Business Intelligence e Mineração de Dados

Marco Rodrigues nº4652

Mestrado em Engenharia de Software

2016-2017

Índice

1. Introdução

O presente documento tem como objetivo explorar a técnica de data mining Árvores de Decisão, fazendo um enquadramento teórico e seguindo depois exemplos práticos de forma a consolidar conhecimento e aplicar os conceitos teóricos.

1.1. Estrutura do Documento

Este documento está estruturado em seis capítulos. O primeiro refere-se á introdução do documento. O segundo faz uma pequena introdução ao tema "data mining". O terceiro capitulo faz um enquadramento ás Árvores de Decisão. O quarto capítulo contém os casos práticos realizados com recurso a algoritmos de Árvores de Decisão.

2. Data Mining

O termo Data Mining, que traduzindo para português resulta em Mineração de Dados, consiste num conjunto de técnicas avançadas para utilizar sobre grandes quantidades de dados á procura de padrões consistentes, de forma que das mesmas seja possível extrair conhecimento [1].

Esse conjunto de técnicas inclui as seguintes:

- Árvores de Decisão (Classificação)
- Regras de Associação
- Redes Neuronais Artificiais
- Algoritmos Genéticos
- Avaliação de Técnicas e Algoritmos

3. Árvores de Decisão

As Árvores de Decisão são uma técnica utilizada para classificação de dados. São usadas para classificar uma coleção de dados através da determinação de uma categoria para os mesmos. A utilização desta técnica permite, por exemplo, classificar entidades ou tipificar cenários, tarefas que se tornaram bastante comuns em soluções de suporte á decisão e correspondem a uma forma de representação de um conjunto de regras que segue uma hierarquia de classes ou valores, tornando o esquema representado graficamente semelhante a uma árvore. Encaixam-se no tipo de aprendizagem supervisionada.

De forma resumida, as Árvores de Decisão permitem classificar instâncias desde o nó raíz até aos terminais (folhas) e cada nó da árvore especifica um teste para os atributos da instância e cada ramo que descende desse nó corresponde a um dos valores possíveis para esse atributo.

Para construir uma árvore de decisão é necessário um conjunto de dados de treino e um conjunto de dados de teste. O conjunto de dados de treino é utilizado para identificar um modelo que classifica os dados considerando a variável de saída. Este conjunto deve ter casos perfeitamente definidos e corretamente classificados, dado que servem para que o algoritmo possa aprender como se deverá comportar perante situações idênticas. O conjunto de dados de teste serve para verificar o desempenho do algoritmo e avaliar a sua utilidade em tarefas de decisão. Representam a única forma de garantir que a estrutura resultante será bem sucedida em previsões de casos no futuro.

Do ponto de vista do utilizador do sistema, o objetivo será encontrar uma árvore que melhor se adapte ao problema, ou seja, a que melhor classifique as instâncias do domínio em questão.

No caso de uma árvore de decisão não classificar todos os casos de forma correta, as exceções são adicionadas ao conjunto de treino de forma que o algoritmo possa “aprender” essas exceções e considerá-las no futuro.

Há dois tipos de árvores de decisão, as Árvores de Classificação e as Árvores de Regressão.

As Árvores de Classificação servem para qualificar os registos e associá-los com a classe determinada e garantir que essa mesma classificação esteja correta.

As Árvores de Regressão realizam estimativas do valor de uma determinada variável.

Há vários algoritmos de implementação de árvores de decisão que seguem uma metodologia top-down como o CART, CHAID, ID3 ou o C4.5.

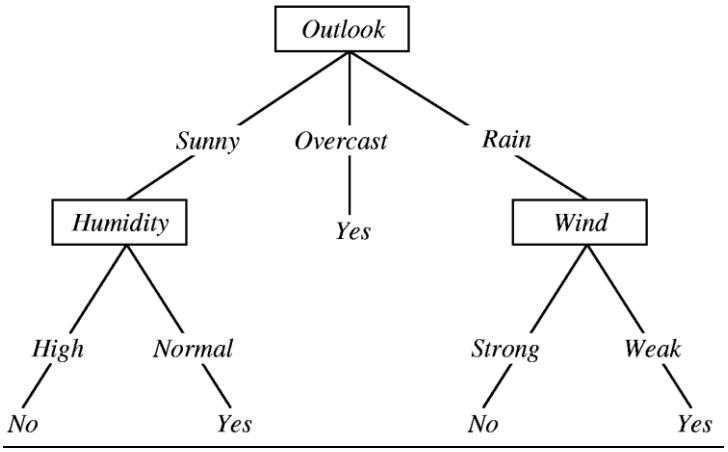

Figura 1 - Exemplo de uma Árvore de Decisão

4. Trabalhos Relacionados

Em vários setores da indústria tem sido cada vez mais aposta a análise avançada de dados, e as Árvores de Decisão são técnicas utilizadas para classificar e desta forma prever algo com um determinado grau de confiança.

Na China a nefropatia por IgA (doença dos rins causada por depósito de anticorpos) é uma das causas mais comuns de Glomerulonefrite (inflamação das estruturas dos rins que são compostas por pequenos vasos sanguíneos) e a medicina tradicional chinesa é uma estratégia de tratamento utilizada para tratar a doença mencionada. No entanto, não é simples identificar a síndrome desta doença de forma precisa de acordo com os sintomas que são apresentados pelos pacientes.

Assim, foi feito um estudo[2] onde foram recolhidos dados clínicos para o intervalo temporal de 2010 a 2016 relativos a 464 casos de adultos com nefropatia comprovada por biópsia e foram construídos modelos em árvores de decisão de classificação e regressão para diferenciar os tipos de síndrome. Os resultados da classificação foram: M1=97.6%, E1=14.6%, S1=50% e T1=52.2%/T2=18.4%, confirmando assim que era um método válido para identificar síndromes da medicina tradicional chinesa de nefropatia por IgA.

Um segundo caso de aplicabilidade de Árvores de Decisão na área da Saúde[4] revela a utilização do algoritmo C4.5 para prever riscos de complicações durante a gravidez de uma mulher. Os resultados ditaram 71.30% de precisão com dataset com dados standardizados e 66.08% de precisão com dataset de dados não standardizados.

Outro caso de aplicabilidade das Árvores de Decisão foi publicado em artigo[3], em que foi proposto um sistema para reconhecimento automático de texto escrito em Bangla (idioma do Bangladesh) que consistia numa árvore de decisão com um MLP (Multilayer Perpeptron) nos nodos terminais. O dataset utilizado consistia num conjunto de 200 imagens e 50 documentos digitalizados e obteve uma precisão de 70.7% nos caracteres classificados corretamente.

5. Casos Práticos

5.1. Dataset BMW

5.1.1. Introdução e Carregamento dos Dados

Este exemplo prático será realizado com a ferramenta de data mining **WEKA** [5] e vai ser utilizado um dataset que diz respeito a um stand da BMW fictício, em que o mesmo está a começar uma campanha promocional para tentar recuperar antigos clientes com um contrato que garante dois anos extra de garantia.

Os quatro atributos que constam no dataset são: intervalo de vencimento [0=$0-$30k, 1=$31k-$40k, 2=$41k-$60k, 3=$61k-$75k, 4=$76k-$100k, 5=$101k-$150k, 6=$151k-$500k, 7=$501k+], ano/mês da primeira compra BMW, ano/mês da última compra BMW, resposta a oferta de extensão de garantia no passado.

O primeiro passo é carregar o ficheiro “bmw-training.arff” no Weka. Este ficheiro contém 3000 registos acerca de antigos clientes.

Para carregar é necessário abrir o sofware Weka, selecionar “Explorer”, “Open file” e por último escolher o ficheiro referido.

Após carregar os dados o Weka apresenta-se como na figura abaixo.

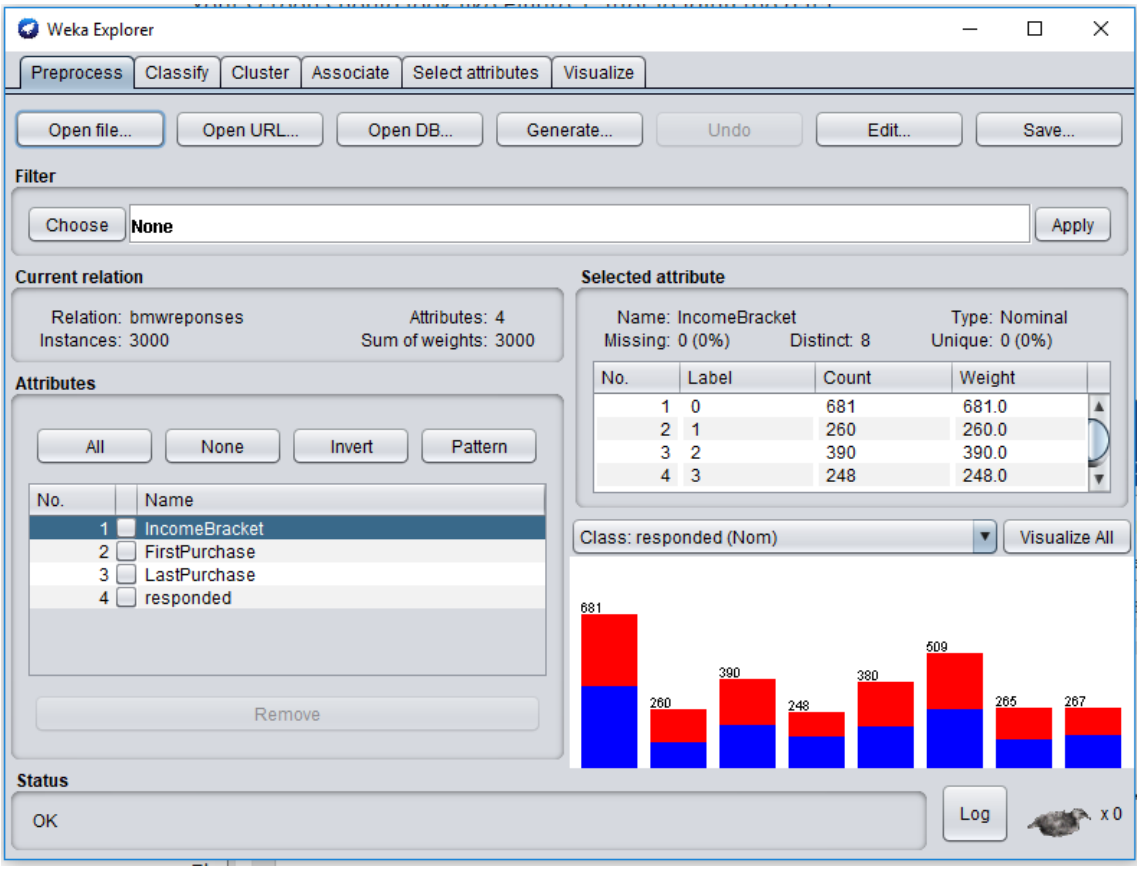

Figura 2 - BMW-Training carregado no Weka

5.1.2. Gerar Modelo Classificador

Neste primeiro caso prático vai ser utilizado um algoritmo de **Árvores de Decisão (Decision Trees)** para classificar. Para tal, é necessário ir á tab "Classify", selecionar a opção "Choose" e na estrutura encontrar "trees" e "J48". O "J48" é uma implementação em Java do algoritmo "C4.5", e este é um algoritmo utilizado para gerar árvores de decisão.

Nas opções "Test Options" deverá estar selecionada a opção "Use training set", ficando o Weka com o seguinte estado antes de se criar o modelo.

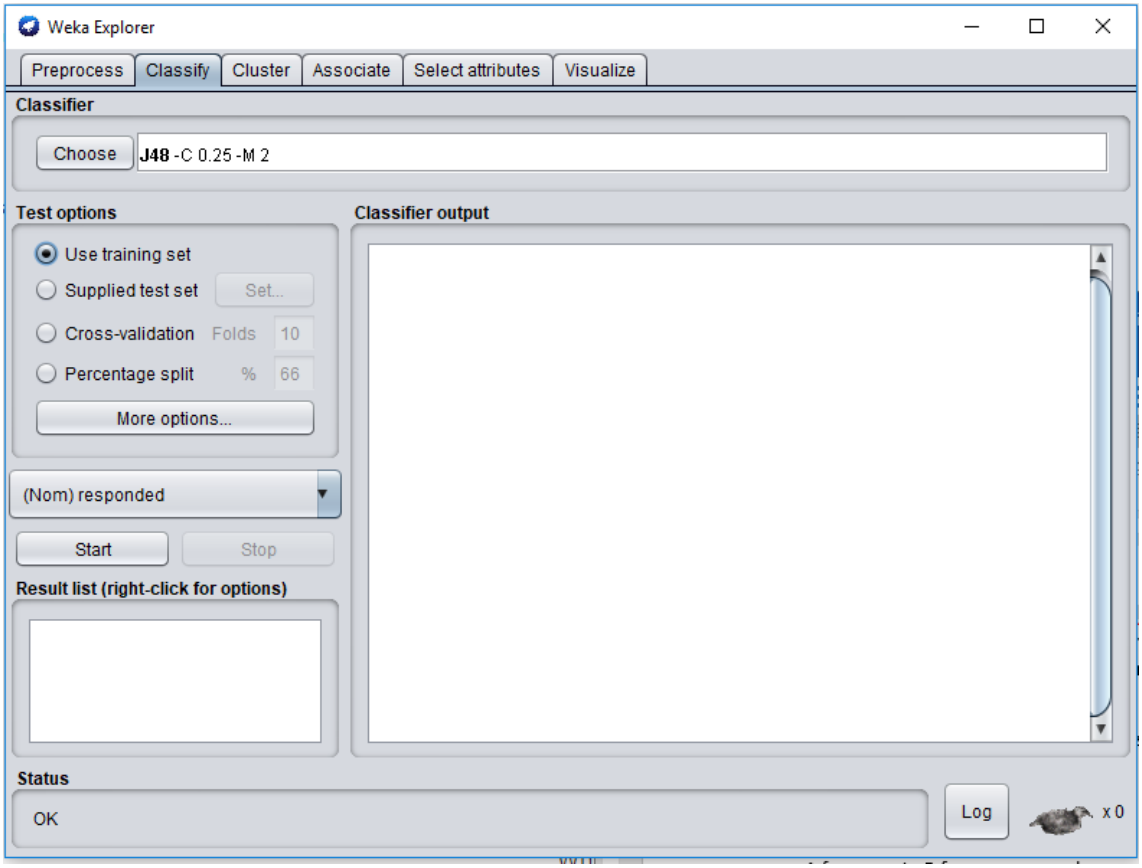

Figura 3 - Classificar

O próximo passo é clicar "Start" para indicar ao Weka que comece a criar o modelo.

Neste caso prático o output gerado é o das imagens abaixo.

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     bmwreponses
Instances:    3000
Attributes:   4
              IncomeBracket
              FirstPurchase
              LastPurchase
              responded
Test mode:    evaluate on training data
```

Figura 4 - Informação geral da execução

No primeiro excerto são apresentados alguns dados gerais acerca da execução, como o número de instâncias utilizadas, o número de atributos e quais foram, etc.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

FirstPurchase <= 200011
|   IncomeBracket = 0: 1 (271.0/114.0)
|   IncomeBracket = 1
|   |   LastPurchase <= 200512: 0 (69.0/21.0)
|   |   LastPurchase > 200512: 1 (69.0/27.0)
|   IncomeBracket = 2: 1 (194.0/84.0)
|   IncomeBracket = 3: 1 (109.0/38.0)
|   IncomeBracket = 4
|   |   LastPurchase <= 200511: 0 (54.0/22.0)
|   |   LastPurchase > 200511: 1 (105.0/40.0)
|   IncomeBracket = 5
|   |   LastPurchase <= 200505
|   |   |   LastPurchase <= 200504: 0 (8.0)
|   |   |   LastPurchase > 200504
|   |   |   |   FirstPurchase <= 199712: 1 (2.0)
|   |   |   |   FirstPurchase > 199712: 0 (3.0)
|   |   LastPurchase > 200505: 1 (185.0/78.0)
|   IncomeBracket = 6
|   |   LastPurchase <= 200507
|   |   |   FirstPurchase <= 199812: 0 (8.0)
|   |   |   FirstPurchase > 199812
|   |   |   |   FirstPurchase <= 200001: 1 (4.0/1.0)
|   |   |   |   FirstPurchase > 200001: 0 (3.0)
|   |   LastPurchase > 200507: 1 (107.0/43.0)
|   IncomeBracket = 7: 1 (115.0/40.0)
FirstPurchase > 200011
|   IncomeBracket = 0
|   |   FirstPurchase <= 200412: 1 (297.0/135.0)
|   |   FirstPurchase > 200412: 0 (113.0/41.0)
|   IncomeBracket = 1: 0 (122.0/51.0)
|   IncomeBracket = 2: 0 (196.0/79.0)
|   IncomeBracket = 3: 1 (139.0/69.0)
|   IncomeBracket = 4: 0 (221.0/98.0)
|   IncomeBracket = 5
|   |   LastPurchase <= 200512: 0 (177.0/77.0)
|   |   LastPurchase > 200512
|   |   |   FirstPurchase <= 200306: 0 (46.0/17.0)
|   |   |   FirstPurchase > 200306: 1 (88.0/30.0)
|   IncomeBracket = 6: 0 (143.0/59.0)
|   IncomeBracket = 7
|   |   LastPurchase <= 200508: 1 (34.0/11.0)
|   |   LastPurchase > 200508: 0 (118.0/51.0)

Number of Leaves  :     28

Size of the tree :      43
Time taken to build model: 0.05 seconds
```

Figura 5 - Árvore de Decisão

De seguida podemos ver o modelo classificador, isto é, a árvore de decisão gerada com o algoritmo "J48" (que corresponde ao C4.5 em Java). Podemos também constatar que o modelo gerou 28 folhas (nós terminais na árvore) e no total tem 43 nós.

```
=== Evaluation on training set ===

Time taken to test model on training data: 0.02 seconds

=== Summary ===

Correctly Classified Instances        1774               59.1333 %
Incorrectly Classified Instances      1226               40.8667 %
Kappa statistic                          0.1807
Mean absolute error                      0.4773
Root mean squared error                  0.4885
Relative absolute error                 95.4768 %
Root relative squared error             97.7122 %
Total Number of Instances             3000

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,662    0,481    0,587      0,662    0,622      0,182    0,616     0,599     1
                 0,519    0,338    0,597      0,519    0,555      0,182    0,616     0,596     0
Weighted Avg.    0,591    0,411    0,592      0,591    0,589      0,182    0,616     0,597

=== Confusion Matrix ===

    a    b   <-- classified as
 1009  516 |    a = 1
  710  765 |    b = 0
```

Figura 6 - Sumário da árvore de decisão

Olhando para o sumário da árvore de decisão gerada, podemos perceber que:

- Correctly Classified Instances 1774 -> 59.13%. Este valor indica-nos que 1774 instâncias foram classificadas corretamente.
- Incorrectly Classified Instances 1226 -> 40.87%. Este valor indica-nos que 1226 instâncias foram classificadas de forma incorreta.
- Kappa Statistic 0.1807. Esta é uma estatística parecida com o coeficiente de correlação, isto é, mede a previsão de acordo com a sua classe verdadeira.
- Mean Absolute Error 0.4773. Esta estatística mede a magnitude dos erros sem considerar a sua direção, isto é, permite medir a precisão para variáveis contínuas.
- Root Mean Squared Error 0.4885. Esta estatística é uma forma diferente de calcular a anterior, o Mean Absolute Error, portanto também mede a magnitude dos erros, no entanto esta dá um peso maior a erros de maior dimensão.
- Relative Absolute Error 95.48%. Este valor é calculado com o Mean Absolute Error dividido pelo erro do classificador ZeroR.
- Root Relative Squared Error 97.71%. Este valor é calculado com o Root Mean Squared Error dividido pelo erro do classificador ZeroR.

Em baixo podemos também analisar a matriz da confusão que é uma tabela que permite identificar imediatamente o desempenho do modelo através do número de falsos positivos e falsos negativos resultantes do mesmo.

Neste caso prático tivemos 516 falsos positivos, isto é, uma instância de dados em que o modelo fez previsão de resultado positivo e o resultado real é negativo, e ainda 710 falsos negativos que são o oposto.

Para visualizar a árvore de decisão representada graficamente é necessário clicar com o botão direito no modelo gerado e selecionar “Visualize Tree”, tal como na figura abaixo.

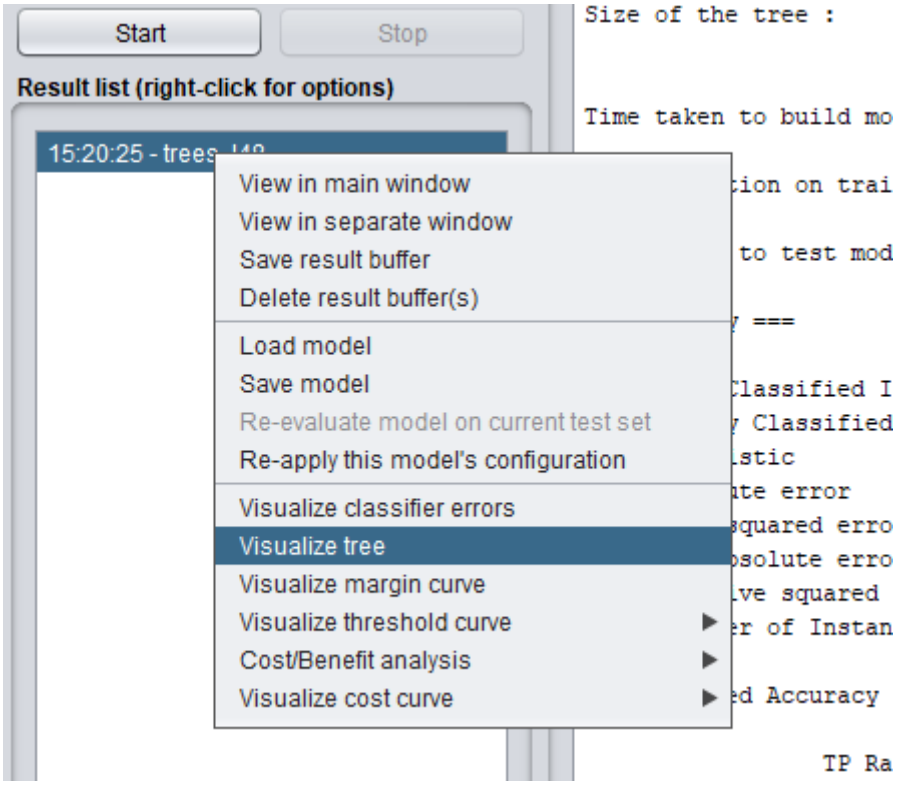

Figura 7 - Visualizar Árvore de Decisão

E o resultado obtido é o seguinte.

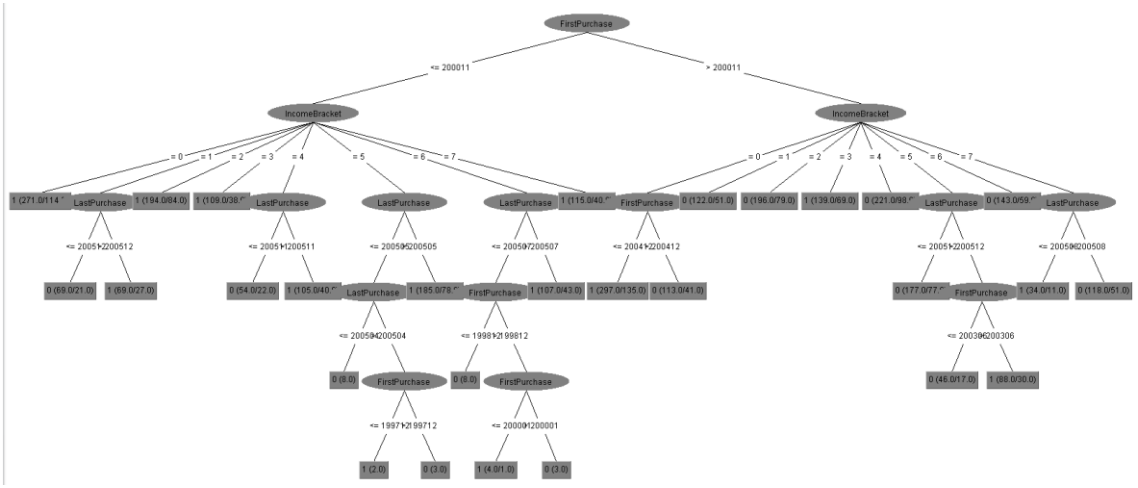

Figura 8 - Árvore de Decisão representada graficamente

Se analisarmos mais detalhadamente a árvore gerada, podemos verificar que o primeiro nodo diz respeito ao atributo “First Purchase”.

Figura 9 - Primeiro nodo da árvore

A primeira regra que o algoritmo atribui aos dados é se a primeira compra é inferior ou igual ao ano 2000 e mês 11 seguem um caminho, senão seguem outro.

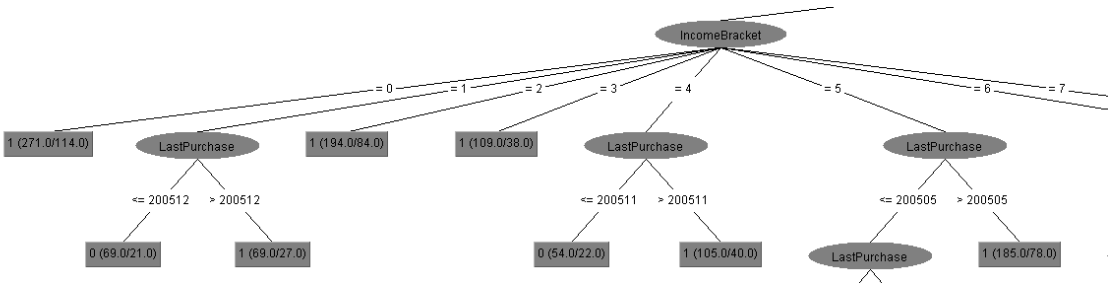

Figura 10 - Mais nodos da árvore

A regra seguinte é relativa ao Income Bracket. Para cada valor possível (entre 0 e 7) segue um nodo diferente. Podemos, no entanto, perceber que se o Income Bracket pertencer ao intervalo 0, o modelo está a prever resultado positivo, isto é que o cliente compre o BMW e aceite o contrato de garantia estendido por dois anos extra. Se o Income Bracket estiver no intervalo 1, o modelo analisa ainda o atributo “Last Purchase” e se este for inferior ou igual ao ano 2005 e ao mês 12, a previsão é negativa, se for superior a previsão é positiva.

Este é o tipo de lógica implementada nas árvores de decisão de classificação, formam uma hierarquia através de conjuntos de regras para permitir obter previsões acerca das classificações para novos conjuntos de dados.

5.1.3. Avaliar Modelo Classificador

De seguida é relevante utilizar um conjunto de dados de teste para avaliar o modelo, ou seja, a sua precisão com os dados de teste não deverá oscilar muito comparando-a com os dados de treino.

Para carregar os dados de teste é necessário na tab “Classify” selecionar a opção “Supplied set test” que é para indicar ao Weka que a próxima execução será para teste e clicar em “Set”, “Open file” e selecionar o “BMW-test.arff”. De seguida fazer “Start”.

O resultado da execução com os dados de teste é o seguinte.

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     bmwreponses
Instances:    1500
Attributes:   4
              IncomeBracket
              FirstPurchase
              LastPurchase
              responded
Test mode:    user supplied test set:  size unknown (reading incrementally)
```

Figura 11 - Execução de dados de teste

Nesta imagem podemos verificar que os dados de teste tinham apenas 1500 instâncias, ao contrário dos casos de treino que tinham 3000.

```
J48 pruned tree
------------------

FirstPurchase <= 200508
|   LastPurchase <= 200512: 0 (812.0/374.0)
|   LastPurchase > 200512: 1 (555.0/234.0)
FirstPurchase > 200508: 0 (133.0/40.0)

Number of Leaves  :     3

Size of the tree :      5


Time taken to build model: 0.03 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0.01 seconds

=== Summary ===

Correctly Classified Instances         852               56.8    %
Incorrectly Classified Instances       648               43.2    %
Kappa statistic                          0.1315
Mean absolute error                      0.4867
Root mean squared error                  0.4933
Relative absolute error                 97.3854 %
Root relative squared error             98.6841 %
Total Number of Instances             1500

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,437    0,306    0,578      0,437    0,498      0,135    0,581     0,538     1
                 0,694    0,563    0,562      0,694    0,621      0,135    0,581     0,563     0
Weighted Avg.    0,568    0,437    0,570      0,568    0,561      0,135    0,581     0,551

=== Confusion Matrix ===

   a   b   <-- classified as
 321 414 |   a = 1
 234 531 |   b = 0
```

Figura 12 - Execução de dados de teste

Nesta imagem podemos constatar que a percentagem de instâncias classificadas corretamente é 56.8% que anda perto dos 59.1% gerados pelos casos de treino. Esta proximidade indica-nos que o modelo não deverá quebrar quando forem aplicados novos casos no futuro, isto é, o modelo vai ser capaz de classificar os mesmos.

No entanto, um modelo com uma percentagem de instâncias corretas na casa dos 60% não é um modelo que dê grandes garantias, não tem muita precisão logo não é um bom modelo.

5.2. Dataset Pima Indians Diabetes

5.2.1. Introdução e Carregamento dos Dados

O “Pima Indians Diabetes” é um dataset que contém informação acerca de análises feitas de diabetes a um povo descendente dos “Pima” (povo nativo nos Estados Unidos).

Os atributos que constam no dataset são: preg (número de vezes que engravidou), plas (concentração de glucose no plasma), pres(pressão sanguínea), skin(espessura da pele dos trícepes em mm),insu(nível de insulina 2 horas depois),mass(índice de massa corporal), pedi(função pedigree de diabetes), age(idade) e class(tested_positive ou tested_negative para diabetes).

O primeiro passo é carregar o ficheiro “diabetes.arff” no Weka. Este ficheiro contém 768 registos.

Para carregar é necessário abrir o sofware Weka, selecionar “Explorer”, “Open file” e por último escolher o ficheiro referido.

Após carregar os dados o Weka apresenta-se como na figura abaixo

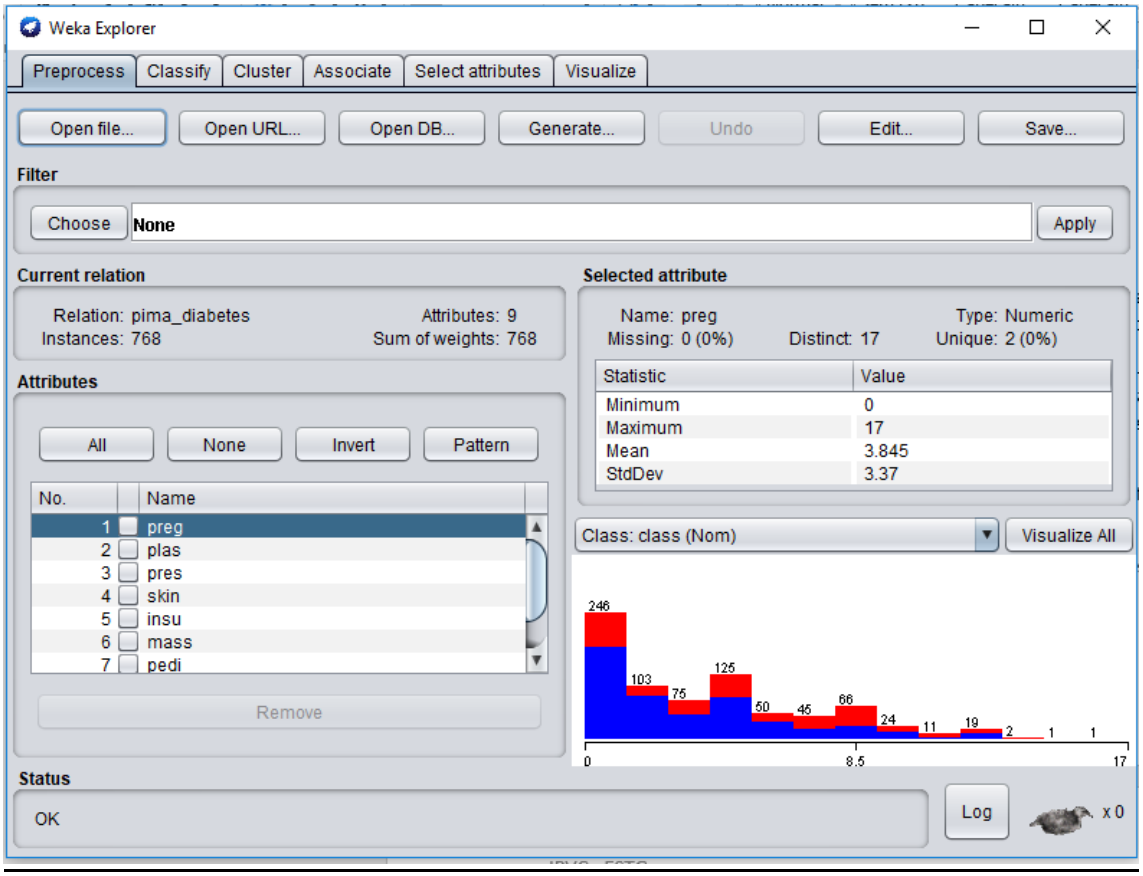

Figura 13 - Dataset Pima carregado

5.2.2. Gerar Modelo Classificador

De seguida na tab “Classify”, clicar em “Choose” e selecionar debaixo de “trees” a opção “J48”.

Em “Test options” selecionar “Use training set” e clicar “Start”. Os resultados são os seguintes.

```
=== Classifier model (full training set) ===

J48 pruned tree
------------------

plas <= 127
|   mass <= 26.4: tested_negative (132.0/3.0)
|   mass > 26.4
|   |   age <= 28: tested_negative (180.0/22.0)
|   |   age > 28
|   |   |   plas <= 99: tested_negative (55.0/10.0)
|   |   |   plas > 99
|   |   |   |   pedi <= 0.561: tested_negative (84.0/34.0)
|   |   |   |   pedi > 0.561
|   |   |   |   |   preg <= 6
|   |   |   |   |   |   age <= 30: tested_positive (4.0)
|   |   |   |   |   |   age > 30
|   |   |   |   |   |   |   age <= 34: tested_negative (7.0/1.0)
|   |   |   |   |   |   |   age > 34
|   |   |   |   |   |   |   |   mass <= 33.1: tested_positive (6.0)
|   |   |   |   |   |   |   |   mass > 33.1: tested_negative (4.0/1.0)
|   |   |   |   |   preg > 6: tested_positive (13.0)
plas > 127
|   mass <= 29.9
|   |   plas <= 145: tested_negative (41.0/6.0)
|   |   plas > 145
|   |   |   age <= 25: tested_negative (4.0)
|   |   |   age > 25
|   |   |   |   age <= 61
|   |   |   |   |   mass <= 27.1: tested_positive (12.0/1.0)
|   |   |   |   |   mass > 27.1
|   |   |   |   |   |   pres <= 82
|   |   |   |   |   |   |   pedi <= 0.396: tested_positive (8.0/1.0)
|   |   |   |   |   |   |   pedi > 0.396: tested_negative (3.0)
|   |   |   |   |   |   pres > 82: tested_negative (4.0)
|   |   |   |   age > 61: tested_negative (4.0)
|   mass > 29.9
|   |   plas <= 157
|   |   |   pres <= 61: tested_positive (15.0/1.0)
|   |   |   pres > 61
|   |   |   |   age <= 30: tested_negative (40.0/13.0)
|   |   |   |   age > 30: tested_positive (60.0/17.0)
|   |   plas > 157: tested_positive (92.0/12.0)

Number of Leaves  :     20

Size of the tree :      39


Time taken to build model: 0.04 seconds
```

Figura 14 – Árvore de Decisão com dataset Pima

Analisando a figura acima é possível perceber que a árvore de decisão tem como primeira regra o atributo plas (concentração de glucose no plasma) e se este for inferior ou igual a 127 segue um ramo se for superior segue outro. Depois podemos, por exemplo, verificar ainda que para quem tem plas inferior ou igual a 127 e tem mass (índice de massa corporal) inferior ou igual a 26.4 a previsão da árvore de decisão é que não tenha diabetes, o resultado é negativo. Ou quem tem glucose superior a 127 e índice de massa corporal inferior ou igual a 29.9 também dá resultado negativo.

```
=== Evaluation on training set ===

Time taken to test model on training data: 0.01 seconds

=== Summary ===

Correctly Classified Instances         646               84.1146 %
Incorrectly Classified Instances       122               15.8854 %
Kappa statistic                          0.6319
Mean absolute error                      0.2383
Root mean squared error                  0.3452
Relative absolute error                 52.4339 %
Root relative squared error             72.4207 %
Total Number of Instances              768

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,936    0,336    0,839      0,936    0,885      0,642    0,888     0,915     tested_negative
                 0,664    0,064    0,848      0,664    0,745      0,642    0,888     0,808     tested_positive
Weighted Avg.    0,841    0,241    0,842      0,841    0,836      0,642    0,888     0,878

=== Confusion Matrix ===

   a   b   <-- classified as
 468  32 |   a = tested_negative
  90 178 |   b = tested_positive
```

Figura 15 - Sumário da Árvore de Decisão

Como sumário do modelo gerado, podemos constatar que o mesmo contém 84.11% de instâncias classificadas corretamente, o que pode indicar um bom modelo, e apenas 15.88% de instâncias classificadas incorretamente. A Matriz da Confusão diz-nos que existem 90 falsos positivos e 32 falsos negativos.

5.2.3. Avaliar Modelo Classificador

De seguida é importante utilizar um conjunto de casos de teste para avaliar o modelo, ou seja, a sua precisão com os casos de teste não deverá oscilar muito relativamente aos casos de treino.

Para carregar os casos de teste é necessário na tab “Classify” selecionar a opção “Supplied set test” que é para indicar ao Weka que a próxima execução será para teste e clicar em “Set”, “Open file” e selecionar o “diabetes-test.arff”. De seguida fazer “Start”.

O resultado da execução com os dados de teste é o seguinte.

```
=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances         130               81.761  %
Incorrectly Classified Instances        29               18.239  %
Kappa statistic                          0.5722
Mean absolute error                      0.2547
Root mean squared error                  0.363
Relative absolute error                 56.3972 %
Root relative squared error             76.6385 %
Total Number of Instances              159

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0,914    0,370    0,828      0,914    0,869      0,580    0,862     0,898     tested_negative
                 0,630    0,086    0,791      0,630    0,701      0,580    0,862     0,745     tested_positive
Weighted Avg.    0,818    0,274    0,815      0,818    0,812      0,580    0,862     0,846

=== Confusion Matrix ===

  a  b   <-- classified as
 96  9 |  a = tested_negative
 20 34 |  b = tested_positive
```

Figura 16 - Avaliação do modelo com casos de teste

Analisando a figura acima é percetível que com os casos de teste o modelo classificou 81.76% de forma correta e 18.23% de forma incorreta, num total de 159 instâncias. Estes valores estão próximos dos gerados com os casos de treino, o que indica que o modelo não deverá quebrar quando for executado com novos casos.

A Matriz da Confusão com os casos de teste diz-nos que houve 20 falsos positivos e 9 falsos negativos.

5.2.4. Outros Algoritmos de Árvores de Decisão

Com o mesmo dataset "Pima Indians Diabetes" foram geradas árvores de decisão recorrendo a outros algoritmos de Árvores de Decisão para efetuar uma comparação com o "J48" e o resultado foi o seguinte:

- **Decision Stump**: Este algoritmo cria uma árvore de decisão de um único nível ligado ao nodo raíz e faz a previsão baseado num único input
  - Instâncias Classificadas Corretamente 73.57%
  - Instâncias Classificadas Incorretamente 26.43%.
- **Hoeffding Tree**: Algoritmo incremental que é capaz de aprender de grandes conjuntos de dados-
  - Instâncias Classificadas Corretamente 77.47%
  - Instâncias Classificadas Incorretamente 22.53%.
- **LMT**: São árvores de classificação com funções de regressão logística
  - Instâncias Classificadas Corretamente 78.52%
  - Instâncias Classificadas Incorretamente 21.48%.
- **REPTree**: Algoritmo de árvore de decisão com aprendizagem rápida
  - Instâncias Classificadas Corretamente 83.07%
  - Instâncias Classificadas Incorretamente 16.93%.

Analisando estes resultados em comparação com o "J48", nenhum destes algoritmos fornece um modelo com uma precisão tão alta como o "J48" que resultou em 84.11% de instâncias classificadas corretamente.

6. Conclusão

Neste trabalho foram realizados dois casos práticos utilizando as Árvores de Decisão como técnica de mineração de dados a dois datasets diferentes.

No primeiro caso foi utilizado o dataset “BMW” que representava um conjunto de dados de antigos clientes de um stand de automóveis. Após a aplicação de algumas técnicas foi criada uma árvore de decisão baseada no algoritmo “J48” que gerou um modelo classificador com uma taxa de classificação correta das instâncias de 59.13% e uma taxa de classificação incorreta de 40.87%. Gerou ainda 516 falsos positivos e 710 falsos negativos. A avaliação do modelo através da utilização de casos de teste ditou que foram classificadas corretamente 56.8% das instâncias.

Com estes valores, este não é propriamente um bom modelo para o conjunto de casos de treino dado que apenas transmite 59.13% de precisão.

No segundo caso prático foi utilizado o dataset “Pima Indians Diabetes”, que diz respeito a análises feitas de diabetes a um povo descendente dos “Pima”, para o qual foi gerada uma árvore de decisão baseada no algoritmo “J48” do Weka. Este modelo continha 84.11% de instâncias classificadas corretamente, contra 15.88% de instâncias incorretas. A matriz da confusão indica também que existiram 90 falsos positivos e 32 falsos negativos. A avaliação do modelo através da utilização dos casos de teste indicou que 81.76% das instâncias foi classificada corretamente e 18.23% de forma incorreta. Este pode ser bom modelo, visto que uma percentagem superior a 80% já dá um valor de precisão interessante.

Por fim, e utilizando o mesmo dataset “Pima Indians Diabetes” foram gerados modelos classificadores com base em outros algoritmos de árvores de decisão, como o “Decision Stump“ que classificou corretamente 73.57% das instâncias, ou o “REPTree” que classificou corretamente 83.07% das instâncias. Nenhum dos algoritmos usados chegou aos 84.11% do “J48”.

7. Referências

[1] – Bibliografia sobre Data Mining. Disponibilizada pelo Prof. Jorge Ribeiro na página do Moodle da disciplina de Business Intelligence e Mineração de Dados do Mestrado em Engenharia de Software.

[2] – Artigo sobre a utilização de Árvores de Decisão para classificação de síndrome de doenças renais. Gu, Yanghui; Wang, Yu; Ji, Chunlan; Fan, Ping; He, Zhiren; Wang, Tao; Liu, Xusheng; Zou, Chuan, "Syndrome Differentiation of IgA Nephropathy Based on Clinicopathological Parameters: A Decision Tree Model.", Mar.2017

[3] – Artigo sobre a utilização de Árvores de Decisão para reconhecimento de texto escrito em Bangla. Ranjit Ghoshala, Anandarup Royb, Bibhas Ch. Dharac and Swapan K. Paruib, "Recognition of Bangla text from outdoor images using decision tree model.", Fev.2017

[4] – Artigo sobre a utilização de Árvores de Decisão para classificação de casos de risco de complicações na gravidez de mulheres. Lakshmi.B.Na, Dr.Indumathi.T.Sb , Dr.Nandini Ravic, "A study on C.5 Decision Tree Classification Algorithm for Risk Predictions during Pregnancy", Dez.2016

[5] – Sítio da ferramenta de Data Mining Weka. Disponível em: http://www.cs.waikato.ac.nz/ml/weka/