# Data Mining Coursework – Part B

Peter Smith

## Part I: Feature Ranking

Feature ranking involves testing each feature according to a metric and eliminating the features that don't reach a certain threshold. There are several methods to achieve this: including Linear Regression and the Information Gain and Correlation Attribute Evaluators.

Linear regression is a function that generates an equation that can be used to identify the correct class. This can also be used for multi-response linear regression, which in some cases is superior. Within the generated equation, we can judge each attribute's importance using the coefficient used alongside each attribute. In order to this properly use linear regression to rank the attributes, we need to normalise the dataset in order to bring each attribute's values into the same range. This means that the coefficients become an indicator for the attribute's importance. This should be done anyway, as most algorithms assume the data has been normalised already.

Within Weka, there are several ways to select the attributes to use, and this includes several which rank the attributes using different metrics. In the 'Select Attributes' tab, both 'Info Gain' and 'Correlation' attribute evaluators can be found, and both use the 'Ranker' search method. This search method takes the evaluation from the evaluator and uses it to rank them.

The 'Info Gain' evaluator evaluates each attribute on the 'information gained' from them. Each attribute's information gain is represented by a number between 1 and 0, which is then ranked and presented to the user. This generated the 'Info Gain' list and rankings in Table 1.

The 'Correlation' evaluator measures each attribute's correlation with the class. This is done using Pearson's correlation coefficient. These coefficients are then used to rank the attributes, and this generates the rankings seen in Table 1. Before anything was executed, any records with missing values in the dataset were deleted. This was to minimise any anomalies that may occur.

*Table 1. Feature ranking using several methods*

| Attribute | Linear Regression | | Information Gain | | Correlation | |
|---|---|---|---|---|---|---|
| | Coefficients | Ranking | Information Gain | Ranking | Correlation Coefficient | Ranking |
| **Sample Code** | | 9 | 0 | 10 | 0.0847 | 10 |
| **Clump Thickness** | 0.5717 | 2 | 0.457 | 8 | 0.7148 | 6 |
| **Uni Cell Size** | 0.3943 | 3 | 0.693 | 1 | 0.8208 | 3 |
| **Uni Cell Shape** | 0.2816 | 6 | 0.67 | 2 | 0.8219 | 2 |
| **Adhesion** | 0.1502 | 8 | 0.462 | 7 | 0.7063 | 7 |
| **Epithelial Cell Size** | 0.185 | 7 | 0.525 | 5 | 0.691 | 8 |
| **Bare Nuclei** | 0.8164 | 1 | 0.596 | 3 | 0.8227 | 1 |
| **Bland Chromatin** | 0.3436 | 4 | 0.55 | 4 | 0.7582 | 4 |
| **Normal Nucleoli** | 0.3351 | 5 | 0.48 | 6 | 0.7187 | 5 |
| **Mitoses** | | 9 | 0.199 | 9 | 0.4234 | 9 |

Within the equation generated using linear regression, neither the ‘Sample Code’ nor ‘Mitoses’ attributes were featured at all, meaning that these attributes aren’t influential enough to be in the equation. I made the decision to rank these at joint 9th place. When viewing these rankings, it’s important to note that the top eight attributes (out of ten) are always the same – the sample code and mitoses attributes offer only a very small amount importance and are disregarded entirely in the linear regression equation.

There are also some attributes which are consistently high in these rankings – for example, Uni Cell Size and Bare Nuclei are always in the top 3, and Bland Chromatin is consistently in fourth place. We can also see that Normal Nucleoli is consistently ranked fifth or sixth, suggesting that it is important enough to be considered, but is not the most influential attribute. In a similar way, Adhesion and Epithelial Cell Size are both consistently towards the bottom of the scale. That said, Epithelial Cell Size was ranked higher in terms of information gain, at rank 5. Uni Cell Shape is ranked very highly by the attribute evaluators but slightly lower by the linear regression equation. Likewise, the ‘Clump Thickness’ attribute is ranked very highly by the equation but slightly lower by the attribute evaluators.

If we look at the values used to determine this ranking, we can start to identify where there are stark differences between the rankings. Viewing the linear regression coefficients, we can see that the coefficient values of the top two attributes (Bare Nuclei and Clump Thickness) are significantly greater than the rest, at 0.82 and 0.57. The next greatest coefficient is 0.39, and the next few coefficient values start to get closer to one another. The first 6 rankings are all greater than 0.28, with the rest of the attributes being less than 0.19.

The information gain metric notably judges the ‘sample code’ attribute to have 0 information gain. The values for the top two ranks are close, at 0.693 and 0.67. Likewise, the values for ranks 3-8 are also reasonably close, with a range of just under 0.14, ranging from 0.596 to 0.457. This suggests that these attributes are consistently good, and a ranking of 7 might not be significantly different from a ranking of 4. In a similar way, the correlation evaluator evaluates the top eight attributes to have values in a range of 0.1317, from 0.8227 to 0.691. In both, the mitoses attribute is given an information gain value of 0.199 and a correlation coefficient of 0.4234, which does suggest that this attribute may have some influence, however this is significantly lower than the other attributes, and may not be necessary. It is likely that the ‘sample code’ attribute bears no real influence on the class, based on the rankings it is given. This makes sense given that it is an ID number.

Therefore, utilising the rankings generated by these methods, the rankings in Table 2 show my interpreted version of the rankings.

*Table 2. Interpreted Rankings*

| Attribute | Ranking |
|---|---|
| **Sample Code** | 10 |
| **Clump Thickness** | 5 |
| **Uni Cell Size** | 2 |
| **Uni Cell Shape** | 3 |
| **Adhesion** | 8 |
| **Epithelial Cell Size** | 7 |
| **Bare Nuclei** | 1 |
| **Bland Chromatin** | 4 |
| **Normal Nucleoli** | 6 |
| **Mitoses** | 9 |

## Part II: Classifier and Dataset Comparison

For my classifiers, I chose the BayesNet and JRip classifiers. BayesNet is a classifier which implements a Bayes Network learning algorithm. JRip is a rules-based classifier that incrementally learns propositional rules that can be used to classify instances. It also has an optimisation function which narrows the ruleset down to fewer rules.

For the modified dataset, I removed the SampleCode and Mitoses attributes. Part I showed that Sample Code wasn't very important at all, and whilst Mitoses did show some low level of influence, it was a significantly lower level of influence when compared to the other eight attributes. Both datasets were also normalised, with any records with missing data removed. Other attributes could have been removed, such as Adhesion and Epithelial Cell Size, which were ranked consistently low, but the values given alongside the rankings suggest that there is still importance to these, as the values are relatively similar to the higher-ranked attributes.

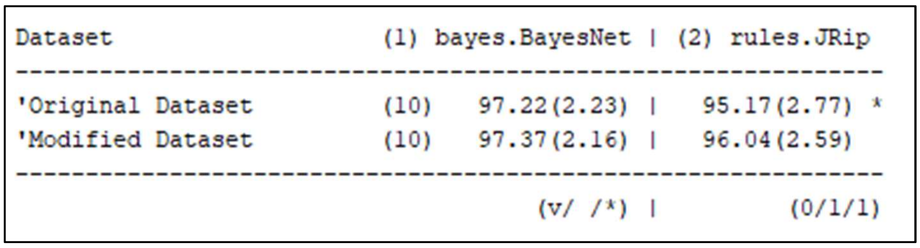
After each classifier was executed, one of the resulting statistics is 'Correctly Classified Instances', which represents how many instances were successfully classified by the model. This value can be used to generate the percentage of correctly classified instances. This value will be used to represent a model's accuracy. When classified through Weka Explorer, the BayesNet and JRip models generate the values seen in **Error! Reference source not found.**.

*Table 3. Accuracy of BayesNet and JRip classifiers on the original and modified datasets.*

| | **Original Dataset** | | **Modified Dataset** | | | |
|---|---|---|---|---|---|---|
| **Classifier** | **Correctly Classified Instances (out of 683)** | **Accuracy** | **Correctly Classified Instances (out of 683)** | **Accuracy** | **Increase in Correctly Classified Instances** | **Added Accuracy from Modified Dataset** |
| **BayesNet** | 664 | 97.2182% | 665 | 97.3646% | 1 | +0.1464% |
| **JRip** | 650 | 95.1684% | 657 | 96.1933% | 7 | +1.0249% |
| **Difference between classifiers** | 14 | 2.0498% | 8 | 1.1713% | | |

We can see in **Error! Reference source not found.** that the BayesNet classifier performed better than the JRip classifier in both cases, though the accuracy was high on all tests – all above 95% accuracy. Interestingly, modifying the dataset led to an increase in the accuracy of both classifiers.

In order to judge whether these differences are significant, t-tests will be run. Using Weka Experimenter, it's possible to perform t-tests, which can be used to determine whether two classifiers or datasets are significantly different. To do this, we run an experiment through the experimenter and then perform a 'Paired T-test (corrected)' on the results , which generate the results seen in Figure 1 and Figure 2. This was testing for a significance of 0.05, which means any conclusions have a confidence level of 95%. The field that was compared was the percentage of correctly classified instances – marked above as the 'accuracy'.



```
Dataset                   (1) bayes.BayesNet | (2) rules.JRip
--------------------------------------------------------------
'Original Dataset         (10)   97.22(2.23) |   95.17(2.77) *
'Modified Dataset         (10)   97.37(2.16) |   96.04(2.59)
--------------------------------------------------------------
                                     (v/ /*) |        (0/1/1)
```

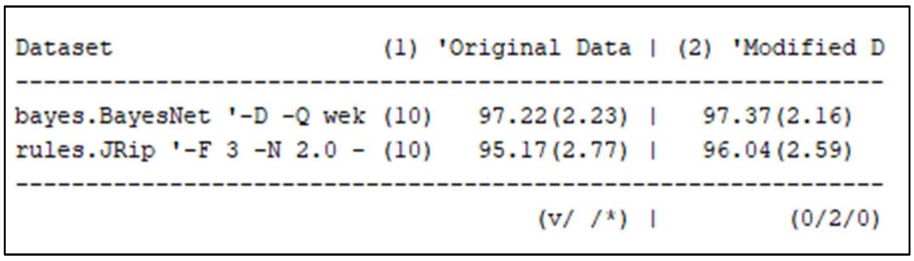
*Figure 1. Classifier T-tests.*

The important output of this table is (0/1/1), which shows us how statistically different JRip is from BayesNet. This set of symbols is interpreted as:

(J significantly greater than B / J not significantly different from B / J significantly less than B)

Where J is JRip's mean and B is BayesNet's mean. Looking again at the result of (0/1/1), we see that there is one instance where JRip's mean is not significantly different from BayesNet's mean, and there is one instance where it is significantly less than BayesNet's mean. We can look next to the values themselves for either a 'v' (greater than BayesNet), a blank space (equal to BayesNet) or a * (less than BayesNet) to identify which instance is which.

We see a blank space next to the values for the modified dataset. This means that when applied to this dataset, the BayesNet and JRip models are not significantly different from each other. There is a * symbol next to the Original Dataset, which indicates that JRip's performance is significantly worse than BayesNet's, when applied to the original data.



```
Dataset                   (1) 'Original Data | (2) 'Modified D
--------------------------------------------------------------
bayes.BayesNet '-D -Q wek (10)   97.22(2.23) |   97.37(2.16)
rules.JRip '-F 3 -N 2.0 - (10)   95.17(2.77) |   96.04(2.59)
--------------------------------------------------------------
                                     (v/ /*) |        (0/2/0)
```

*Figure 2. Dataset T-tests.*

Figure 2 represents the same comparison with the axes swapped. Whereas Figure 1 compared JRip to BayesNet, Figure 2 compares the Modified Dataset to the Original Dataset.

Looking underneath the Modified Dataset's performance, we see (0/2/0), which shows that both instances are not significantly different to the original data. This indicates that even though some attributes were changed, the results are not significantly different when using either the BayesNet or JRip classifiers.

This could be because the Mitoses attribute wasn't very influential, and so by removing it there is only a limited affect on the dataset. Additionally, there were other attributes which were consistently ranked low which could have been removed to potentially have a greater effect on the accuracy.

We see that the BayesNet classifier performs better, giving a higher accuracy than the JRip classifier. Whilst this difference was only significant for the Original Dataset, this does show that the BayesNet classifier is significantly better for some datasets.

## Part III: Equal-Width Binning

The equal-width binning strategy involves splitting up numeric attributes into a certain number of bins. To do this, the range of values is split equally into the specified number of bins. For example, in a range with a minimum of 20 and a maximum of 40, splitting it into four equal-width bins would generate the following bins:

- <=25
- >25 and <=30
- >30 and <=35
- >35

This strategy is a form of discretization – a process of transforming the data from numeric vales into symbols, or into a small number of discrete values. This makes it easier to compare the values. In order to use the equal-binning strategy, we need to use unsupervised discretization – it is unsupervised as we do not need to look at the values in order to split the instances into equal width bins. Splitting it up produces a modified dataset, as there will now be nominal attribute values instead of numeric ones. The better classifier that was identified in Part II was the BayesNet classifier.

*Table 4. Comparing the number of bins in the equal-binning strategy*

| Number of Bins | Accuracy when using BayesNet |
|---:|---:|
| 1 | 65.0073% |
| 2 | 95.754% |
| 3 | 96.3397% |
| 4 | 96.6325% |
| 5 | 97.511% |
| 6 | 97.511% |
| 7 | 97.511% |
| 8 | 97.3646% |
| 10 | 97.3646% |
| 20 | 97.3646% |
| 50 | 97.3646% |
| 100 | 97.3646% |
| 250 | 97.2182% |
| 400 | 97.2182% |
| 600 | 96.9253% |
| 650 | 96.9253% |
| 683 | 96.9253% |

In order to identify the best number of bins, several tests were carried out, as shown in Table 4. As before, the dataset was normalised and any records with missing values were deleted before any discretization took place. Initially the default of 10 bins was tested, and then values either side of 10 were tested to find a maximum accuracy. Interestingly, the accuracy was identical for several different tests – for example, 10, 20 and 100 bins all generated an accuracy of 97.3646%. This could be because using this strategy transforms the numeric data into discrete bins, and this might make the accuracy be one of several discrete values.

Whilst using one bin performed poorly, the accuracy does increase rapidly as more bins are added, reaching a peak at 5 bins, with an accuracy of 97.511%. This stays the same until 8 bins, which has a slightly decreased accuracy of 97.3646%, which appears to remain constant until at least 100 bins. The accuracy begins to decrease further, decreasing to 97.2182% at 250 and 400 bins. As the number of bins approaches the number of instances, we decrease again to 96.9253%, which remains constant until the number of instances itself. This gradual decrease in accuracy could be an indicator that the classifier may be starting to overfit the data.

For the dataset and the BayesNet classifier, the best numbers of bins for this strategy are 5,6, or 7, giving an accuracy of 97.511%. This is compared to the accuracy of BayesNet in Part II in Table 5.

*Table 5. Comparing the Discretized dataset with the Original and Modified datasets from Part II*

| | **BayesNet + Discretized** | **BayesNet + Original** | **BayesNet + Manual Modification** |
|---|---|---|---|
| **Accuracy** | 97.511% | 97.2182% | 97.3646% |

This table shows us that discretizing the dataset with the equal-binning strategy and the optimal number of bins performs better than both the original and modified datasets. The equal-width binning gives a better performance than when no binning is used. It could be worth noting that depending on the number of bins, the accuracy could have been the same as the other datasets, or worse than the other datasets.

One way to explain this could come from looking at the classifier. BayesNet is a classifier that uses a Bayesian belief Network algorithm, which is used to compactly represent a joint probability distribution. A BayesNet algorithm has two parts – one part showing the causal relationships between random variables and a probability part which shows the strengths of those relations.[1] By discretizing the values we reduce the number of variables, which may mean that the BayesNet algorithm has an easier time and is able to be more effective. This is supported by the fact that as more bins are introduced, the accuracy slowly decreases – as soon as the algorithm has enough variables to be useful – around 5 bins – it is the most effective it can be, but as more bins/variables are added, the algorithm slowly gets less effective.

It's also worth noting that if a t-test is run testing the discretized dataset, there is not a significant difference between the discretized dataset's performance and the original dataset's performance, as seen in Appendix A.

Whilst discretization and the equal-width binning strategy has increased the accuracy, there are some flaws with the strategy. Equal-width binning can lead to uneven distribution of data, which could mean that some discrete bins have far more data than others. This could lead the classifier to rely on some bins more than others, which could decrease the accuracy.

This can be fixed with the use of equal-frequency binning, which ensures an even distribution of data in its bins but is more difficult to understand. When using this strategy, if more data is added the distribution may not be even anymore. Both of these strategies do not consider the instance's

[1] S. S. Fatima, Class Lecture, Topic "Bayesian Belief Networks" 19COC101, Department of Computer Science, Loughborough University, Loughborough, Nov. 12, 2019.

classes when splitting them into bins. This means that it can be unsupervised which is quicker and easier to execute, but it also means that some of the splitting may not be optimal. It can cause some bad boundaries.

Discretization has some benefits and makes each dataset potentially quicker to execute by making the numeric values into discrete symbols. This makes the classification process quicker and discrete values are easier to compare. Algorithms also assume that data has been normalised which isn't always the case – discretization makes this not a problem.

Using the equal-binning strategy is easy to understand and with bins of equal width, it can split data into them relatively quickly. Identifying the right number of bins is important for this strategy to be effective, and this may take some time and trial-and-error.

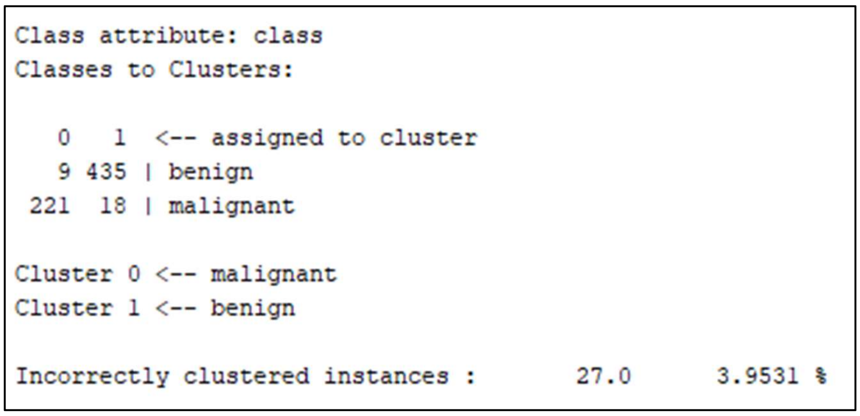
I would recommend using equal-width binning, because it can increase the performance of a dataset, however some experimentation is needed to find the best number of bins, and this could also depend on which classifier is used. It works well with BayesNet, but it may perform differently when paired with a different classifier or dataset. Before being discretized, the dataset was already normalised – if this isn't the case, and the dataset hasn't been prepared well, some anomalous values could affect the bins and lead to a extremely uneven distribution, as this would add an extreme maximum or minimum which would lead to bad boundaries.

This strategy also only works well if all of the data is numeric and working with nominal data may not work as intended. With some other datasets and classifiers the effectiveness may not work as well but it is potentially worth the time and effort to experiment with using it, as it can improve the performance and accuracy of classification.

## Part IV: K-means Clustering

K-means clustering is a method where the data's similarity to each other is used. The unclustered data of one of the attributes is used. First, centroids are randomly placed on this data a specified number of times. These are initially used to assign instances to certain clusters based on the distance to each centroid. These randomly placed centroids are then replaced by the mean point of each cluster. Each instance is then re-assigned based on these mean points – these new centroids. This can use multiple attributes to justify the clusters.

K-means clustering can be used to create a number of clusters that can then attempt to classify the data. With the SimpleKMeans clustering algorithm in Weka, with two clusters defined (one for each class in our dataset), we can then test the data using the 'classes to clusters evaluation'. This compares the clusters' ability to classify the data with the true classes. This gives us the confusion matrix that can be seen in Figure 3. Before clustering, the dataset was normalised and any records with missing data were deleted.



```
Class attribute: class
Classes to Clusters:

   0   1  <-- assigned to cluster
   9 435 | benign
 221  18 | malignant

Cluster 0 <-- malignant
Cluster 1 <-- benign

Incorrectly clustered instances :       27.0      3.9531 %
```

*Figure 3. Confusion Matrix for K-means Clustering*

As we can see in Figure 3, this clustering incorrectly clustered 27 instances, meaning it correctly clustered instances with an accuracy of 96.0469%. We can see that out of the 444 instances in the 'benign' class, 9 (2%) were clustered incorrectly, whilst 18 out of 239 'malignant' instances (7.5%) were clustered incorrectly – a higher proportion of errors. This is a good accuracy and this high performance does suggest that the clusters are successful and largely do manage to correctly classify the instances. This is a good accuracy, however compared to other classification methods it isn't perfect. Table 6 compares the performance of the clustering algorithm to the other classifiers used in this coursework on the same original dataset.Table 1

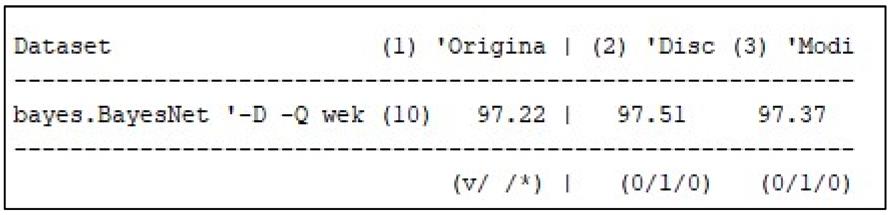
*Table 6. Comparing accuracy of different classification methods*

| | **Clustering algorithm** | **BayesNet + Original** | **JRip + Original** |
|---|---|---|---|
| **Accuracy** | 96.0469% | 97.2182% | 95.1684% |

Table 6 demonstrates that k-means clustering does perform at the same level as other classification algorithms, performing better than the JRip classifier, but worse than the BayesNet classifier. This does indicate that clustering can be used in good faith to cluster accurately. It isn't the best classifier and may perform differently on different datasets and with different parameter configurations, but clustering can perform better than some classification algorithms, and so it is worth trying and testing when classifying data, as it may perform better.

There are some flaws with K-means clustering that could affect the distribution of errors and overall effectiveness of the algorithm. If the algorithm is only run once, then the placement of the randomly placed clusters could also affect the final clusters negatively. This isn't frequent, but the centroids could lead to a local minimum, where a cluster of data is split into two clusters due to specifically placed clusters. To mitigate this, the algorithm needs to be run multiple times. By default, the Weka algorithm has a maximum number of iterations set at 500. K-means clustering also struggles to work with nominal attributes as it only works with numerical attributes. Clustering is worth using as it could perform better than alternative classification algorithms but it could depend on the dataset whether it is ideal to use clustering.

## Appendix A – Discretized T-test



```
Dataset                    (1) 'Origina | (2) 'Disc (3) 'Modi
-------------------------------------------------------------
bayes.BayesNet '-D -Q wek (10)   97.22 |   97.51     97.37
-------------------------------------------------------------
                               (v/ /*) |   (0/1/0)   (0/1/0)
```