UNIVERSIDADE KATYAVALA BWILA
INSTITUTO SUPERIOR POLITÉCNICO
CIÊNCIAS DA COMPUTAÇÃO

# APRENDIZAGEM AUTOMÁTICA

## PRÁTICA DE LABORATÓRIO 2:

## TREINO DE CLASSIFICADORES COM O SUPORTE DA FERRAMENTA WEKA

Outubro de 2015

UNIVERSIDADE KATYAVALA BWILA
INSTITUTO SUPERIOR POLITÉCNICO
CIÊNCIAS DA COMPUTAÇÃO

# APRENDIZAGEM AUTOMÁTICA

## PRÁTICA DE LABORATÓRIO 2:

## TREINO DE CLASSIFICADORES COM O SUPORTE DA FERRAMENTA WEKA

**Elaborado por:**
**Venâncio Ezequias Sapalo**
**&**
**Zinga Firmino René**

**Professores:**
**Phd. Lázaro Makili**
**&**
**Moisés Ferreira**

Outubro de 2015

# Objectivos

- Utilizar os métodos Naïve Bayes e kNN para treinar classificadores para conjuntos de dados concretos
- Analisar o modelo aprendido ao treinar um classificador Naïve Bayes
- Analisar uma estratégia de selecção do valor ideal para o parâmetro de um classificador com um conjunto de dados concreto
- Explorar o efeito produzido no rendimento de um classificador kNN pela introdução de ruído num conjunto de dados
- Analisar uma estratégia de selecção do subconjunto de atributos mais efectivo para a criação de um classificador

# Introdução

No presente informe usar-se-á dois métodos para treino de classificadores para conjuntos de dados concretos. Examinaremos um procedimento para a selecção do valor ideal do parâmetro associado a um algoritmo ao treinar um classificador e outro para a selecção de um subconjunto de atributos, o mais efectivo, para a construção de um classificador. Para tais efeitos, serão utilizados os conjuntos de dados *Spambase* e *Glass*.

É fundamental o estudo dos seguintes aspectos:

- Avaliação dos modelos e métodos de validação:
  - Medidas de desempenho,
  - *holdout*,
  - validação cruzada,
  - matriz de confusão

- Algoritmos Naïve Bayes e KNN
- Selecção de atributos

## Avaliação dos modelos e métodos de validação

*Avaliação* é a chave para fazer um progresso real na mineração dos dados. Existem mitos modelos, de formas a determinar que modelo usar na resolução de um problema, precisamos de maneiras sistemáticas de avaliar como cada método diferente trabalha e fazer comparação uns com os outros.
*Validação* consiste em avaliar a proporção do erro que se espera que um algoritmo cometa no caso de um problema determinado.

Alguns desses modelos:

- **Medidas De Desempenho:** No caso de problemas de classificação a medida mais comum é a designada *taxa de erro*. Se define através da proporção dos exemplos incorrectamente classificados de entre a totalidade dos exemplos de teste. Em muitas situações fala-se de três conjuntos de dados: conjunto de treino, o de validação de dados, e o de teste. Conjunto de treino é usado por um ou mais modelos de aprendizagem que classificam. O conjunto de validação usa-se para optimizar parâmetros dos classificadores, ou para seleccionar um em particular. O conjunto de teste é usado para calcular a taxa de erro do final e optimizado método. Os três conjuntos devem ser escolhidos independemente. No caso do conjunto de teste ser diferente dos outros conjuntos possibilita obter taxas de erros confiáveis.

- **Holdout:** O método holdout reserve parte do conjunto de dados para testes e o restante para treino (e põe de parte deste se necessário para a validação). Em termos prácticos, é comum o

holdout usar 1/3 (um terço) dos dados para teste e o restante 2/3 (dois terços) para treino.

- **Validação Cruzada:** Na validação cruzada, decide-se uma quantidade fixa de folhas (k), ou partições dos dados. Suponha que usamos três (K=3), logo os dados são divididos em três partições aproximadamente iguais; cada um de cada vez é usado para testes o resto para treino. Assim, usa-se dois terços (2/3 ou k-1) para dados de treino e um terço para testes (1/3 ou k=1), e repete-se o procedimento três vezes, até que no final todas instâncias sejam usadas exactamente uma vez para teste.

- **Matriz De Confusão:** também chamada tabela de contingência permite quantificar diferentes tipos de erro, tornando assim possível visualizar o desempenho do algoritmo.
  Considerando um conjunto de teste com N exemplos
  – Para um exemplo positivo
  • Se a predição for positiva
    - ❖ Verdadeiro positivo (tp)

  • Caso contrário
    - ❖ Falso negativo (fn)

  – Para um exemplo negativo
  • Se a predição for negativa
    - ❖ Verdadeiro negativo (tn)

  • Caso contrário
    - ❖ Falso positivo (fp)

| | | Classe Verdadeira | | |
|---|---|---|---|---|
| | | 0 (+) | 1 (-) | Total |
| Classe predita | 0 (+) | tp | fp | p' |
| | 1 (-) | fn | tn | n' |
| Total | | p | n | N |

## Algoritmos Naïve Bayes

Implementam a inferência desde um ponto de vista probabilístico, baseando-se no pressuposto de que as variáveis de interesse são governadas por distribuições de probabilidades e que é possível tomar decisões óptimas com base nas referidas probabilidades e nos dados observados.

Designados métodos de aprendizagem bayesianos, pelo facto de a base teórica ter sido implementada por Thomas Bayes.

Em aprendizagem automática muitas vezes nos interessa determinar a melhor hipótese de um espaço H, dado um conjunto de exemplos observados E. Uma forma de especificar, o que virá a ser a melhor hipótese, é dizer que buscamos a mais provável hipótese, dado os exemplos E mais qualquer conhecimento inicial sobre as probabilidades a prior de várias hipóteses em H.

O teorema de Bayes provê um método directo para o cálculo de tais probabilidades:

$$P(h \mid E) = \frac{P(E \mid h)P(h)}{P(E)}$$

P(h) : denota a probabilidade inicial de que a hipótese h seja correcta, antes de observarmos os dados de treino (*probabilidade a priori*).

P(E|h) : denota a probabilidade associada à observação dos dados E caso a hipótese h seja a correcta.

P(E) : denota a probabilidade de que os dados de treino E sejam observados sem nenhum conhecimento de que $h \in H$

P(h|E) : denota a probabilidade de que a hipótese h seja correcta uma vez observados os dados de treino E (*probabilidade a posteriori*).

Num cenário de aprendizagem, geralmente considera-se um conjunto de hipóteses candidatas H e trata-se de achar a hipótese $h \in H$ mais provável dados os exemplos de treino E:

$$h_{MAP} = \arg\max_{h \in H} P(h \mid E) = \arg\max_{h \in H} \frac{P(E \mid h)P(h)}{P(E)}$$

$$= \arg\max_{h \in H} P(E \mid h)P(h)$$

A fórmula acima denomina-se *hipótese máxima a posteriori* (MAP).

O classificador deste algoritmo chama-se classificador Naïve Bayes porque aplica-se a problemas de classificação nos quais se pode assumir que os valores dos atributos são independentes uns dos outros. Dado um conjunto de treino e uma nova instância, descrita por um vector de valores dos atributos, o algoritmo trata de predizer a classe correspondente à nova instância.

O enfoque bayesiano consiste em atribuir à nova instância a classe mais provável, dado o vector de atributos que descreve a mesma.

$$c_{MAP} = \arg\max_{c_i \in C} P(c_i \mid a_1, a_2, \ldots, a_n)$$

A classe denomina-se *classe máxima a posteriori*, cuja saída e produzida por $c_{NB} = \arg\max_{c_i \in C} P(c_i) \prod_j P(a_j \mid c_i)$

Variantes:

1. Estimação-m: para evitar probabilidades condicionais iguais a 0.

$$p(A = v \mid c) = \frac{n_c + m.p}{n + m}$$

- $n_c$ - # de instâncias que têm o valor v no atributo A e pertencem à classe c
- $n$ - # de instâncias que pertencem à classe c
- $p$ - estimação a priori da probabilidade que se deseja calcular; se não está disponível, assumir uma distribuição uniforme (se o atributo tem $k$ valores, $p = 1/k$)
- $m$ – parâmetro corrector, denominado *tamanho da mostra equivalente.* Determina o peso a atribuir a $p$ relativamente aos dados observados

2. Valores contínuos: usado para os atributos numéricos, onde se estima as probabilidades com os cálculos das correspondentes médias e desvios padrões.

Se utiliza a expressão correspondente à função de densidade de probabilidade para uma distribuição normal com média μ e desvio padrão σ

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

# Algoritmos KNN

Um tipo de algoritmo de aprendizagem baseado em instância, consistindo apenas em armazenar o conjunto de dados de treino apresentado. Ao classificar um novo exemplo, se recupera da memória um conjunto de instâncias similares e estas são utilizadas como base da classificação. Por relegar a aprendizagem ao momento da classificação são denominadas técnicas de aprendizagem ociosas (lazy learning). A recuperação das instâncias da memória é feita com base numa função da distância. Na maioria dos casos se utiliza a distância euclidiana:

$$d(\mathbf{x_1}, \mathbf{x_2}) = \sqrt{\sum_{i=1}^{n} (a_{1i} - a_{2i})^2}$$

O algoritmo KNN (*k-NEAREST NEIGHBOR*), é o mais básico da aprendizagem baseado em instância assume que todas as instâncias correspondem a pontos num espaço n-dimensional $R^n$

Considerando uma função objectivo discreta (problema de classificação), o algoritmo retorna a classe mais comum entre os k exemplos de treino mais próximos da instância de teste. Para um problema com duas classes o valor de k deve ser ímpar e, em geral, não deve ser múltiplo do número de classes.

A fórmula assume implicitamente que os atributos são numéricos, logo para atributos ordinais:

– A diferença entre dois valores distintos se considera igual a 1

– Se for o mesmo valor a diferença é igual a 0

Caso existam valores omissos, a diferença entre os valores dos atributos é tomada para que seja a maior possível.

• Para atributos nominais:

– Se um ou ambos valores são omissos a diferença entre os mesmos se considera igual a 1

• Para atributos ordinais:

– Se ambos valores forem omissos, a diferença é igual a 1

– Se apenas um dos valores for omisso, a diferença se toma como o maior valor entre o valor de atributo presente e um menos o referido valor.

Algoritmo kNN com distâncias ponderadas é uma variante do método consistindo em apenas pesar a contribuição dos k vizinhos mais próximos na função de decisão de acordo à sua proximidade à instância a ser classificada

• Atribui-se maior peso aos vizinhos mais próximos da instância a ser classificada. Tem a seguinte formulação:

$$c(x_t) = \arg\max_{c \in C} \sum_{i=1}^{k} w_i \delta(c, c(x_i)) \qquad w_i = \frac{1}{1 + d(x_t, x_i)}$$

# Selecção de atributos

Muitos algoritmos de aprendizagem são desenhados para aprender o atributo mais apropriado a usar na tomada de decisão. Por exemplo as árvores de decisões, devem escolher o atributo mais prometedor para sua expansão — teoricamente — e nunca selecionar atributos irrelevantes ou inúteis. Ter-se mais atributos deve certamente — teoricamente — resultar em mais poder de descriminação. Na prática não é bem o caso, porque adicionar atributos irrelevantes ao conjunto de dados muitas vezes confunde o sistema de aprendizagem. No caso das árvores de decisões quanto mais profunda se vai tornando a árvore, atributos irrelevantes parecerão bons para a tomada de decisões reduzindo a taxa de êxitos para aprendizagem na fase de teste.

Por causa do efeito negativo dos atributos irrelevantes, em muitos esquemas de aprendizagem, é comum proceder a aprendizagem através da seleção de atributos, uma etapa que visa eliminar todos excepto os atributos relevantes. A melhor maneira de o fazer, é selecionando os atributos relevantes manualmente, baseado num entendimento profundo do problema de aprendizagem e o significado real do atributo. No entanto, métodos automáticos podem ser úteis e levados a cabo. A redução da dimensão dos dados por apagar atributos inconvenientes melhora o desempenho de aprendizagem do algoritmo de aprendizagem. Mais importante ainda, a redução da dimensão dos dados, torna mais fácil a interpretação do conceito alvo, concentrando a atenção do usuário nos atributos mais relevantes.

# Experimentos

Depois da consideração dos conceitos importantes subjacentes as tarefas que havemos de executar, é chegada altura da concretização das mesmas. Relembrar que todos os experimentos foram realizados com suporte do software WEKA.

## 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.

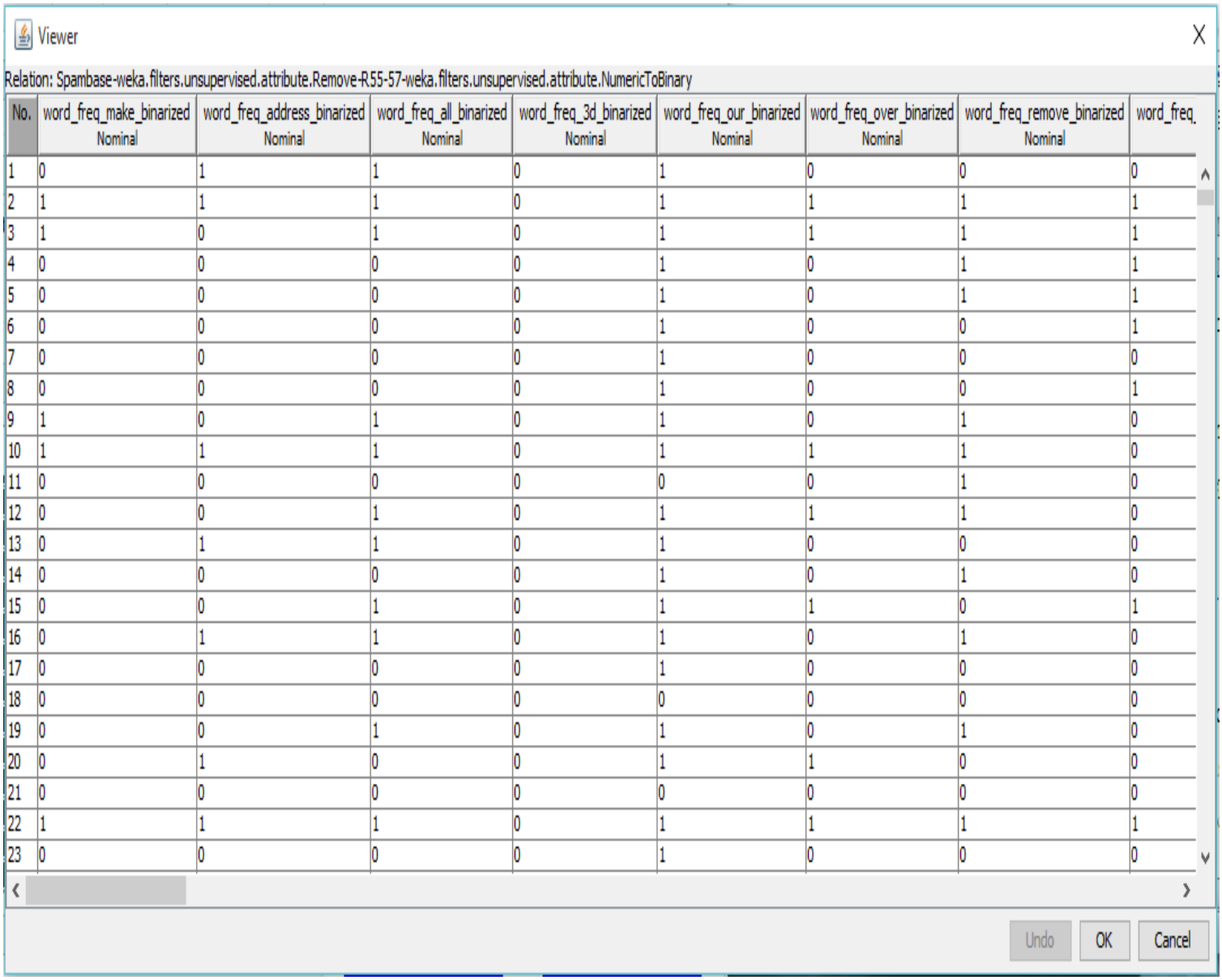
No painel *Preprocess* carregamos o conjunto de dados *spambase.arff*.

A lista completa dos "58" atributos do conjunto de dados aparece na secção "*Attributes*".

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.

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 *ba of words* (é uma forma derepresentação simplista uma vez que não tem em conta a ordem das palavras).

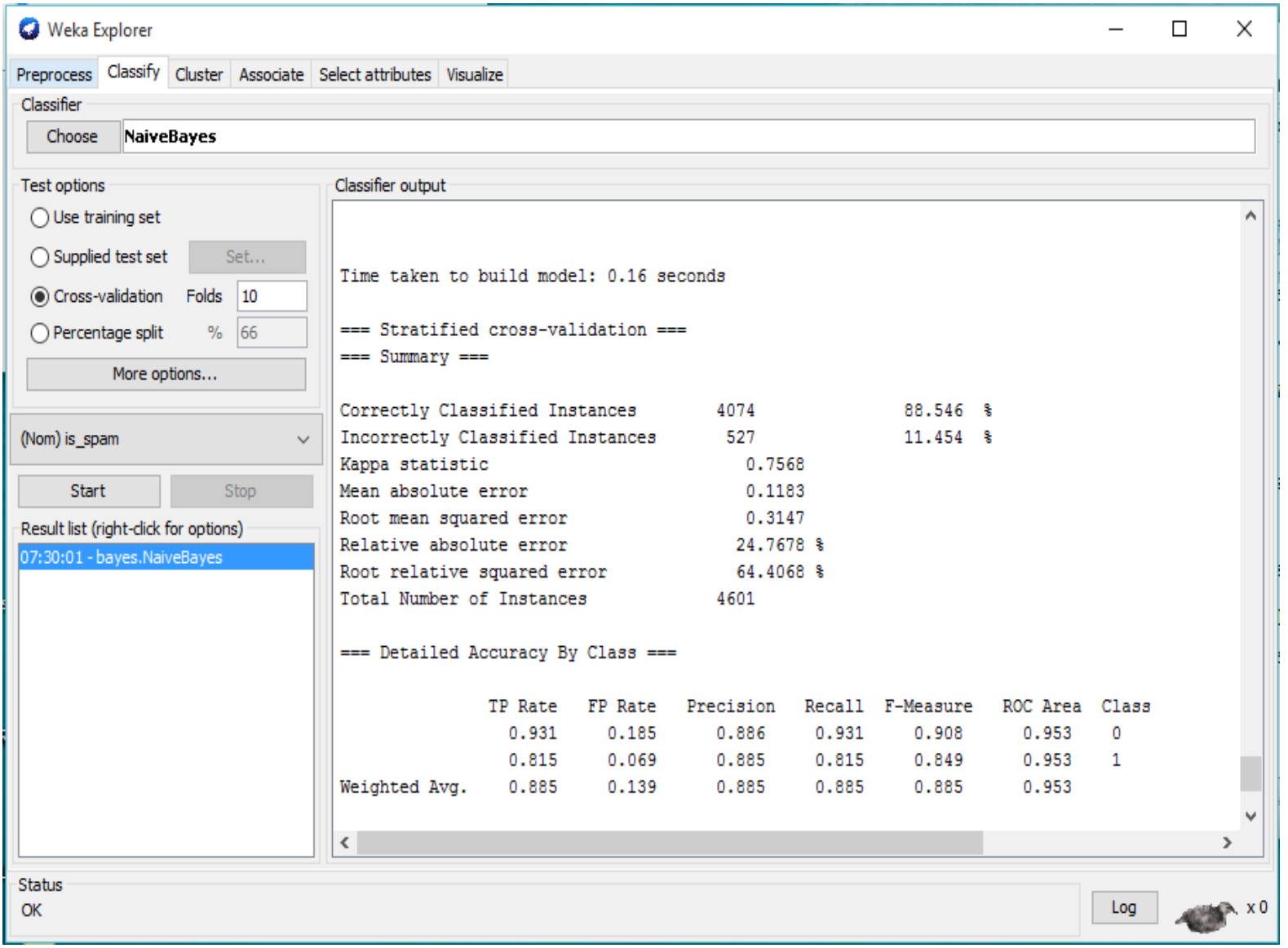


Guardamos os dados pré-processados para posterior uso. Utilizamos para tal o botão *Save*.

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*:

Fizemos um *click* no botão *Choose* e seleccionamos *classifiers* > *Bayes* > *Naivebayes*.

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 incorrectas que representa 11,454% do total de instâncias.




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

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.

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.

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.

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. E sim realmente o e-mail é classificado correctamente com uma percentagem de 100% de instâncias correctamente classificadas.

**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, carregamos o ficheiro *glass.arrf*.

- ✓ O conjunto de dados possui 10 atributos com valores numéricos.
- ✓ Os seus nomes são: RI, Na, Mg, AL, Si, K, Ca, Ba, Fe e Type.
- ✓ O atributo que correspondente à classe é "Type".

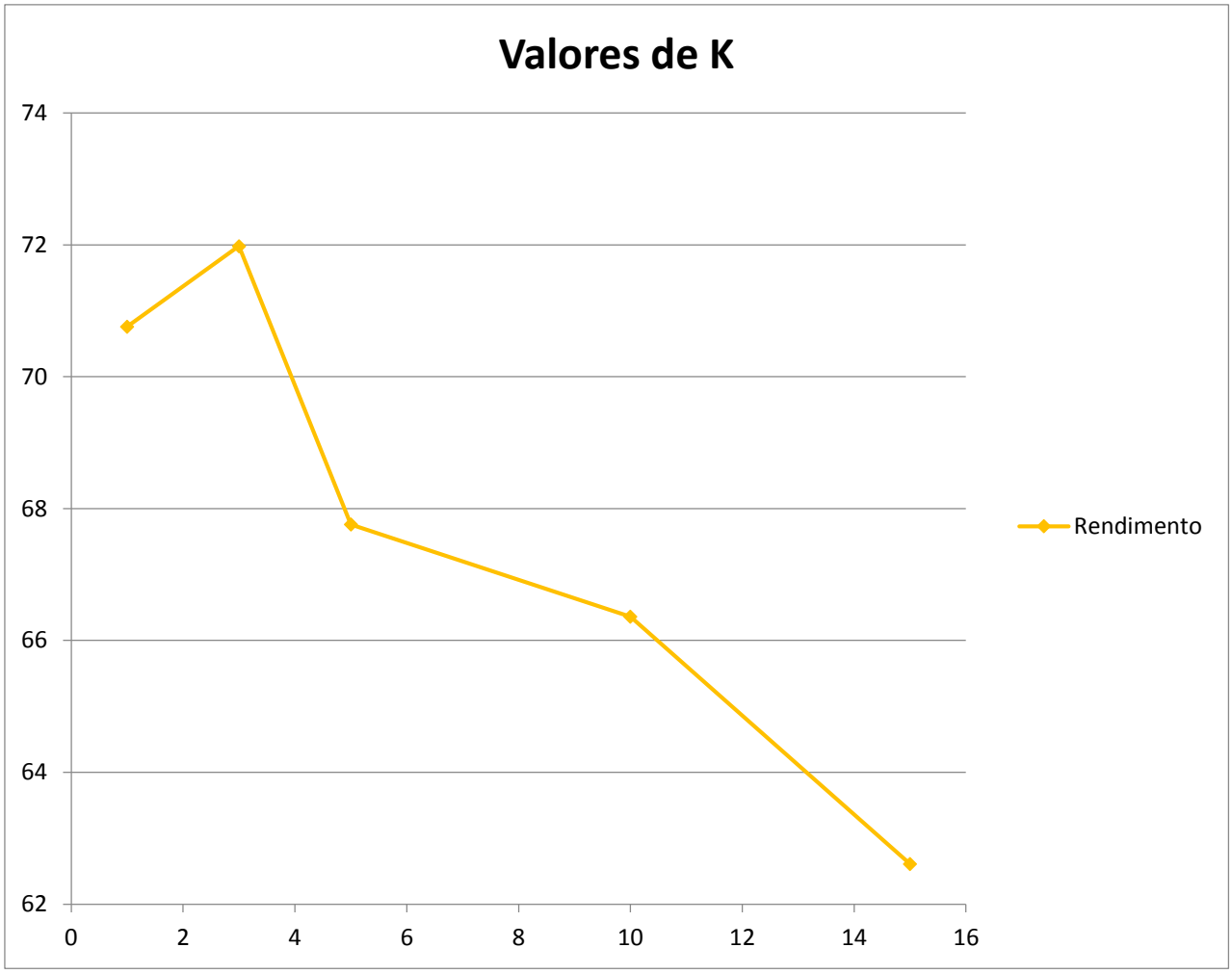
No separador *Classify*, seleccione *classifiers* > *lazy* > *Ibk* . Esta opção corresponde ao método *dos k vizinhos mais próximos* (kNN).
No painel *Test options*, seleccionamos *Use training set* e pressionamos o botão *start*.
Observe a saída do classificador e considere o seguinte: o rendimento do classificador é positivo de 100%.

Se considera uma má ideia testar um classificador kNN (k = 1) sobre o próprio conjunto de treino porque com K = 1, pode tender a causar sobre-encaixe. A votação é feita baseando-se apenas num único exemplar, o mais próximo, tornando-o muito sensível a ruídos, outliers ou até mesmo erros na etiqueta do dado. Usar um maior valor para K lhe dá maior poder discriminatório e o torna mais robusto a distorções.
Agora vamos avaliar o classificador utilizando uma validação cruzada. Executamos o classificador, utilizando o número de folhas fixado por defeito (10). O rendimento do classificador relativamente ao caso anterior é positivo: que 70,56% das instâncias foram classificadas correctamente e 29,43% delas incorrectamente.

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*.

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%.

A figura abaixo representa graficamente os resultados (valor de k no eixo x e o rendimento no eixo y).




É possível concluir algo observando o gráfico, 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.

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:

| **Percentagem De Ruido** | **Rendimento** | | |
|---|---|---|---|
| | **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%** |

Considerações:

1) O aumento do ruido na maioria das vezes baixou o rendimento associado a classificação.
2) 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. Isso porque no caso das árvores em algum ponto da árvore que é aprendida atributos irrelevantes, são escolhidos pra ramificar causando erros aleatórios a quando do 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. 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.

Construímos uma tabela mostrando o melhor subconjunto de atributos e a menor taxa de erro obtidos em cada iteração.

| **Tamanho do Subconjunto (# Atributos)** | **Atributos no "Melhor" Subconjunto** | **Taxa de Erro** |
|---|---|---|
| 9 | RI, Na, Mg, Al, Si, K, Ca, Ba, Type | 22.8972% |
| 8 | RI,Na,Mg,Al,K,Ca,Ba,Type | 22.4299% |
| 7 | RI,Na,Mg,K,Ca,Ba,Type | 21.028% |
| 6 | RI,Na,Mg,K,Ca,Type | 21.9626% |
| 5 | (RI,Mg,K,Ca,Type ou ***RI,Na,K,Ca,Type***)* | 22.8972% |
| 4 | RI,K,Ca,Type | 26.1682% |
| 3 | RI,k,Type | 35.0467% |
| 2 | k,Type | 50.9346% |
| 1 | Type | 64.486 % |

*Atributos com a mesma taxa de erro, selecionada o subconjunto em negrito e itálico.

# Conclusões

Na primeira actividade utilizamos o Weka para treinar um classificador Naïve Bayes, para efeitos de detecção de *spam.* Para tal efeito usamos o conjunto de dados *spambase.arff.*

Na segunda actividade experimentamos a classificação com o método de *k vizinhos mais próximos (kNN)* usamos o conjunto de dados *glass.arff.*

E por último examinamos um procedimento para a selecção de atributos usando como algoritmo de aprendizagem o *kNN (ibk no weka).*

Realizadas as actividades podemos constatar e de forma sumária, os seguintes aspecto:

1. O classificador Naïve Bayes realiza a inferência de facto na perspectiva probabilística, constituindo assim um bom modelo para aprendizagem, e com bom rendimento de classificação.
2. O kNN, ou o ibk se preferirmos, é um classificador intuitivo, porém conseguimos apurar que há que ter precaução com o valor K=1, dado a sua sensibilidade quando k tem este valor.
3. A selecção de atributos é um artefacto a se ter em conta sobretudo quando queremos fazer a interpretabilidade, a distinção entre atributos relevantes e os irrelevantes.

# Bibliografia

Mitchell, pg. 154 – 158, 177 – 184, 230 – 236

Witten, secção 7.1, pg. 307 – 314

Apostilas Aulas de Aprendizagem Automática: Lázaro Makili,

- ClassificadoresBayesianos2015
- AprendizagemBaseadaInstancias2015