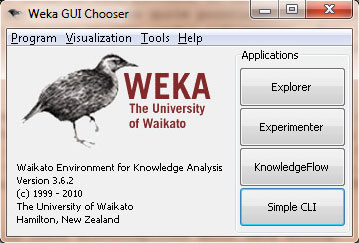
2

UNIVERSIDADE KATYAVALA BWILA

INSTITUTO SUPERIOR POLITÉCNICO

**TRABALHO DE APRENDIZAGEM AUTOMATICA**

**RELATÓRIO DA PRÁTICA 2**



4º Ano – Ciências da Computação

Período – Regular

O Docente

Dr. Lazaro Makili

Benguela, Outubro de 2015

UNIVERSIDADE KATYAVALA BWILA

INSTITUTO SUPERIOR POLITÉCNICO

**TRABALHO DE APRENDIZAGEM AUTOMATICA**

**RELATÓRIO DA PRÁTICA 2**

Elaborado Por:

Nome

Manuel Lucas Mungondi

Sérgio Kapanda Chissevo

4º Ano – Ciências da Computação.

O Docente

Dr. Lazaro Makili

Benguela, Outubro de 2015

**OBJECTIVOS**

Com a prática se pretende alcançar os seguintes 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**

Nesta prática serão utilizados dois métodos para treino de classificadores para conjuntos de dados concretos (Naive Bayes e o algoritmo KNN). Também se examinará 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.

Na mesma serão utilizados os conjuntos de dados ***Spambase*** e ***Glass*.**

O conjunto de dados

***Spambase***

**Conceito**: classificar um e-mail como spam ou não spam

**Quantidade de atributos**:57

**Valores omissos**:0

***Glass.arff***

**Conceito**: classificar um determinado tipo de vidro

**Quantidade de atributos**:10

**Valores omissos**:0

Neste trabalho abordaremos também de alguns conceitos básicos como: Operação de pré-processamento, e modelos e métodos de validação.

**Operação de Pré-processamento:**

* **A normalização** que consiste em escalonar os dados de modo a caírem em uma faixa de pequenos valores. Tem como propósito principal minimizar problemas oriundo do uso de unidades.
* **A discretização** consiste na transformação de atributos numéricos em atributos nominal, essencial se o método inteligente só manuseia atributos nominais. Existem dois tipos de discretização: *não supervisionada*, não tem em conta as classes das instâncias, e *supervisionada*, tem em conta as classes das instâncias ao criar os intervalos (uteis para problemas de agrupamento de dados, onde não se conhecem classes de saída). O principal método de *discretização não supervisionada* existente no Weka é *Weka.filters.unsupervised.attribute.Discretize*. O mesmo implementa duas variantes: *discretização com intervalos igual largura* (seleccionado por defeito) e com *intervalos de igual frequência*.

**Métodos de validação**

* **Medidas de desempenho**: é definido mediante proporções de exemplos incorrectamente classificados de entre a totalidade de exemplos de teste. No caso de problemas de classificação a medida mais comum é a taxa de erro. Existem diversas medidas de desempenho como: Taxa de erro, Taxa de êxito, Especificidade, Sensibilidade, Proporções de falsos positivos;
* **Holdout**: Consiste em dividir os dados em duas partes, um para treino e outra parte para teste. Geralmente usa-se 1/3 dos dados como conjunto de teste e o resto para treinar o algoritmo;
* **Validação cruzada**: Consiste em dividir os dados em k subconjuntos com tamanhos aproximadamente iguais, sendo utilizados k-1 subconjuntos para treinar algoritmo e o resto pra testar. A estimação da taxa de erro é feita através da média das taxas obtidas ao repartir o processo k vezes com as diferentes combinações de k-1 subconjunto de treino.
* **Matriz de Confusão**: é utilizada em classificação, possibilita uma visualização inequívoca dos resultados de um determinado modelo. Os resultados são apresentados sob forma de tabela de duas entradas (considerando problemas de apenas duas classes): uma das entradas é constituída pelas classes desejadas, a outra pelas classes prevista pelo modelo. As células, por sua vez, são preenchidas com o número de instâncias que correspondem ao cruzamento de entrada.

**Algoritmo Naive Bayes**: o classificador bayesiano representa um método de aprendizagem supervisionada, como um bom método para classificação de estatística. Classificação bayesiana fornece algoritmos de aprendizagem prática e conhecimento prévio e dados observados podem ser combinados. Classificação Bayesiana fornece uma perspectiva útil para a compreensão e avaliação de muitos algoritmos de aprendizagem. Ele calcular probabilidades explícitas para hipótese e é robusto ao ruído em dados de entrada

**Uso do classificador naive bayes**

O classificador abyesiano é utilizado como um método de aprendizagem probabilístico. O classificador Naive Bayes está entre os algoritmos mais bem-sucedido conhecido para aprender a classificar documento de teste.

**Algoritmo KNN**

**Tarefa 1**

Nesta actividade utilizaremos 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*.

Para tal usamos o weka como ferramenta, clicamos no painel *Preprocess* e carregamos o conjunto de dados Spambase, aparecendo assim uma lista completa de todos os atributos do conjunto de dados na secção “*Attributes*”, como o da figura abaixo:

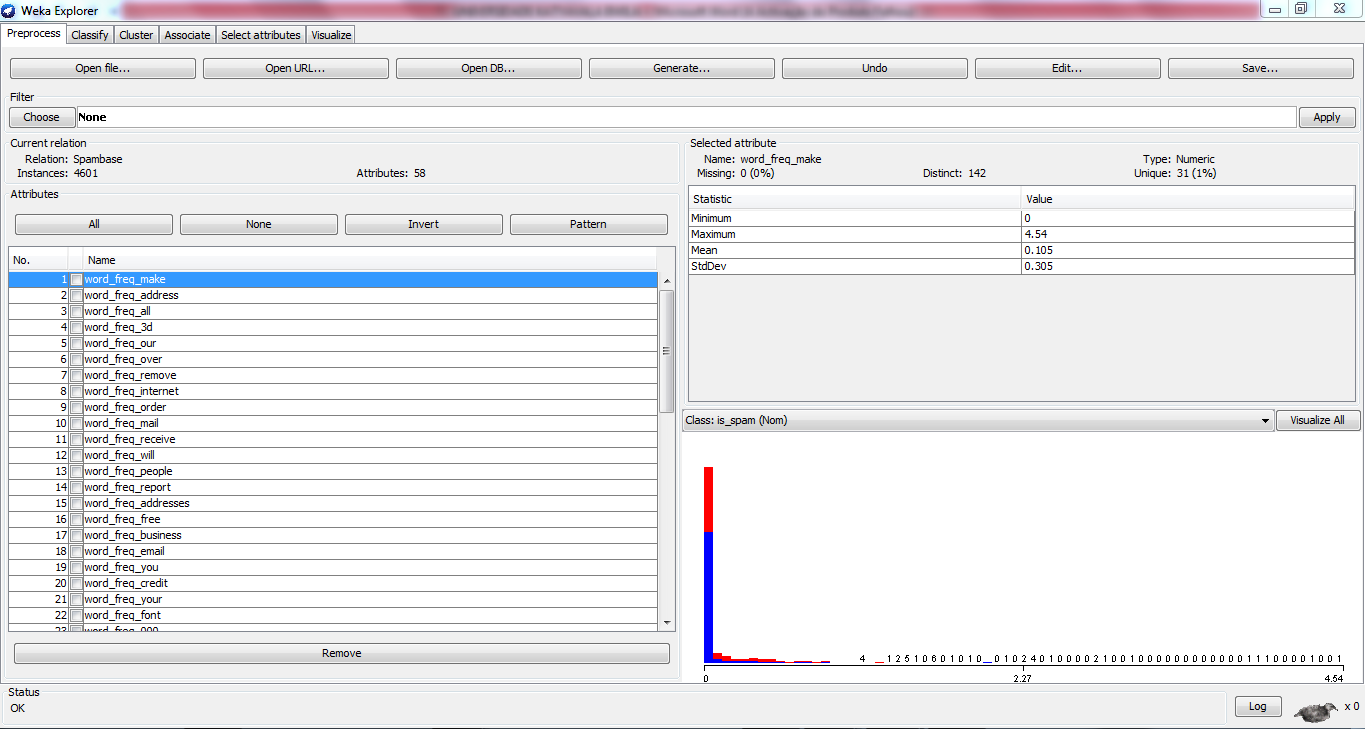
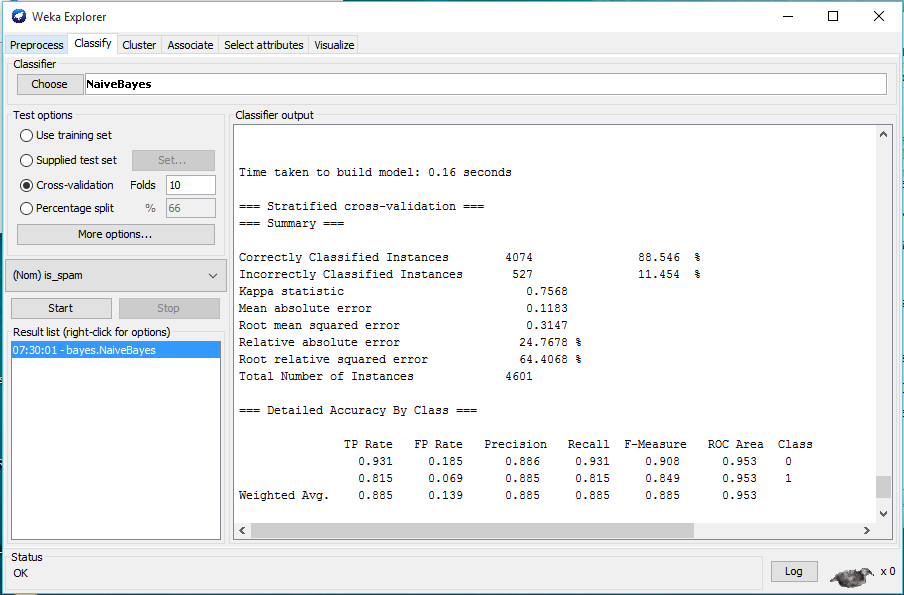


Ilustração 1

A continuação 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*. Os restantes dos atributos representam frequências relativas de várias palavras e caracteres importantes presentes em mensagens de *e-mail.*

Convertemos as referidas frequências em valores booleanos: 1 se a palavra ou caracter está presente no *e-mail*, 0 se não está presente. Para tal, selecione 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). Guardamos os dados pré-processados para uso futuros clicando no botão *save.*

Com os dados carregados e modificados, treinamos 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*: Fazemos *click* no botão *Choose* e seleccione *classifiers > Bayes > Naivebayes*, e opção de por defeito mantem-se. Verificamos que o classificador levou **0.16 segundos**, para treinar o algoritmo, onde foram são classificadas 4061 instâncias, das quais 4074 foram bem classificadas fazendo um total de 88,54% e 527 instâncias foram classificados incorrectamente totalizando 11,45%, como ilustrado na figura abaixo:



Ilustração

Examinamos o modelo produzido pelo Weka (apresentado antes da informação sobre o rendimento do classificador). Observamos as probabilidades *a priori* para cada classe, e obtivemos os seguintes resultados: Classe 0 = 0,61 e Classe = 0,39.

O algoritmo calcula a probabilidade de que um email pertença a classe *span* ou *não*, 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 já 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*): O conjunto de dados possui 4601 instâncias das quais 61% destas são classificadas com sendo *não span* e 39% são classificadas como  *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.**

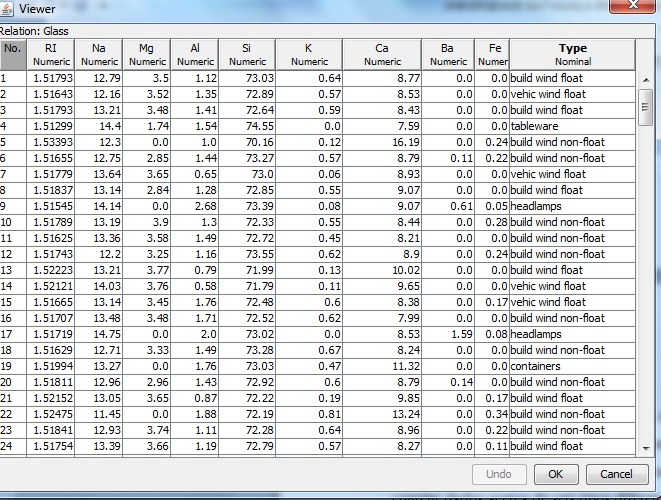
Agora, assumamos que somos *spammers* tentando burlar um sistema de detecção de *spam* baseado num classificador Naïve Bayes no sentido de classificar uma mensagem *spam* como *ham* (i.e. uma mensagem válida). Utilizamos o conjunto de dados de treino completo para treinar o classificador e apliquemos 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*. Executamos o classificador sobre este conjunto de teste. Sim realmente o e-mail foi classificado correctamente com uma percentagem de 100% (spam) de instâncias correctamente classificadas.

Abrimos o ficheiro de teste *spambase\_test.arff* num editor de texto ( o bloco de notas). Identificamos boas palavras, características da classe *não spam* e acrescente-as ao *e-mail*. Ao testar o Weka ignora este atributo e utiliza o classificador treinado previamente para predizer a etiqueta correspondente ao *e-mail*. Executamos novamente o classificador com o ficheiro de teste modificado, onde observou-se a alteração da classe atribuída, isto é, instancia foi classificada como não spam.

**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 preprocess carregamos o conjunto de dados Glass.arff, onde observamos que o mesmo possui 10 atributos, cujos nomes são: RI, Na, Mg, AL, Si, K, Ca, Ba, Fe e Type (correspondente a classe). Como mostra a figura



Ilustração

Na sequencia treinamos o conjunto de dados com o algoritmo KNN, para tal no separador *Classify*, seleccionamos *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*.

Observamos que a saída do classificador possui um rendimento de 100%, como se vê na figura abaixo.

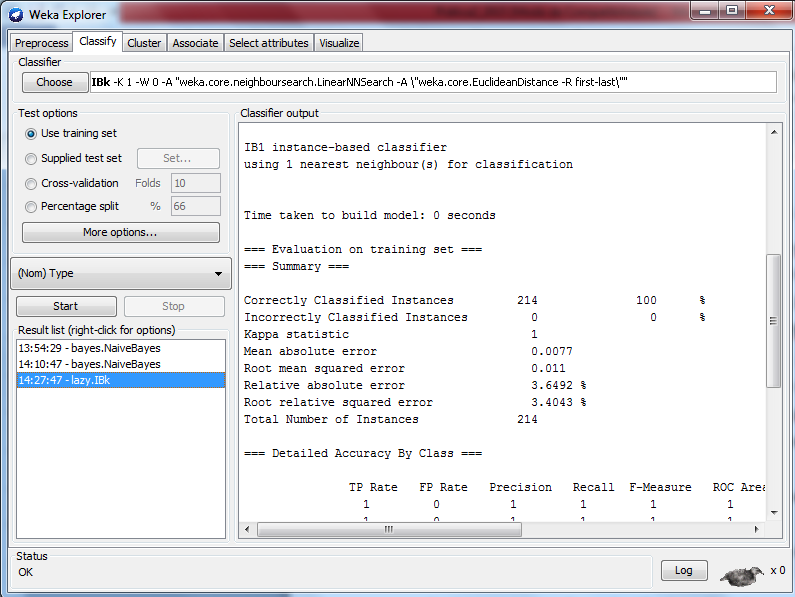


Ilustração 4

Verificamos também que é uma má ideia testar o classificador KNN, quando k=1, porque o algoritmo KNN classifica uma nova instância em função da classe mais representada dos vizinhos mais próximo, por isso quando k=1 classificará sempre em função de uma única instância.

Verificamos também que para uma classificação de teste não seja tão boa como esta, porque a taxa de erro determinada a partir dos dados utilizados ao treinar o algoritmo de aprendizagem (conjunto de teste) não é um bom indicador do seu desempenho futuro.

Na sequência avaliamos o classificador utilizando uma validação cruzada, com o número de folhas 10. Observamos que o rendimento do classificador altera para 70,56% em relação ao caso anterior, onde o rendimento era de 100%, como se vê na figura.

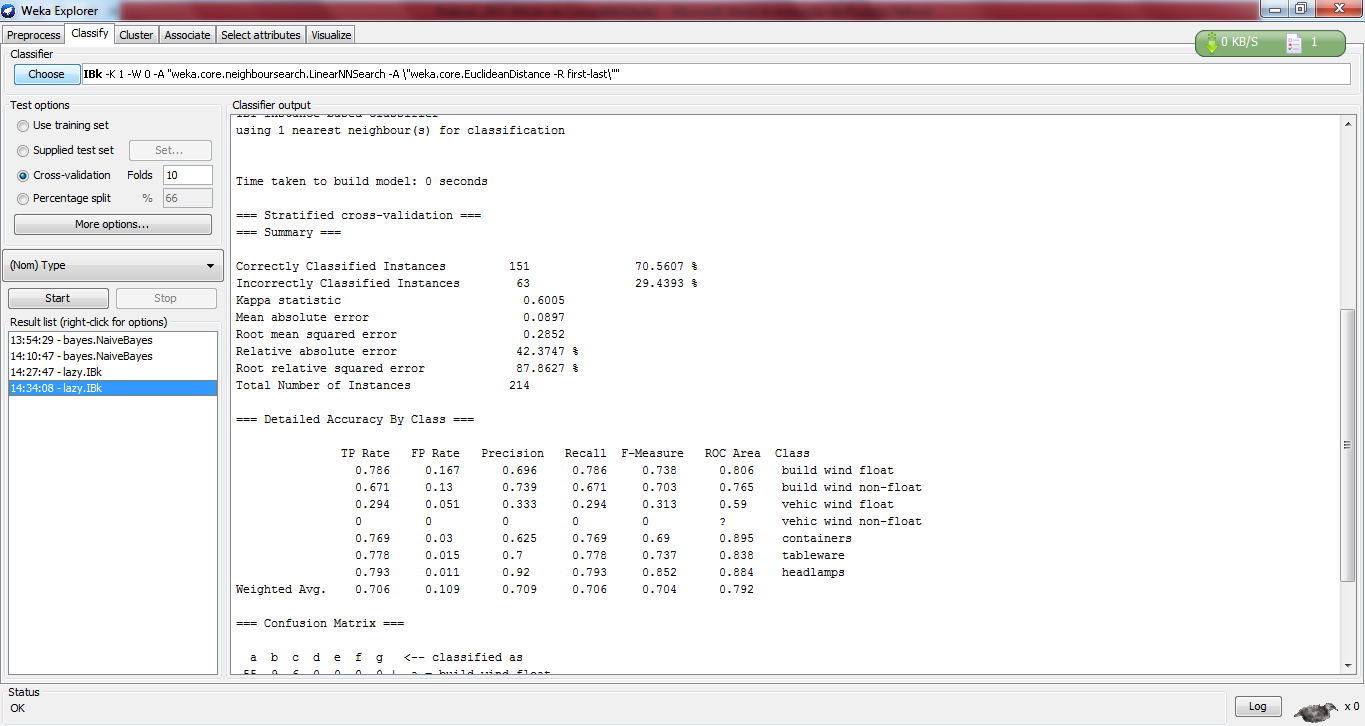


Ilustração 5

A continuação exploramos 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.

**REPRESENTAÇÃO GRAFICA**

Pudemos verificar que 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, rendimento=70,56%); (K=3, rendimento=71,96%); (K=5, rendimento=67,76%) (K=10, rendimento=66,36) (K=15, rendimento=62,61); com excepção da transição de K=1 para K=3 ai o rendimento do classificador aumentou 1,4%.

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.

|  |  |  |  |
| --- | --- | --- | --- |
| **% De ruido** | **Rendimento do classificador com K=1** | **Rendimento do classificador com K=3** | **Rendimento do classificador com K=5** |
| 0% | 100% | 80,84 | 78,97 |
| 10 | 99,53 | 75,70 | 73,83 |
| 20 | 99,53 | 71,49 | 67,28 |
| 30 | 99,53 | 60,74 | 58,87 |
| 40 | 99,53 | 52,80 | 47,19 |
| 50 | 99,53 | 57,94 | 49,53 |

Observamos que o aumento da percentagem do ruido altera os diferentes valores de k vizinhos, isto é, quanto maior for o valor percentual do ruido, menor será o valor do rendimento em função dos valores de k.

**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

| **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 | RI, Mg, Na, Al, K, Ca, Ba, Type | 0,0704, 77,5701% |
| 7 | RI, Na, Mg, K, Ca, Ba, type | 0,0665, 78,972 |
| 6 | RI, Na, Mg,Ca, Ba, Type | 0,0756 77,5701 |
| 5 | RI, Na, Mg, Al, Type | 0,0781 74, 7664 |
| 4 | RI,Ca, Ba, Type | 0,1 66,8224 |
| 3 | RI, Na, Type | 0,1271 57,0093 |
| 2 | RI, Type | 0,1559 47,1963 |
| 1 | type | 0,2106 35,514 |