# University of Reading

**Course:** BSc. Computer Science
**Module:** Data Mining
**Module Convenor:** Dr Giuseppe Di Fatta
**Student:** Vinícius de Oliveira Silva – **Student Number:** 24023627

# Data Mining Coursework Report

## 1. Introduction:

Analysing data has become an impressively important activity nowadays. In the information era, more and more data is generated every day, and quite often, this huge amount of data hides information that might be extremely useful in a wide range of fields.

Hospitals, for instance, can analyse their patient records and spot patterns that provide valuable information about how a certain disease spreads. Supermarkets might make detailed discoveries about their customers consume behaviours and use this information to improve their services. The list of applications of data analysing is endless and shows that this is a topic which has potential to change the society as we know.

However, the problem of how to analyse the available data is not simple. Selecting the best technique to apply in each scenario is a task that intrigues even the most experienced data scientists. There are many algorithms available in the literature and only a deep understanding of how they work can enable an individual to efficiently tackle a data analysis problem he has in hands.

In this coursework, three different data mining problems are going to be approached and evaluated. The algorithms used to solve the problems are going to be presented and discussed. The achieved results for each one are also going to be shown and their quality will be measured either analytically or by using precise techniques.

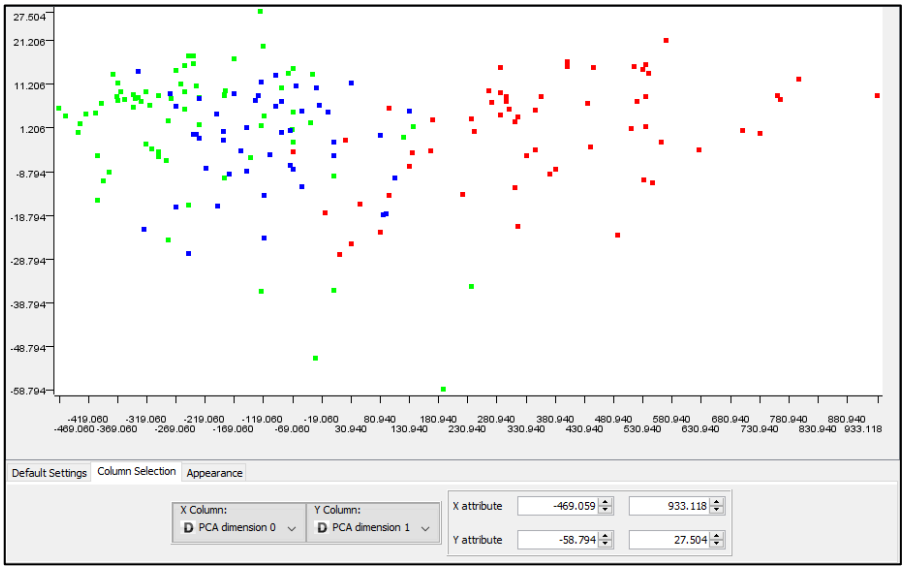
## 2. Data Mining Tasks:

- ### Task 1 – Data Exploration and Clustering:

#### Task 1.1 – Clustering without normalisation:

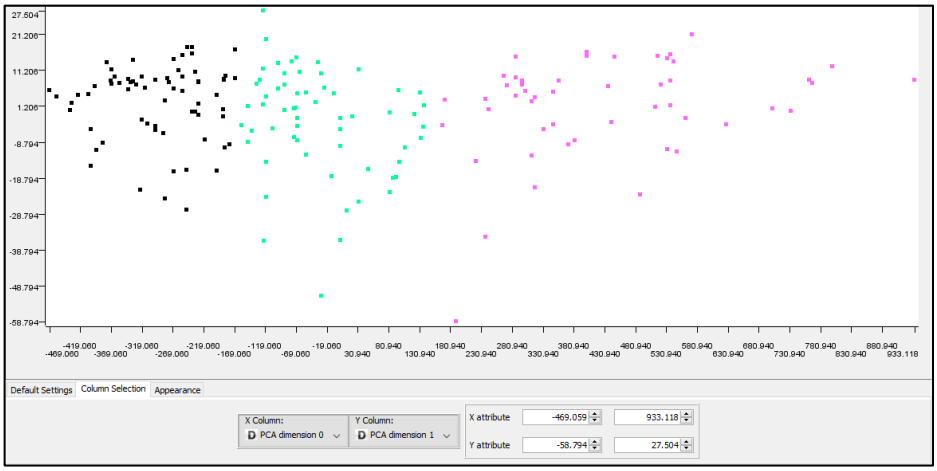
In this task it is required that three clusters be generated from a data set without applying normalisation to its attributes. The data set was obtained from a chemical analysis of wines grown from three different cultivars. The analysis determined the quantities of 13 chemical constituents found in each wine and so each data record contains the cultivar ID (1, 2 or 3) and 13 numerical attributes.

In order to tackle this problem, first it was required to apply principle component analysis to the data in order to extract the dimensions with larger variance and visualize the disposal of the data in a 2D plot. The generated plot is presented below:




Plot 1.1.1 – 2D Plot of Data PCA without normalisation
Subtitle:
Red: Cultivar 1
Green: Cultivar 2
Blue: Cultivar 3

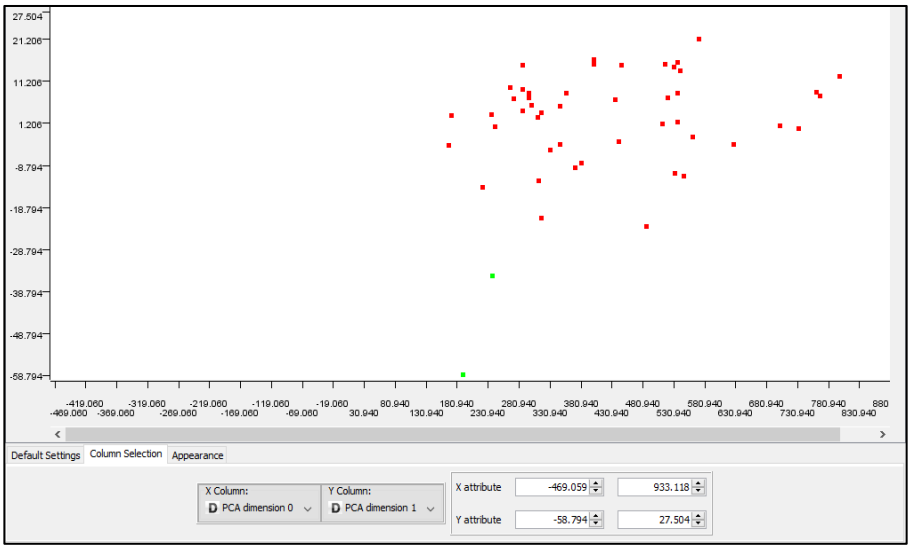
After generating the 2D plot for the two principal components of the data, the coursework specification requires us to apply a clustering algorithm to the data set and generate three partitions. Another 2D plot was generated, but this time, the colours are associated with the cluster ID, not the class of each entry.



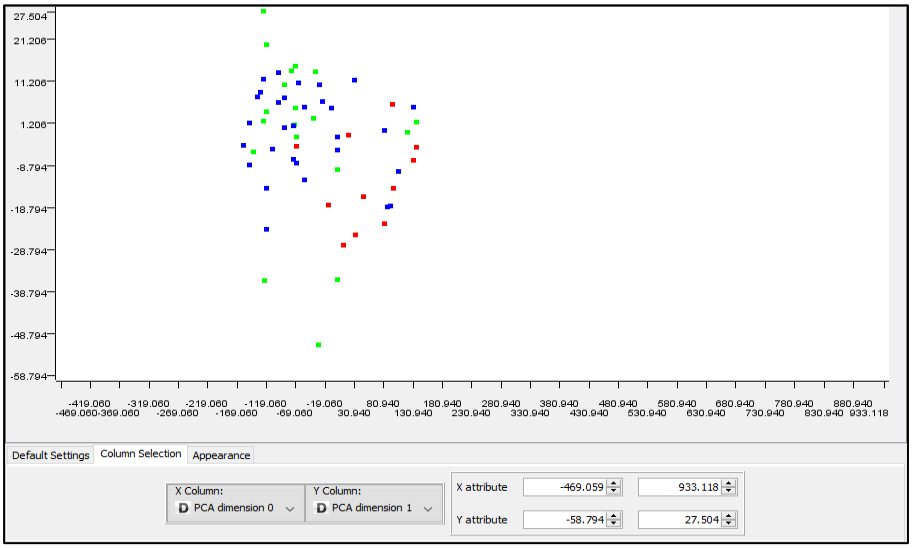
Plot 1.1.2 – 2D Plot of Data Clustering without normalisation
Subtitle:
Pink: Cluster 1
Green: Cluster 2
Black: Cluster 3

The algorithm used to generate these clusters is known as k-Medoids. It works by randomly electing k (a pre-defined number of) data points as "medoids", which are representative points of the clusters. All the other points are assigned to the cluster which has the closest medoid. The algorithm uses a distance matrix as input in order to determine the distances between every data point to any other.
All the non-medoids input data points are tested as representative points for each cluster and if it reduces the cost, this point replaces the old medoid as a representative point to that cluster. This process is done until no more cost reduction is possible.
In order to determine the dissimilarity between two points, the Euclidean distance was used. This measure was chosen because it is simple and the data is not high dimensional.

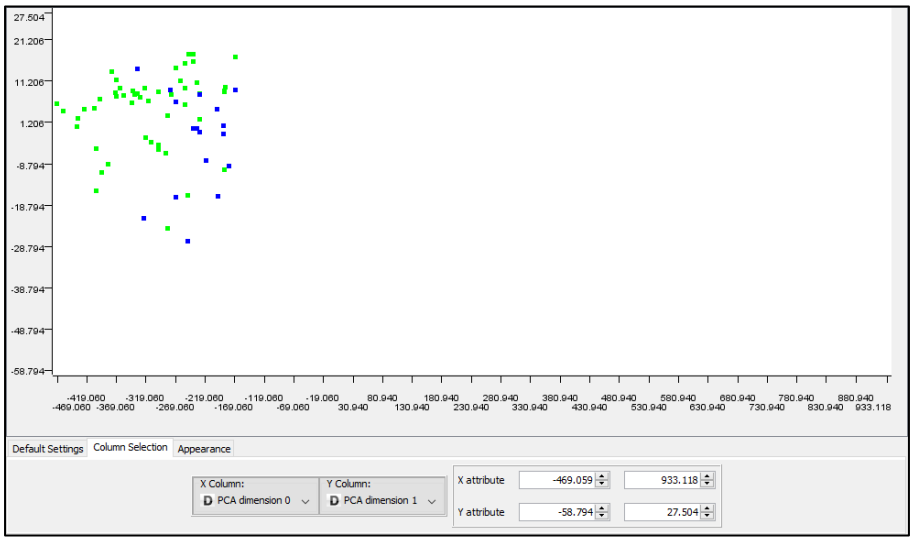
In order to visually verify the distribution of class labels in each cluster, three more 2D plots were generated. Each one describes a cluster and the plotted points were coloured accordingly to their class (the same colours of plot 1 were used). These plots offer a visual insight of how accurately the clustering describes the available data.




Plot 1.1.3a – 2D Plot of Cluster 1 without normalisation
Subtitle:
Red: Cultivar 1
Green: Cultivar 2



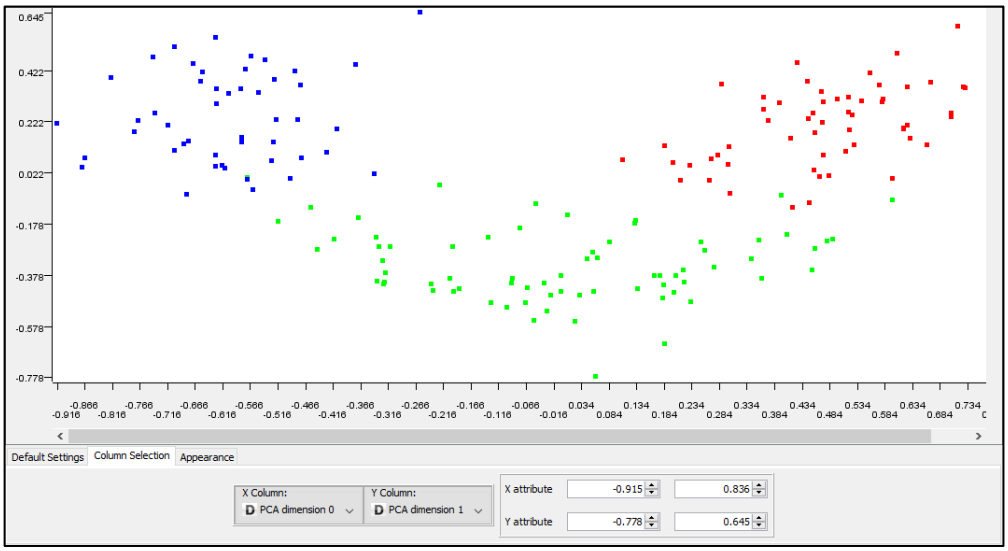

Plot 1.1.3b – 2D Plot of Cluster 2 without normalisation
Subtitle:
Red: Cultivar 1
Green: Cultivar 2
Blue: Cultivar 3




Plot 1.1.3c – 2D Plot of Cluster 3 without normalisation
Subtitle:
Green: Cultivar 2
Blue: Cultivar 3

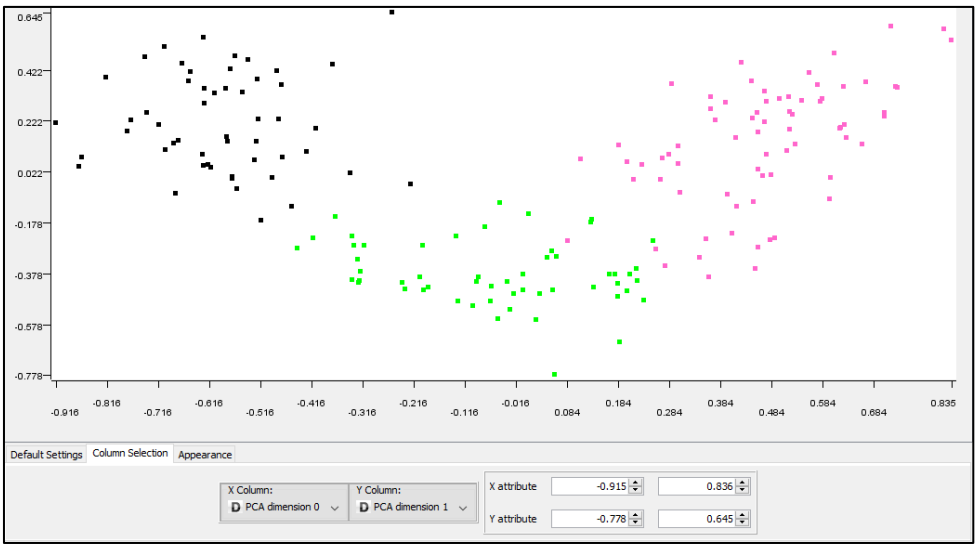
## Task 1.2 – Clustering with normalisation:

For task 1.2, it was required to basically repeat the steps done in task 1.1 with the difference that in this case, the data should be normalized. Normalization is a process that reduces the difference between attributes with highly discrepant values. It is especially useful for preventing that attributes using a smaller scale dominate the calculation of the dissimilarity.
The same plots generated for task 1.1 were generated for task 1.2 and are presented below.




Plot 1.2.1 – 2D Plot of Data PCA with normalisation
Subtitle:
Red: Cultivar 1
Green: Cultivar 2
Blue: Cultivar 3

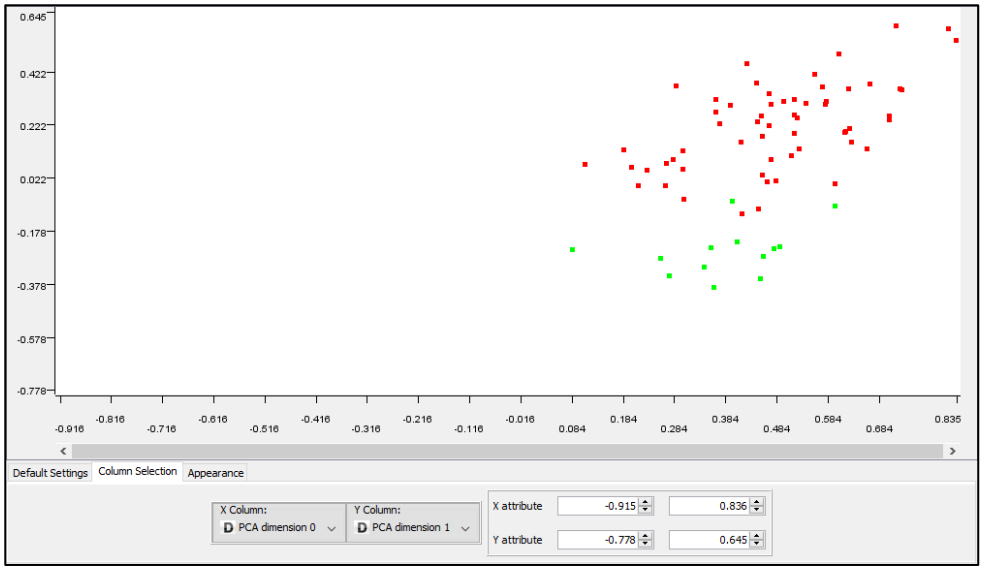
As we can already see, the normalization made the plot look clearer, making the points from different cultivars become more separate than before.
The better distinction among the points helps the clustering algorithm, which now generates more homogeneous clusters, as it is noticeable in the next plots.



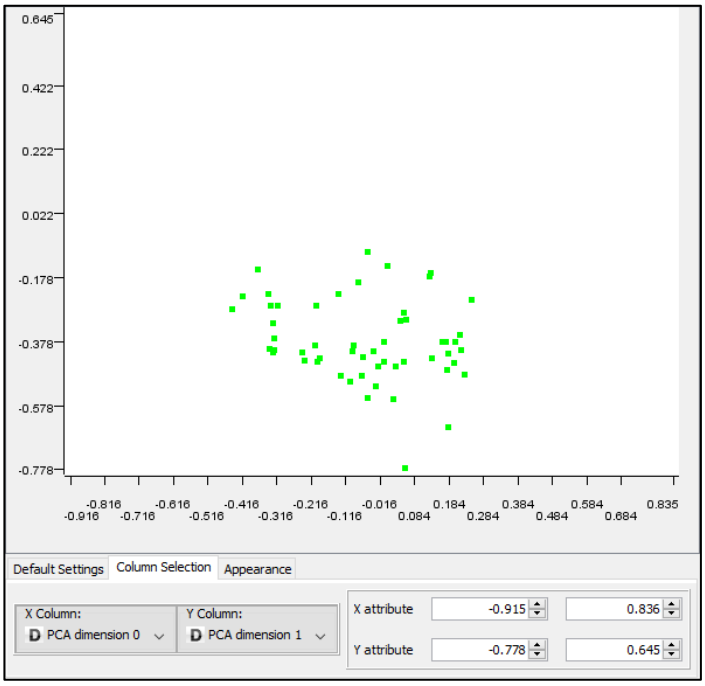
Plot 1.2.2 – 2D Plot of Data Clustering with normalisation
Subtitle:
Pink: Cluster 1
Green: Cluster 2
Black: Cluster 3

The same algorithm used for generating the clustering showed in Plot 1.1.2 was also used for generating this clustering. As the only difference between both is the presence of normalization, it is possible to observe the impact of pre-processing the data before applying a data analysis algorithm to it.

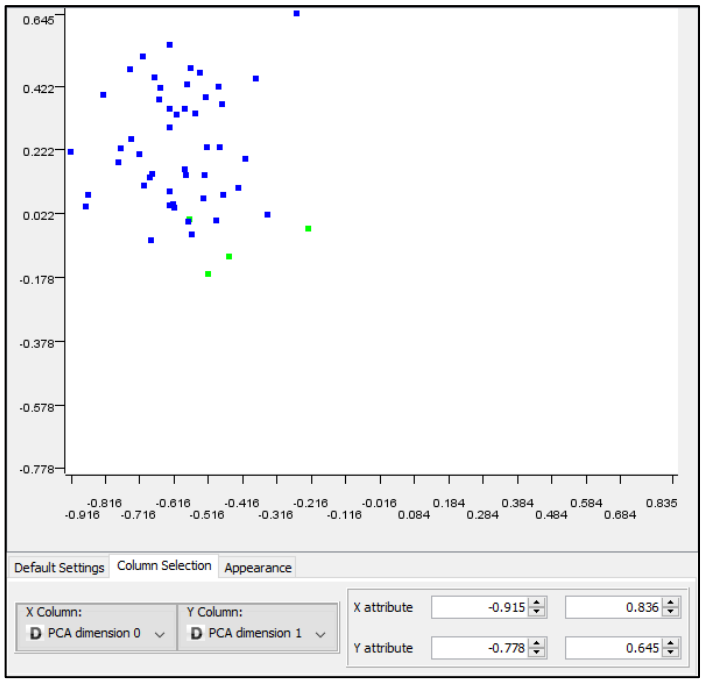
The next plots show the distribution of the classes among the obtained clusters:




Plot 1.2.3a – 2D Plot of Cluster 1 with normalisation
Subtitle:
Red: Cultivar 1
Green: Cultivar 2




Plot 1.2.3b – 2D Plot of Cluster 2 with normalisation
Subtitle:
Green: Cultivar 2




Plot 1.2.3c – 2D Plot of Cluster 3 with normalisation
Subtitle:
Green: Cultivar 2
Blue: Cultivar 3

## Task 1– Evaluation:

In order to objectively measure and compare the quality of the two different clustering results, the entropy of every cluster in both cases were calculated and the results are shown below:

**Clustering statistics**

Data Statistics

| Statistics | Value |
|---|---|
| Number of clusters found: | 3 |
| Number of objects in clusters: | 178 |
| Number of reference clusters: | 3 |
| Total number of patterns: | 178 |

Data Statistics

| Score | Value |
|---|---|
| Entropy: | 0.8834 |
| Quality: | 0.4426 |

| Row ID | Size | Entropy | Normali... | Quality |
|---|---|---|---|---|
| Row17 | 50 | 0.242 | 0.153 | ? |
| Row72 | 68 | 0.834 | 0.526 | ? |
| Row135 | 60 | 1.474 | 0.93 | ? |
| Overall | 178 | 0.883 | 0.557 | 0.443 |

Clustering 1 statistics

**Clustering statistics**

Data Statistics

| Statistics | Value |
|---|---|
| Number of clusters found: | 3 |
| Number of objects in clusters: | 178 |
| Number of reference clusters: | 3 |
| Total number of patterns: | 178 |

Data Statistics

| Score | Value |
|---|---|
| Entropy: | 0.3899 |
| Quality: | 0.754 |

| Row ID | Size | Entropy | Normali... | Quality |
|---|---|---|---|---|
| Row106 | 54 | 0 | 0 | ? |
| Row148 | 52 | 0.391 | 0.247 | ? |
| Row35 | 72 | 0.681 | 0.43 | ? |
| Overall | 178 | 0.39 | 0.246 | 0.754 |

Clustering 2 statistics

The Entropy of a clustering is an external measure of its quality, indicating how well it succeeds in capturing data points of a specific class within each cluster. The higher the class purity within a cluster, the lower is its entropy. In order to calculate the entropy value, the following formula is applied:

$$\text{Entropy} = \sum_i p_i \log(p_i)$$

Where $p_i$, the probability of a member of a cluster *j* belongs to the class i, is computed as follows:

$$p_i = \frac{m_{ij}}{m_j}$$

Where $m_{ij}$ is the number of elements of class $i$ in cluster $j$ and $m_j$ is the total number of elements of cluster $j$.

By analysing the entropy values of both clusterings, it is possible to state that the second one was more successful in distinguishing the elements of each class, indicating it has a better quality.

- **Task 2 – Comparison of Classification Models:**

In this task it is required that two classification models be implemented in order to determine the class of previously unknown data points. The data set used as input for these models is the same as the used in Task 1, an analysis of chemical components in wines grown in three different cultivars.
The basic difference between Task 1 and Task 2 is that the former is a *Descriptive Task*, while the latter is a *Predictive Task*. A descriptive task is aimed to provide a way to describe the data, enabling data scientists to have a deeper understanding of the kind of information they are dealing with. A predictive task, however, has a different objective: determine an unknown information about a data point based on a model generated by the analysis of previous records. When the unknown information is a class, the predictive method used to determine it is said to be a classification algorithm.
The objective of this section is to implement and compare two different classification algorithms, analysing how they work and evaluating the generated results.

## Classification Algorithm #1 – Decision Trees

Decision tree algorithms generate a flowchart like structure where each internal node denotes a test on a data attribute, each branch corresponds to an outcome of that test, and the external (leaf) node denotes a class prediction. At each node, the algorithm chooses the "best" attribute to partition the data into individual classes.

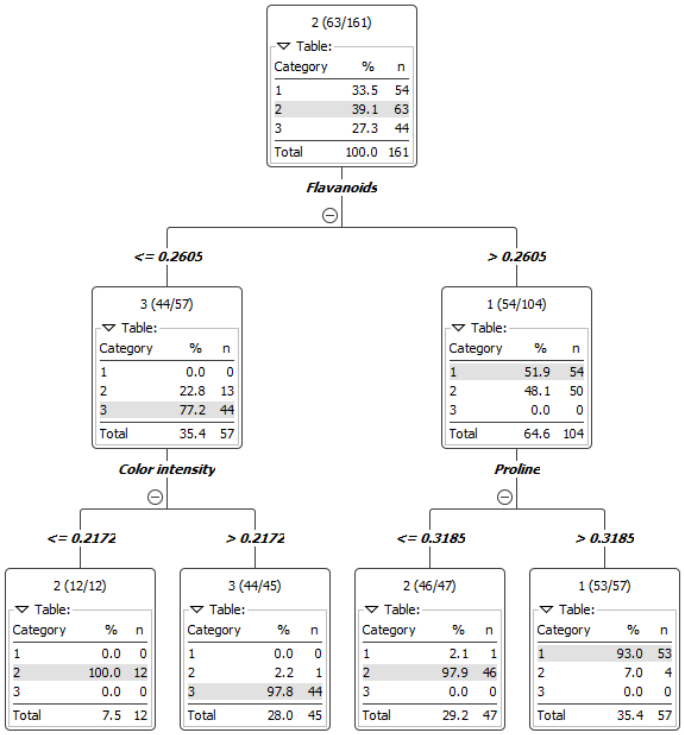
In order to determine what partitions are interesting in the sense of separating data points from a class to the others, decision trees consider the purity of its nodes. The purity is a measure that allows one to objectively evaluate the class distribution of the data in the nodes. The highest the purity, more elements of a specific class are present in the node compared to the elements of the other classes.

In the implemented algorithm the chosen way to determine which partition should be done is the Gain Ratio it provides compared to other possible partitions. The Gain Ratio is calculated considering the entropy of each class, using the following formula:

$$GAIN_{split} = Entropy(p) - \sum_{i=1}^{k} \frac{n_i}{n} Entropy(i)$$

Where $p$ is the parent node, $n_i$ is the number of records in partition $i$ and $n$ is the total number of records in the data set. The entropy calculation is as shown in "Task 1- Evaluation" section.

The generated tree for the given data set is shown below:




Obs.: The algorithm was run using normalized data.

## Classification Algorithm #2 – Naïve Bayesian Method

Bayesian Classifiers are based on statistics. Basically, a Bayesian classification algorithm assumes that a record has a higher chance to belong to a certain class if there are many other records in that class which are sufficiently alike the new record to be classified. The Naïve Bayesian algorithm assumes that each attribute in a data point is independent and although this assumption is not always true, it helps the computation, making it simpler and faster.

Bayesian classifiers work by computing the probability of a record to belong to every possible class, and the classification is done by selecting the class of which this probability is the highest. The probability calculation is done by using the Bayesian theorem, stated below:

$$P(C_i|X) = \frac{P(X|C_i)P(C_i)}{P(X)}$$

This equation means that the probability of a record $X$ belongs to the class $C_i$ is proportional to the probability of finding an object sufficiently alike $X$ in the class $C_i$ and to the size of $C_i$ itself.

It is known that $P(X|C_i)$ can be determined by using the following equation:

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$

Where $x_k$ is the value of $A_k$, the k-esim attribute of $X$.

For continuous valued attributes, $P(x_k|C_i)$ can be calculated by using the formula presented below:

$$P(x_k|C_i) = \frac{1}{\sqrt{2\pi}\sigma_{C_i}} e^{-\frac{(x_k-\mu_{C_i})^2}{2{\sigma_{C_i}}^2}}$$

Where $\sigma_{C_i}$ is the mean value of the attribute $A_k$ within the class $C_i$ and $\mu_{C_i}$ is the standard deviation of the values of $A_k$ within $C_i$.

## Task 2 – Evaluation:

After implementing the algorithms, the coursework specification requires that the obtained results be evaluated and compared. By using 10-fold cross validation method, this comparison is possible, and the performance of both in classifying the data points is shown below:

| Row ID | Error in % | Size of Test Set | Error Count |
|---|---|---|---|
| fold 0 | 0 | 18 | 0 |
| fold 1 | 11.111 | 18 | 2 |
| fold 2 | 5.556 | 18 | 1 |
| fold 3 | 22.222 | 18 | 4 |
| fold 4 | 5.556 | 18 | 1 |
| fold 5 | 16.667 | 18 | 3 |
| fold 6 | 22.222 | 18 | 4 |
| fold 7 | 11.111 | 18 | 2 |
| fold 8 | 5.882 | 17 | 1 |
| fold 9 | 0 | 17 | 0 |

Decision Tree Classifier Error Rates
Average error rate ≈ 10.03%

| Row ID | Error in % | Size of Test Set | Error Count |
|---|---|---|---|
| fold 0 | 5.556 | 18 | 1 |
| fold 1 | 0 | 18 | 0 |
| fold 2 | 0 | 18 | 0 |
| fold 3 | 0 | 18 | 0 |
| fold 4 | 0 | 18 | 0 |
| fold 5 | 0 | 18 | 0 |
| fold 6 | 16.667 | 18 | 3 |
| fold 7 | 5.556 | 18 | 1 |
| fold 8 | 0 | 17 | 0 |
| fold 9 | 5.882 | 17 | 1 |

Bayesian Naïve Classifier Error Rates
Average error rate ≈ 3.36%

For this specific case, the Bayesian predictor had a much better performance than the Decision Tree, but accordingly to the literature, for the general case they usually have approximately the same accuracy.

The 10-fold cross validating method consists in splitting the available data in 10 different partitions and using nine of them as a training set and one as the test set. The process continues

iteratively switching the partition used as the test set. The image displayed above shows the error rates in every partition used as well as the size of each one.

- ## Task 3 – The Search for God Particle: a Binary Classification Task

This task requires that a binary classification model be implemented in order to determine whether or not a collision produced by the Large Hadron Collider (LHC) results in a Higgs Boson, the so called God particle. Typically, only 0.0000003% of the total number of produced collisions generates this particle, which makes that the data collected from the LHC be highly imbalanced.
Imbalanced data sets represent a challenge to data mining algorithms, mainly because a static predictor can easily produce highly accurate predictions just by classifying every data point as belonging to the majority class. Considering the Higgs boson problem, if a predictor says that none of the collisions result on this particle, it would be right in about 99.9999997% of the cases. Although this number seems impressively good, such a predictor is completely useless for the scientists who are actually looking for the Higgs boson, once it will never show them the results they want to find. On the other hand, if a predictor says that a collision will result in the God particle, it is very likely that it is wrong, given that the great majority of the collisions will not.
Considering this scenario, it is possible to state that the challenge for this task is to generate a useful highly accurate classification model that eventually will classify some data points as potential Higgs boson generators. Although doing such a thing seems difficult, fortunately the Data Mining literature provides some efficient approaches for dealing with unevenly distributed data. In this coursework, some of these techniques are going to be demonstrated and the obtained results are going to be evaluated.

### Classification Algorithm #3:

When dealing with unbalanced data sets, one of the most important steps to take is try to balance the available data. Some simple ways to do it include increasing the amount of data points belonging to the minority class or reducing the amount of points belonging to the majority class.
In the algorithm implemented for completing this tasks it was chosen to increase the number of data points representing collisions that result in the Higgs boson in the data set used for learning. The generation of new tuples belonging to the rare class is done by using an algorithm named SMOTE (Synthetic Minority Over-sampling Technique), detailed below.

The SMOTE algorithm:

The main goal of this technique is to generate more data points belonging to the minority class. To do so, the algorithm creates synthetic rows by extrapolating between a real object of the minority class and one of its nearest neighbours of the same class. It picks a point along the line between these objects and then is able to determine the values of the new record's attributes.
In this coursework, SMOTE was used to generate a new data set in which 50% of the records are classified as resultants in Higgs boson.
In this scenario, SMOTE was used to pre-process the data-set before a data-mining algorithm is applied to it, improving its accuracy. It is a good example of how data pre-processing can significantly contribute to achieving a better solution in data-mining problems.

Classification model:

In order to accurately predict the class of the rows, a model named Random Forest Predictor was used. It works by computing a variety of different decision trees and using the prediction of each one as a vote for the final classification. Basically, it assumes that the prediction made by the majority of the trees has a low probability of being wrong. The idea behind of the generation of each tree is the same as the one applied in the algorithm described in Task 1. However, there are some differences between them, the main one being the fact that Random Forest Predictors use random attribute selection when deciding what is the best split for each node. This is the reason why the trees within the forest are different from each other.
One of the ways to build a Random Forest model of $k$ (a user defined value) trees is to sample with reposition a training set $D_i$ $(i = 1,2,3 \dots k)$ of $d$ tuples from the original training set $D$. By doing this, each $D_i$ is a bootstrap sample of $D$, so that some tuples may occur more than once in $D_i$ and others may be excluded. Each $D_i$ is then assigned to a different tree.
In order to decide what attributes should be used to determine the splits, the algorithm randomly select, at each node, a number $F$ of attributes as candidates for the split, $F$ being much smaller than the total number of available attributes.
The best amongst the $F$attributes is then chosen based on the information gain ratio it provides to the tree.

## Task 3– Evaluation:

The results achieved by this algorithm were very satisfactory considering the training set provided, reaching nearly 95% accuracy, as shown in the picture below:

| Row ID | Error in % | Size of Test Set | Error Count |
|---|---|---|---|
| fold 0 | 4.111 | 3600 | 148 |
| fold 1 | 5.083 | 3600 | 183 |
| fold 2 | 4.556 | 3600 | 164 |
| fold 3 | 4.444 | 3600 | 160 |
| fold 4 | 4.5 | 3600 | 162 |
| fold 5 | 4.444 | 3600 | 160 |
| fold 6 | 5.083 | 3600 | 183 |
| fold 7 | 4.278 | 3600 | 154 |
| fold 8 | 4.835 | 3599 | 174 |
| fold 9 | 4.112 | 3599 | 148 |

Random Forest Predictor Error Rates
Average error rate ≈ 4.54%
(10-fold cross validation method used)

Although the obtained results are satisfactory, the Random Tree Classification algorithm has some let-downs. Due to the fact that it has to compute multiple decision trees, its execution time may be a problem. In this coursework, due to the limitations of the computer used, much of the training data-set had to be cut in order to make the algorithm viable. A stratified sampling was applied on the data set, resulting on only 20% of the original records being used.
This fact along with the insertion of synthetic records in the SMOTE algorithm may negatively impact the accuracy on the final test set.

Other factors that should be taken in consideration when analysing ways to improve this algorithm is the fact that the provided training set include 7 attributes that are not present on the test-set. These 7 attributes are high level attributes, which means they are dependent on the other 21 low-level attributes. Because of this fact, the 7 extra attributes were considered redundant information when developing the algorithm and therefore were completely ignored. However, the best thing to do was to ignore the 21 low-level attributes and use only the high-level attributes, avoiding the so called “curse of dimensionality”. The problem with this approach is that as the test set does not contain these high-level attributes, it would be impossible to apply the learned model from the training set to the actual test set.
A way of partially solving this problem is to calculate the 7 attributes of the records from the test set and then apply the learned model. However, doing these calculations is not a simple task, since it would require 7 polynomial interpolations in 7 dimensions each. Unfortunately, the KNIME platform used for this coursework does not provide resources to do this kind of

calculations and as there was not enough time to hand-code these procedures, other solutions had to be used.

## 3. Conclusion:

The development of this coursework occurred relatively well, no major difficulties were found and it was possible to gain highly valuable knowledge in the Data Mining field. However, as stated on the evaluation of Task 3, this piece of coursework has plenty of room for improvement, given that some algorithms have not been implemented due to lack of available time.

In general, it is possible to say that the development of this coursework was challenging and satisfactory, significantly contributing to our formation as computer scientists.

## 4. References:


- *Han, Jiawei - Kamber, Micheline - Pei, Jian (2011) – Data Mining: Concepts and Techniques, 3rd edition*. "Foreword" (page xix), "Decision Tree Induction" (page 105), "Random Forests" (pages 382 & 383).

- KNIME Documentation. Accessed on March 17, 2016. Available at: https://tech.knime.org/documentation.

- Chawla, Nitesh. *Data Mining For Imbalanced Datasets: An Overview.* Accessed on March 17, 2016, available at https://www3.nd.edu/~dial/papers/SPRINGER05.pdf


## 5. Appendix

The algorithms used in the development of this coursework were implemented using the KNIME platform and all the flowcharts created are included in a zip file provided along with this report.

The zip file is organized in directory structure containing 3 folders, one for each Data Mining Task. The folder "Task1" has two subfolders, one for each sub-task (Clustering with / without normalisation).

Each KNIME flowchart was exported in a zip format and in order to see the charts it is necessary to import the zip files to KNIME as a flowchart.