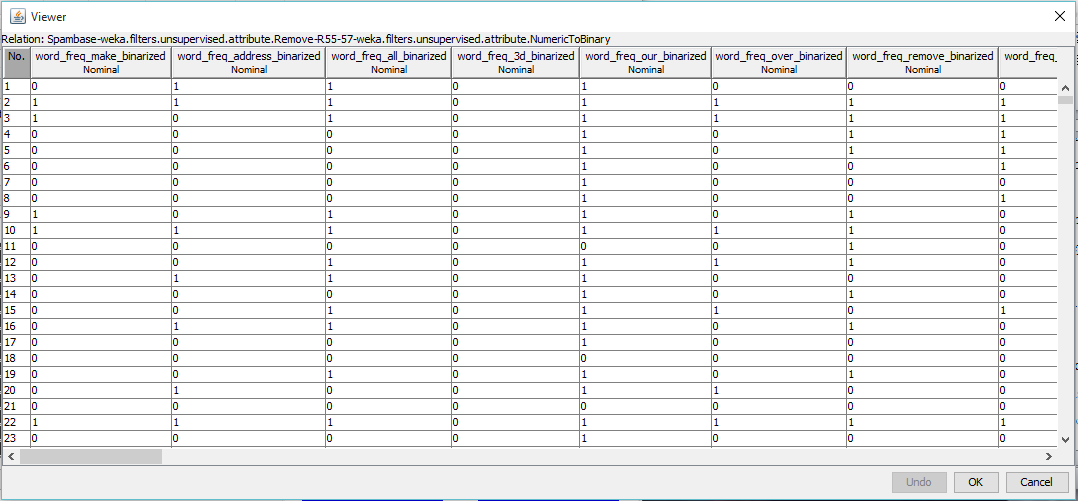
**Tarefa 1: filtragem de *spam***

Nesta actividade utilizamos o Weka para treinar um classificador Naïve Bayes, para efeitos de detecção de *spam*.

Como conjunto de dados utilizaremos o conjunto *Spambase*, constituído por um conjunto de *e-mails* marcados a partir de uma única conta de *e-mail*.

Algumas operações simples de pré-processamento são necessárias antes dos dados estarem prontos a ser utilizados. faremos isso com o Weka:

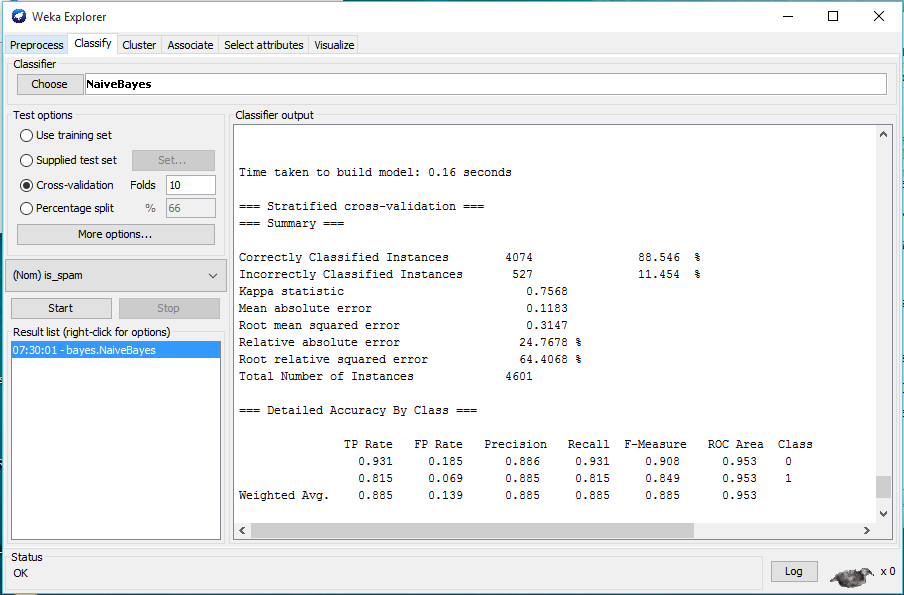
1. No painel *Preprocess* carregamos o conjunto de dados *spambase.arff*.
2. A lista completa dos “58” atributos do conjunto de dados aparece na secção "*Attributes*".
3. Eliminamos os atributos *capital\_run\_length\_average*, *capital\_run\_length\_longest* e *capital\_run\_length\_total*, marcando-os na caixa à sua esquerda e fazendo *click* no botão *Remove*. Sobrando 55 atributos.
4. Os demais atributos representam frequências relativas de várias palavras e caracteres importantes presentes em mensagens de *e-mail*. convertendo as referidas frequências em valores booleanos: 1 se a palavra ou caracter está presente no *e-mail*, 0 se não está presente. Para tal, selecionamos o botão *Choose* na secção *Filter* na parte superior da janela e escolha *Filters > unsupervised > attribute > NumericToBinary* e *click* no botão *Apply*. Todas as frequências numéricas nos atributos são convertidos para valores booleanos. Cada *e-mail* é agora representado por um vector com 55 dimensões que representa se uma determinada palavra existe ou não no *e-mail*. Esta forma de representação é denominada *bag of words* (é uma forma de representação simplista uma vez que não tem em conta a ordem das palavras).



1. Guardamos os dados pré-processados para posterior uso. Utilizamos para tal o botão *Save*.

Com os dados carregados, desejamos treinar um classificador Naïve Bayes para diferenciar os *e-mails* regulares do *spam* ajustando-se à distribuição do número de ocorrências de cada palavra nas duas classes de *e-mail*. No separador *Classify*:

1. Fizemos um *click* no botão *Choose* e seleccionamos *classifiers > Bayes > Naivebayes*.
2. Com as opções seleccionadas por defeito, fizemos *click* no botão *Start* para construir o classificador. Analisamos a saída produzida, particularmente as percentagens de instâncias classificadas correcta e incorrectamente. Obtivemos o seguinte resultado: das 4601 instâncias de treino 4074 foram classificadas correctamente a que corresponde um saldo posititvo de 88,546% e 527 foram classificadas de forma incorrecta que representa 11,454% do total de instâncias.



* 1. O classificador empregou 0.16 segundos ao treinar e classificar os dados.

1. Examinamos o modelo produzido pelo Weka (apresentado antes da informação sobre o rendimento do classificador). Observamos as probabilidades *a priori* para cada classe: Classe 0 = 0,61 e Classe = 0,39.
   1. R: A probabilidade de que o email pertença a classe *span* ou *não* é calculada multiplicando a probabilidade *a priori* de cada uma das diferentes classes pelas probabilidades da ocorrência de cada valor de todos atributos disponíveis na condição de as classes ja se tenham ocorridas.
   2. Determinamos a probabilidade condicional de observar a palavra "3d" dado que uma mensagem pertence à classe *spam* P(3d|*spam*) ou *não spam* P(3d|*não spam*). Para isso utilizamos as contagens produzidas pelo modelo, apresentadas na janela de saída (*Classifier output*) no separador *Classify*. R: O conjunto de dados possui 4601 instâncias das quais 61% destas são classificadas com sendo *não span* e 39% são classificadas como *não span* convertendo esses valores em quantidades de instâncias pela fómula (valor% \*4601/100%) temos: *span =* 1794,39 e *não* *span* = 2806,61. Do conjunto de dados 40 das instâncias *span* são observadas com a palavra 3d e 90 delas não. A probabilidade condicional de se observar a palavra "3d" dado que uma mensagem pertence à classe *spam* P(3d|*spam*) = 40 / 1794,39 = 0,0222917 e *não spam* P(3d|*não spam*) = 90 / 2806,61 = 0,03206716.
2. Assumamos agora que somos *spammers* tentando burlar um sistema de detecção de *spam* baseado num classificador Naïve Bayes no sentido de classificar uma mensagem *spam* como *ham* (i.e. uma mensagem válida). Utilizamos o conjunto de dados de treino completo para treinar o classificador e apliquemos o modelo aprendido a um conjunto de teste dedicado. Carregamos o conjunto de teste no Weka. No separador *Classify*, seleccionamos *supplied test set > set > open file* e seleccionamos o ficheiro *spambase\_test.arff*. Este ficheiro contém um vector de dados binário representando um *e-mail* considerado *spam*. Execute o classificador sobre este conjunto de teste. É o *e-mail* classificado correctamente?R: Sim realmente o e-mail é classificado correctamente com uma precentagem de 100% de instâncias correctamente classififcadas.
3. Abra o ficheiro de teste *spambase\_test.arff* num editor de texto (por ex. o bloco de notas). Identifique boas palavras, características da classe *não spam* e acrescente-as ao *e-mail*. Importante: não altere a etiqueta de classe do ficheiro de teste (último valor dos atributos). Ao testar o Weka ignora este atributo e utiliza o classificador treinado previamente para predizer a etiqueta correspondente ao *e-mail*. Execute novamente o classificador com o ficheiro de teste modificado. Alterou-se a classe atribuída ao *e-mail*?R:

**Tarefa 2: classificação de vidro**

Nesta actividade vamos experimentar a classificação com o método de *k vizinhos mais próximos*. Na mesma utilizamos o conjunto de dados *glass*. Este conjunto foi criado pelo Serviço de Ciência Forense dos Estados Unidos (*U. S. Forensic Science Service*) e contém dados acerca de seis tipos diferentes de vidro. O vidro é descrito pelo seu índice de refracção e pelos elementos químicos contidos no mesmo.

No Explorador do Weka, carregue o ficheiro *glass.arrf*.

* 1. R: O conjunto de dados possui 10 atributos com valores numéricos.
  2. R: Os seus nomes são: RI, Na, Mg, AL, Si, K, Ca, Ba, Fe e Type.
  3. O atributo que correspondente à classe é “Type”.

1. No separador *Classify*, seleccione *classifiers > lazy > Ibk* . Esta opção corresponde ao método *dos k vizinhos mais próximos* (kNN).
2. No painel *Test options*, seleccionamos *Use training set* e pressionamos o botão *start*.
3. Observe a saída do classificador e considere o seguinte:
   1. O rendimento do cassificador é positivo de 100%.
   2. Porquê se considera uma má ideia testar um classificador kNN (k = 1) sobre o próprio conjunto de treino? R: Porque com K = 1, dependendo do conjunto de treino isso se o conjunto de treino tiver muitas instâncias e apenas uma instância mais próxima de todas em função da distancia e com uma classificação de classe diferente das outras instancias também próximas o classificador vai obter a classe da única instância seleccionada como a mais próxima enquanto que existem múltiplas instâncias próximas com classe diferente o que corresponde a maioria.
   3. Se espera que a classificação sobre um conjunto de teste seja tão boa quanto esta?R: Dependo muito da quantidade de instâncias para o conjunto de treino e da quantidade do conjunto de teste como também do parámetro K valores mais próximos.
4. Agora vamos avaliar o classificador utilizando uma validação cruzada. Executamos o classificador, utilizando o número de folhas fixado por defeito (10).
   1. O rendimento do classifcador relativamente ao caso anterior é positivo: que 70,56% das instâncias foram classificadas correctamente e 29,43% delas incorrectamente.
5. Exploremos agora o efeito do parâmetro k. Para tal, executamos várias vezes o classificador com diferentes valores do parâmetro (1, 3, 5, 10, 15), utilizando sempre para a avaliação a validação cruzada em 10 folhas. podemos alterar o parâmetro do algoritmo no painel de opções, fazendo *click* diante do botão *Choose*.
   1. R: O rendimento do classificador para a variação dos valores de K altera-se, quanto maior é o valor de K menor é o rendimento do classificador: (K=1, redimento=70,56%); (K=3, redimento=71,96%); (K=5, redimento=67,76%) (K=10, redimento=66,36) (K=15, redimento=62,61); com excessão da transição de K=1 para K=3 ai o rendimento do classificador aumentou 1,4%.
   2. A figura abaixo representa graficamente os resultados (valor de k no eixo x e o rendimento no eixo y).
   3. É possível concluir algo observando o gráfico? R: Sim conclui-se que quanto maior é o valor de K para o referido conjunto de treino e o método de validação menor é o rendimento do classificador.

O método dos vizinhos mais próximos, a semelhança de outros métodos de aprendizagem, é sensível à presença de ruído nos dados de treino. Agora faremos a injecção de certa percentagem de ruído nos valores de classe e observaremos o seu efeito no rendimento do classificador. O ruído pode ser introduzido através do filtro *AddNoise* (*filters > unsupervised > attribute > AddNoise*). Neste caso é importante que o ruído seja introduzido somente nos dados de treino, permanecendo inalterados os dados de teste. Para tal utilizaremos um *metaclassificador*, designado *FilteredClassifier* (*classifiers > meta > FilteredClassifier*). O *metaclassificador* deve ser configurado para utilizar *IBk* como classificador e *AddNoise* como filtro. Isto pode ser efectuado utilizando a correspondente janela de opções.

1. Construindo uma tabela com o rendimento estimado do classificador, através da validação cruzada em 10 folhas, para seis diferentes percentagens de ruído (0%, 10%,.., 50%) e diferentes valores de k (1, 3 e 5). Obtivemos o seguinte resultado:

O rendimento estimado do classificador através da validação cruzada em 10 folhas:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  | **Rendimento** |  |
| **Percentagem De Ruido** | **K = 1** | **K = 3** | **K = 5** |
| **0%** | **100%** | **80,84%** | **78,97%** |
| **10%** | **99,53%** | **75,70%** | **73,83%** |
| **20%** | **99,53%** | **67,76%** | **66,82%** |
| **30%** | **99,53%** | **60,28%** | **54,20%** |
| **40%** | **99,53%** | **53,74%** | **48,13%** |
| **50%** | **99,53%** | **54,67%** | **48,13%** |

* 1. Que efeito provoca o aumento do ruído?

R: O aumento do ruido na maioria das vezes baixou o rendimento associado a classificação.

* 1. Qual é efeito derivado da alteração do valor de k?

R: A experiência mostra que quanto maior for o valor de K menor será o rendimento do classificador.

**Tarefa 3: selecção de atributos**

Investiguemos agora que subconjunto dos atributos produz o menor erro de validação cruzada sobre o conjunto de dados *Glass*, utilizando o algoritmo *KNN*.

A realização de uma busca exaustiva de todos os subconjuntos possíveis é impraticável (porquê?)R: É impraticável Porque no caso das árvores em algum ponto da árvore que é aprendida atributos irrelevantes, são invariavelmente escolhido pra ramificar causando erros aleatórios quando o teste dos dados é processados (a medida que vamos ramificando a árvore cada vez menos dados estarão disponíveis para ajudar a tomada de decisões, nalgum ponto com poucos dados atributos aleatórios pareceram bons), por isso empregaremos um procedimento de eliminação para trás (ver ref. Witten). Para tal, primeiro consideremos a eliminação de cada atributo individualmente do conjunto de dados completo e a realização de uma validação cruzada para cada versão reduzida do conjunto de dados. Uma vez determinado o melhor subconjunto de oito atributos, repete-se o procedimento para a escolha do melhor subconjunto de sete atributos e assim sucessivamente.

1. Construa uma tabela mostrando o melhor subconjunto de atributos e a menor taxa de erro obtidos em cada iteração. Compare as taxas de erro obtidas neste processo com a correspondente ao conjunto de dados completo.

| **Tamanho do Subconjunto**  **(Nº de Atributos)** | **Atributos no “Melhor” Subconjunto** | **Taxa de Erro** |
| --- | --- | --- |
| 9 | RI, Na, Mg, Al, Si, K, Ca, Ba, Type | 0,0717 |
| 8 |  |  |
| 7 |  |  |
| 6 |  |  |
| 5 |  |  |
| 4 |  |  |
| 3 |  |  |
| 2 |  |  |
| 1 |  |  |