 11391 |     a = Yes
   753 33788 |     b = No
```

Figura 8: NaiveBayes com discretização

O classificador *BayesNet* 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:

```
Time taken to build model: 0.17 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       45957               79.0944 %
Incorrectly Classified Instances     12147               20.9056 %
Kappa statistic                          0.5329
Mean absolute error                      0.2906
Root mean squared error                  0.3909
Relative absolute error                 60.281  %
Root relative squared error             79.6229 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.517    0.022    0.942      0.517    0.667      0.584    0.788     0.803     Yes
                 0.978    0.483    0.748      0.978    0.848      0.584    0.788     0.791     No
Weighted Avg.    0.791    0.296    0.826      0.791    0.774      0.584    0.788     0.796

=== Confusion Matrix ===

     a     b   <-- classified as
 12172 11391 |     a = Yes
   756 33785 |     b = No
```

Figura 9: *BayesNet* com o algoritmo de pesquisa K2

```
Time taken to build model: 0.28 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       45958               79.0961 %
Incorrectly Classified Instances     12146               20.9039 %
Kappa statistic                          0.5329
Mean absolute error                      0.2906
Root mean squared error                  0.3908
Relative absolute error                 60.2676 %
Root relative squared error             79.5909 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.516    0.022    0.942      0.516    0.667      0.584    0.788     0.803     Yes
                 0.978    0.484    0.748      0.978    0.848      0.584    0.788     0.791     No
Weighted Avg.    0.791    0.296    0.827      0.791    0.774      0.584    0.788     0.796

=== Confusion Matrix ===

     a     b   <-- classified as
 12161 11402 |     a = Yes
   744 33797 |     b = No
```

Figura 10: *BayesNet* com o algoritmo de pesquisa *HillClimber*

```
Time taken to build model: 15.14 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       46250               79.5987 %
Incorrectly Classified Instances     11854               20.4013 %
Kappa statistic                          0.5406
Mean absolute error                      0.2953
Root mean squared error                  0.3844
Relative absolute error                 61.2385 %
Root relative squared error             78.2845 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.500    0.002    0.993      0.500    0.665      0.607    0.797     0.810     Yes
                 0.998    0.500    0.745      0.998    0.853      0.607    0.797     0.800     No
Weighted Avg.    0.796    0.298    0.846      0.796    0.777      0.607    0.797     0.804

=== Confusion Matrix ===

     a     b   <-- classified as
 11787 11776 |     a = Yes
    78 34463 |     b = No
```

Figura 11: *BayesNet* com o algoritmo de pesquisa *SimulatedAnnealing*

O classificador J48 baseia-se em árvores de decisão, através da escolha dos melhores atributos para cada subárvore correspondente. Os resultados obtidos foram os seguintes:

```
Time taken to build model: 1.77 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       46260               79.6159 %
Incorrectly Classified Instances     11844               20.3841 %
Kappa statistic                          0.5407
Mean absolute error                      0.295
Root mean squared error                  0.3841
Relative absolute error                 61.1752 %
Root relative squared error             78.2229 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.498    0.001    0.998      0.498    0.665      0.608    0.798     0.806     Yes
                 0.999    0.502    0.745      0.999    0.854      0.608    0.798     0.799     No
Weighted Avg.    0.796    0.298    0.847      0.796    0.777      0.608    0.798     0.802

=== Confusion Matrix ===

     a     b   <-- classified as
 11746 11817 |     a = Yes
    27 34514 |     b = No
```

Figura 12: J48 com *subtree raising*, com *pruning* e sem *Laplace smoothing*

```
Time taken to build model: 1.5 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       43813               75.4044 %
Incorrectly Classified Instances     14291               24.5956 %
Kappa statistic                          0.4706
Mean absolute error                      0.2944
Root mean squared error                  0.4318
Relative absolute error                 61.0602 %
Root relative squared error             87.942  %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.582    0.129    0.755      0.582    0.658      0.480    0.758     0.740     Yes
                 0.871    0.418    0.754      0.871    0.808      0.480    0.758     0.753     No
Weighted Avg.    0.754    0.301    0.754      0.754    0.747      0.480    0.758     0.748

=== Confusion Matrix ===

     a     b   <-- classified as
 13723  9840 |     a = Yes
  4451 30090 |     b = No
```

Figura 13: J48 com *subtree raising*, sem *pruning* e sem *Laplace smoothing*

```
Time taken to build model: 1.66 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       46262               79.6193 %
Incorrectly Classified Instances     11842               20.3807 %
Kappa statistic                          0.5407
Mean absolute error                      0.295
Root mean squared error                  0.3841
Relative absolute error                 61.1737 %
Root relative squared error             78.2196 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.498    0.001    0.998      0.498    0.665      0.608    0.798     0.806     Yes
                 0.999    0.502    0.745      0.999    0.854      0.608    0.798     0.799     No
Weighted Avg.    0.796    0.298    0.848      0.796    0.777      0.608    0.798     0.802

=== Confusion Matrix ===

     a     b   <-- classified as
 11742 11821 |     a = Yes
    21 34520 |     b = No
```

Figura 14: J48 sem *subtree raising*, com *pruning* e sem *Laplace smoothing*

```
Time taken to build model: 1.61 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       46260               79.6159 %
Incorrectly Classified Instances     11844               20.3841 %
Kappa statistic                          0.5407
Mean absolute error                      0.295
Root mean squared error                  0.3841
Relative absolute error                 61.1925 %
Root relative squared error             78.2224 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.498    0.001    0.998      0.498    0.665      0.608    0.798     0.806     Yes
                 0.999    0.502    0.745      0.999    0.854      0.608    0.798     0.799     No
Weighted Avg.    0.796    0.298    0.847      0.796    0.777      0.608    0.798     0.802

=== Confusion Matrix ===

     a     b   <-- classified as
 11746 11817 |     a = Yes
    27 34514 |     b = No
```

Figura 15: J48 com *subtree raising*, com *pruning* e com *Laplace smoothing*

```
Time taken to build model: 1.63 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       43813               75.4044 %
Incorrectly Classified Instances     14291               24.5956 %
Kappa statistic                          0.4706
Mean absolute error                      0.3064
Root mean squared error                  0.4099
Relative absolute error                 63.549  %
Root relative squared error             83.4904 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.582    0.129    0.755      0.582    0.658      0.480    0.773     0.792     Yes
                 0.871    0.418    0.754      0.871    0.808      0.480    0.773     0.770     No
Weighted Avg.    0.754    0.301    0.754      0.754    0.747      0.480    0.773     0.779

=== Confusion Matrix ===

     a     b   <-- classified as
 13723  9840 |     a = Yes
  4451 30090 |     b = No
```

Figura 16: J48 com *subtree raising*, sem *pruning* e com *Laplace smoothing*

```
Time taken to build model: 1.59 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances       46262               79.6193 %
Incorrectly Classified Instances     11842               20.3807 %
Kappa statistic                          0.5407
Mean absolute error                      0.295
Root mean squared error                  0.3841
Relative absolute error                 61.1912 %
Root relative squared error             78.2195 %
Total Number of Instances            58104

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.498    0.001    0.998      0.498    0.665      0.608    0.798     0.806     Yes
               0.999    0.502    0.745      0.999    0.854      0.608    0.798     0.799     No
Weighted Avg.  0.796    0.298    0.848      0.796    0.777      0.608    0.798     0.802

=== Confusion Matrix ===

     a     b   <-- classified as
 11742 11821 |     a = Yes
    21 34520 |     b = No
```

Figura 17: J48 sem *subtree raising*, com *pruning* e com *Laplace smoothing*

Modelo	**Tempo**
NaiveBayes sem discretização	0.09
NaiveBayes com discretização	0.06
BayesNet com o algoritmo de pesquisa K2	0.17
BayesNet com o algoritmo de pesquisa *HillClimber*	0.28
BayesNet com o algoritmo de pesquisa *SimulatedAnnealing*	15.14
J48 com *subtree raising*, com *pruning* e sem *Laplace smoothing*	1.77
J48 com *subtree raising*, sem *pruning* e sem *Laplace smoothing*	1.5
J48 sem *subtree raising*, com *pruning* e sem *Laplace smoothing*	1.66
J48 com *subtree raising*, com *pruning* e com *Laplace smoothing*	1.61
J48 com *subtree raising*, sem *pruning* e com *Laplace smoothing*	1.63
J48 sem *subtree raising*, com *pruning* e com *Laplace smoothing*	1.59

Tabela 1: Tabela comparativa dos tempos

Posto isto, concluo que o melhor modelo para implementar em tempo real é o *NaiveBayes* com discretização, uma vez que é aquele que apresenta a melhor relação tempo para construir o modelo/instâncias corretamente classificadas.

2.2 Em termos de classe "No-show"=yes qual o melhor modelo? Justifique.

Em primeiro lugar, é importante explicar como é que as medidas "Precision", "Recall"e "F-Measure"são calculadas.

A "F-Measure"é calculada recorrendo às medidas "Precision"e "Recall"segundo as fórmulas que se seguem:

"Precision"= t_p / (t_p + f_p)

"Recall"= t_p / (t_p + f_n)
"F-Measure"= 2 * "Precision"* "Recall"/ ("Precision"+ "Recall")

onde t_p é o número de verdadeiros positivos, f_p o número de falsos positivos e f_n o número de falsos negativos.

Em segundo lugar, é importante explicar o que são e para que servem as curvas *ROC*. A análise de curvas *ROC* investiga a relação entre a proporção de positivos e negativos corretamente classificados e, através das mesmas, podemos estudar comportamentos de modelos em relação a classes.

Modelo	**Precision**	**Recall**	**F-Measure**	**ROC Area**
NaiveBayes sem discretização	0.941	0.516	0.667	0.788
NaiveBayes com discretização	0.942	0.517	0.667	0.788
BayesNet com o algoritmo de pesquisa K2	0.942	0.517	0.667	0.788
BayesNet com o algoritmo de pesquisa *HillClimber*	0.942	0.516	0.667	0.788
BayesNet com o algoritmo de pesquisa *SimulatedAnnealing*	0.993	0.500	0.665	0.797
J48 com *subtree raising*, com *pruning* e sem *Laplace smoothing*	0.998	0.498	0.665	0.798
J48 com *subtree raising*, sem *pruning* e sem *Laplace smoothing*	0.755	0.582	0.658	0.758
J48 sem *subtree raising*, com *pruning* e sem *Laplace smoothing*	0.998	0.498	0.665	0.798
J48 com *subtree raising*, com *pruning* e com *Laplace smoothing*	0.998	0.498	0.665	0.798
J48 com *subtree raising*, sem *pruning* e com *Laplace smoothing*	0.755	0.582	0.658	0.773
J48 sem *subtree raising*, com *pruning* e com *Laplace smoothing*	0.998	0.498	0.665	0.798

Tabela 2: Tabela comparativa das medidas de desempenho

Visto que estas medidas não são suficientes para escolher o modelo com o melhor desempenho, tive de recorrer novamente à Tabela 1, onde constatei que o melhor modelo em termos de classe "No-show"=yes é J48 sem *subtree raising*, com *pruning* e com *Laplace smoothing*.

2.3 Para o modelo derivado do algoritmo J48 mostre as várias árvores possíveis de obter por diferentes configurações de *pruning*. Tente explicar os vários desempenhos (dos vários modelos derivados).

Pruning consiste na simplificação das árvores por forma a evitar árvores cobertas por poucos casos. Existem duas possíveis estratégias:

1. *Pre-pruning*: parar de expandir um ramo quando a informação se torna pouco fiável (sem significância). Pode parar o processo demasiado cedo e derivar em *underfitting*;

2. *Post-pruning*: deixar crescer a árvore até ao fim. Depois podar as subárvores pouco fiáveis. É mais fácil implementar esta estratégia e é a que melhores resultados origina na prática.

Em primeiro lugar, corri o algoritmo J48 sem *pruning*. Esta escolha de hiperparâmetros resultou nos valores da Figura 13 e na árvore que se apresenta de seguida:

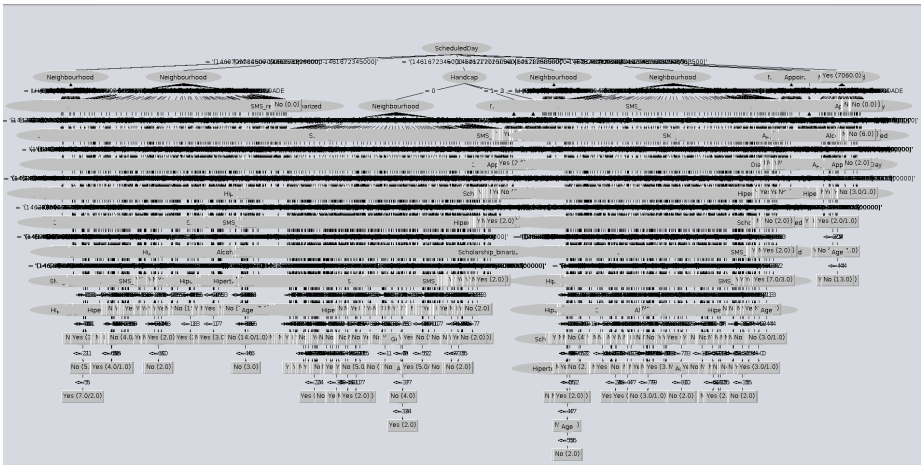

Figura 18: J48 sem *pruning* (árvore com 9273 nodos e 6224 folhas)

Em segundo lugar, corri o mesmo algoritmo optando pela alteração do valor do fator de confiança. Este parâmetro testa a eficiência do *post-pruning*. Ao diminuir este valor, diminui também a quantidade de *post-pruning* ao qual a árvore é sujeita.

Fator de confiança	**Rigor**
0.1	79.6193%
0.2	79.6262%
0.3	79.5987%
0.4	78.0686%
0.5	77.1083%

Tabela 3: Resultados obtidos com a variação do fator de confiança

De facto, verifica-se um pico na percentagem de instâncias corretamente classificadas quando o fator de confiança é 0.2 (79.6262%) (cuja árvore gerada é igual à da Figura 5), após o qual os restantes resultados obtidos demonstram efeitos de *overfitting*, através do decréscimo na precisão de classificação das instâncias. O melhor caso, apresenta uma percentagem de 20.3738% de instâncias incorretamente classificadas, numa árvore com 11 nós e 10 folhas.