Contents

Introduction

The substantial amount of data used and collected throughout the world on a daily basis, has led to extensive research being carried out on data analysis, in the hope of elucidating the complexity of valuable information. Data mining focuses on solving problems by performing such analysis and "is defined as the process of discovering meaningful patterns in data that lead to some advantage".[1] It is a method in which raw data is transformed into useful information, inconsistent data is removed, relations and patterns between variables are extracted and data visualization is achieved. This report aims to investigate a dataset from a fictional chain of stores and recommend a successful predictive model, which estimates the annual revenue of each individual store. The following task requires inspection of the variables contained within the data set in order to determine those that satisfy an average error of less than half a million pounds and thus, make a valid prediction. Data mining is capable of processing vast quantities of data, enabling computers to gain the necessary knowledge to perform tasks impossible to program. The report outlines a complete data mining project and draws conclusions regarding the best model for the given task. This is achieved through the detailed explanation of data preparation, otherwise known as pre-processing, and a further analysis of two major data mining techniques: decision trees and neural networks. Comparison of the trained models and their outcomes, allows for recommendations to be made and thus providing an explanation as to why certain stores are financially more successful than others. As requested by the client, the software package used for building the predictors, is Weka. Furthermore, an investigation of the provided data set is carried out to evaluate the significance of specific variables, with regard to future data mining procedures.

Data Preparation

As Ian H. Witten asserts "a problem that plagues practical data mining is poor quality of the data. Errors in large databases are extremely common. Attribute values, and class values too, are frequently unreliable and corrupted."[2] While the process of data preparation can be exhaustive and time-consuming, it is vital for successful data mining and therefore, producing a correct training set. Real world data is generally inconsistent, containing incongruities in values otherwise known as noise. Noise in data takes the form of outliers, minority values and typographic data entry errors. This fictional task, as any real-world project, contains both numerical and nominal attributes within its dataset. This causes a mixed-attribute problem, that can make the data preparation process more difficult.

Inspecting the data's distribution can expose hidden errors in a dataset and the easiest way to investigate a frequency distribution, is through plotting a histogram. Histograms are commonly used for data visualization and serve as a means to identify errors in a data set, during the pre-processing phase of a data mining project. Such visual representations of the given data insure that outliers, which indicate potential errors within the data are made explicitly clear. **Figures 1.1 - 1.7** show how histograms are utilised to expose outliers, minority values as well as flat and wide variables, in order to deduce meaningful information. Furthermore, data attributes in large datasets can contain values which are incorrectly missing; missing values pose an obvious threat. Nonetheless, the given dataset does not

[1] Witten, Ian. H. & Frank, Eibe. '*Data Mining Practical Machine Leaning Tools and Techniques'*. San Francisco, Elsevier, 2005, p.5

[2] Ibid: p. 312

contain any missing values, thus the data cleansing process is solely focused on removing or editing instances, containing outliers and minority values.

Data entry errors lead to incorrect values. With regard to nominal attributes, a misspelled input can create an extra possible value. By observing **Figure 1.1** it is evident that the Car Park attribute follows two patterns regarding the presence of parking spaces. While the majority of the values map the Yes/No pattern, seven instances follow the Y/N scheme. This is the product of human error during the data collection process. Removing the instances containing the latter input scheme is not necessary, since the values can be modified to fit the Yes/No pattern.

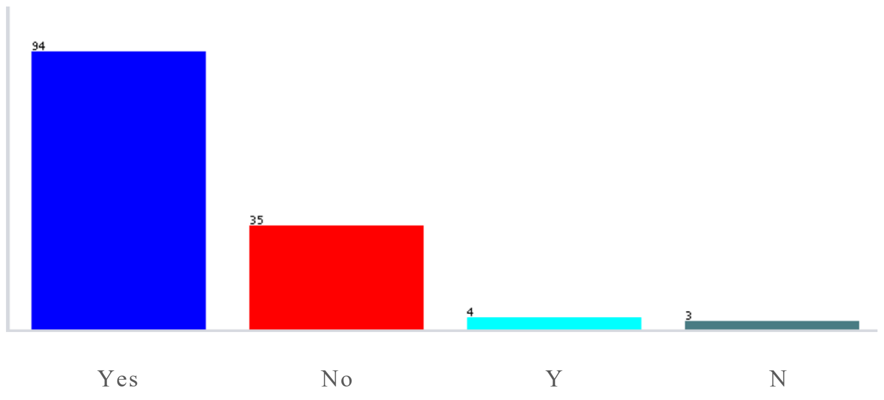


Figure 1.1 - Car Park Attribute Histogram: Y axis is the total number of instances

As already mentioned, minority values pose a major threat to the pre-processing of data. These values appear infrequently in the data and, should it not be possible to collect further similar data, their removal is necessary. **Figure 1.2**, presents evidence of such minority values. The Location attribute contains a Village value which is only encountered once and therefore is not representative of the dataset.

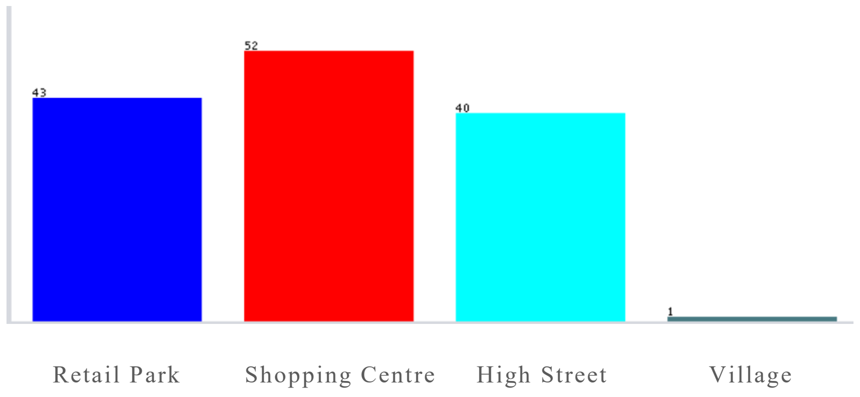


Figure 1.2 – Location Attribute Histogram: Y axis is the total number of instances

Furthermore, another occurrence of minority values is spotted on the *Country* attribute of the dataset. Prior to attribute selection, all variables are considered to carry equal weight and therefore, data cleaning should be applied to the particular attribute, regardless of its logical significance. **Figure 1.3** indicates that two instances hold a *France* value in the *Country* attribute and consequently should be removed from the dataset.

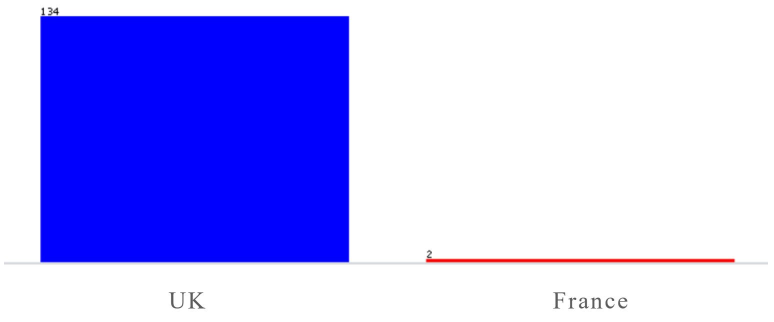


Figure 1.3 – Country Attribute Histogram: Y axis is the total number of instances

In addition, another problem encountered during the data preparation process, is the existence of flat and wide variables. These are attributes, where all their values are minority values and as such they should be excluded from the model, since they obstruct the general pattern search. It is evident by **Figure 1.4**, that such is the case for the Town variable, where each data point is completely different to the rest.

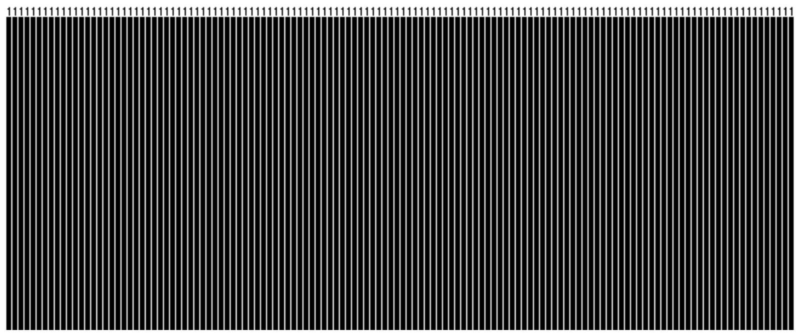


Figure 1.4 – Town Attribute flat and wide distribution: X axis indicates a new town value and Y axis indicates the times each value occurs.

Similarly, the same observation can be made for the Manager name attribute. In this case however, some of the values seem to overlap. This may be explained as pure coincidence given that different managers can have the same name.

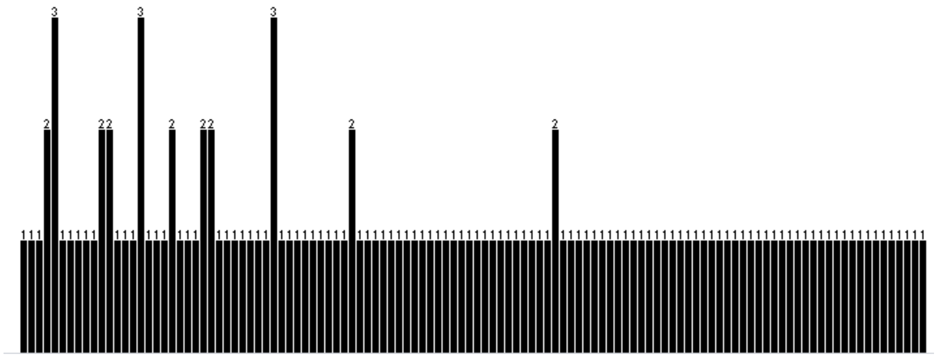


Figure 1.5 – Manager name flat and wide distribution: X axis indicates a new name value and Y axis indicates the times each value occurs.

The last occurrence of flat and wide variables, concerns the Store ID attribute. Since Store ID is a numeric variable, the values are represented in bins and thus it is visually harder to spot the flat and wide distribution, as shown in **Figure 1.6**. Nonetheless, each id is unique, hence all three aforementioned attributes should be removed from the data set.

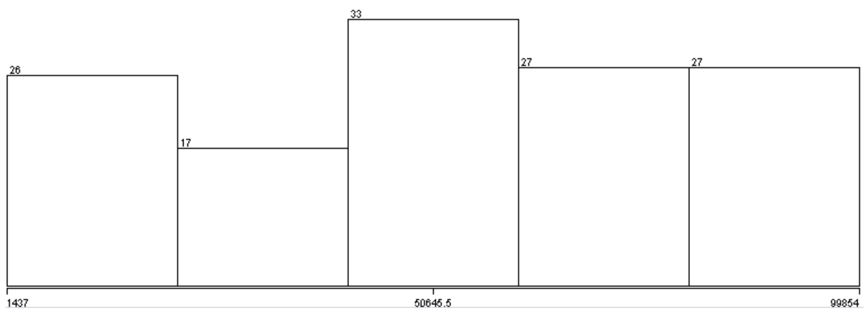


Figure 1.6 – Store ID flat and wide distribution: X axis represents the bin spectrum and Y axis represents the number of instances contained in a bin

"Typographic or measurement errors in numeric values generally cause outliers that can be detected by graphing one variable at a time." [3] Values which appear to substantially deviate from the existing pattern are considered to be erroneous. **Figure 1.7** visualises the distribution of the Staff attribute and exposes inconsistent values within the dataset. These outliers consist of values which significantly deviate from the distribution set. In more detail, two instances hold the values 300 and 600 for their Staff attribute, which are much larger than all the others. In a similar manner, another instance of the dataset holds a value of -2 for the aforementioned attribute, causing a logical inaccuracy. As such, these instances should be removed to further reduce the data noise.

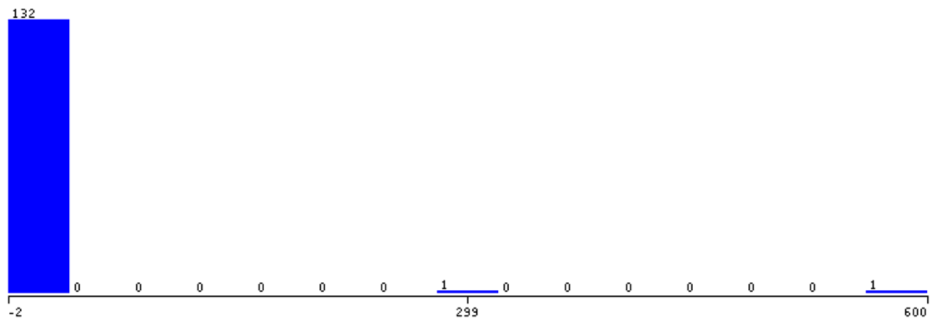


Figure 1.7 – Staff Attribute Histogram: Y axis is the total number of instances

The completion of the data preparation process, sets the foundations for the training set. For the purposes of this report, a thorough technical description of two main data mining techniques is provided below, before advancing to variable investigation, attribute selection and the model training.

[3] Witten, Ian. H. & Frank, Eibe. '*Data Mining Practical Machine Leaning Tools and Techniques'*. San Francisco, Elsevier, 2005, p.59

Techniques Description

As already mentioned, one of the goals of this report is to analyse in depth the decision tree and multilayer perceptron data mining techniques. For each of the techniques, the algorithms used are discussed in order to fully explain how they learn, how they represent the solution and how they make predictions. The mathematical formulas behind each concept are further analysed to provide an insightful evaluation of each technique.

Decision Tree – J48

First of all, before explaining decision trees and how they work, it is essential to introduce the concept of classification. The process of classification can be perceived as the assignment of an object to a certain class, based on its similarity to previous examples of other elements, in a specific data set. Decision trees, among other broadly used classification methods "employ a learning algorithm to identify a model that fits the relationship between the attribute set and the class label of the input data." [4]

The J48 decision tree used for this task, is an extension of ID3 and the implementation of the C4.5 algorithm. The algorithm is originally based on a divide and conquer approach and works better on nominal attributes, rather than numerical ones.

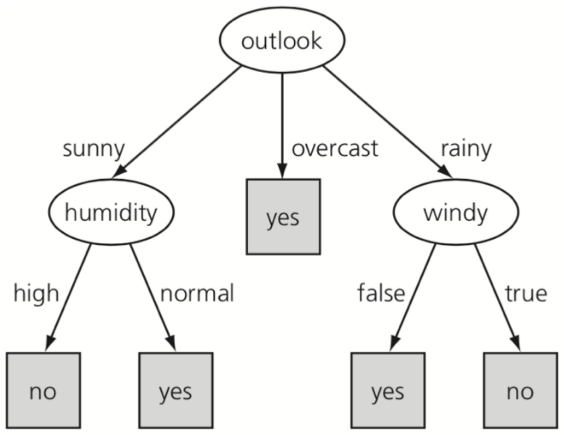


Figure 2.1 – Decision Tree for weather data (Data Mining Practical Machine Learning Tools and Techniques – Chapter 4.3 p.101)

Figure 2.1 points to the existence of three different nodes within a decision tree: a single *root* node, *internal* nodes and *leaf* or *terminal* nodes. Starting from the root node, branches are created where variable test conditions are performed to separate data with different characteristics. This process is recursively repeated until a lead node is reached, in which case, no more branches can be created and thus the object is classified. While the divide and conquer approach is simple in its execution, a number of problems arise. Undeniably, the biggest question is how to determine which attribute to split on.

[4] Tan, Pang-Ning. Steinbach, Michael. Kumar, Vipin. '*Introduction to Data Mining (First Edition)'*, Boston MA, USA, Addison-Wesley Longman Publishing Co. Inc., 2005, p. 148

The ID3 algorithm extends the divide and conquer approach, using information theory to solve this issue. This is achieved, by splitting on the variable that has the greatest measure of purity, known as *information*. Information can be thought of as, a measure of uncertainty and is based on the probability of something happening. The mathematical formula to calculate the information is: $I(e) = -\log_2(p_e)$, where p_e is the probability of an event *e* occurring. The weighted average information of all the possible values of an attribute is called *information value* or *entropy* and is evaluated as the total amount of the probability of each possible event, multiplied by its *information value.* Entropy is calculated using the following formula, where H(X) is considered to be a measure of impurity and uncertainty in variable X.

$$H(X) = \sum P\,(x_i)I(x_i) = -\sum P\,(x_i)\,log_2(P(x_i))$$

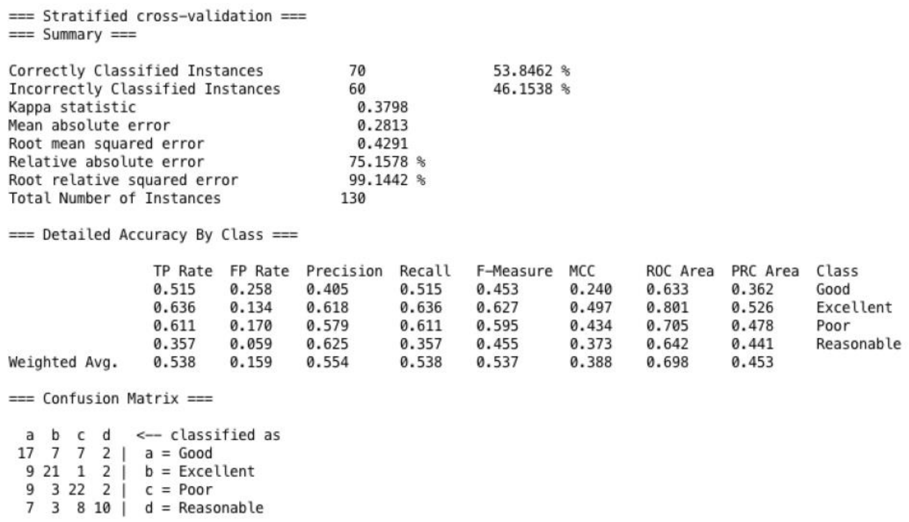
The more even the distribution of X becomes, the higher the entropy gets. In order to quantify the amount of information needed to evaluate the outcome of a variable X, given a known input, the conditional entropy needs to be calculated. The conditional entropy formula $H(Outcome\ |Known\ Input)$ is essentially used to calculate entropy after performing a variable split. Furthermore, comparing the entropy computed before and after the split, is used to measure the *Information Gain of Input*. "The information gain is based on the decrease in entropy after a dataset is split on an attribute. Constructing a decision tree is all about the finding attribute that returns the highest information gain." [5] Therefore, the ID3 algorithm picks the root node by calculating the information gain of the output class for each input variable, choosing the one that removes the most uncertainty. Additionally, branches are created for each value the chosen attribute can take, and the same information gain calculations are recursively repeated for every internal node until no more branching is required. When the entropy of a branch reaches 0, a leaf node has been reached and thus, no further splitting is necessary. The algorithm also stops when all data has been classified.

$$Information\ Gain = H(Outcome) - H(Outcome\ |Input)$$

In addition to the technique description, it is important to highlight the manner in which the particular decision tree algorithm represents a solution. The "evaluation of the performance of a classification model is based on the counts of tests records correctly and incorrectly predicted by the model. These counts are tabulated in a table known as a *confusion matrix*."[6] **Figure 2.3** depicts the confusion matrix for a J48 decision tree model on the task's pre-processed dataset with attribute selection being carried out, using stratified ten-fold cross-validation as the testing option.

[5] Dr Sayad, Saed. *Decision Tree – Classification,* http://www.saedsayad.com/decision_tree.htm

[6] Tan, Pang-Ning. Steinbach, Michael. Kumar, Vipin. '*Introduction to Data Mining (First Edition)'*, Boston MA, USA, Addison-Wesley Longman Publishing Co. Inc., 2005, p. 149



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          70               53.8462 %
Incorrectly Classified Instances        60               46.1538 %
Kappa statistic                          0.3798
Mean absolute error                      0.2813
Root mean squared error                  0.4291
Relative absolute error                 75.1578 %
Root relative squared error             99.1442 %
Total Number of Instances              130

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.515    0.258    0.405      0.515    0.453      0.240    0.633     0.362     Good
                 0.636    0.134    0.618      0.636    0.627      0.497    0.801     0.526     Excellent
                 0.611    0.170    0.579      0.611    0.595      0.434    0.705     0.478     Poor
                 0.357    0.059    0.625      0.357    0.455      0.373    0.642     0.441     Reasonable
Weighted Avg.    0.538    0.159    0.554      0.538    0.537      0.388    0.698     0.453

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 17  7  7  2 |  a = Good
  9 21  1  2 |  b = Excellent
  9  3 22  2 |  c = Poor
  7  3  8 10 |  d = Reasonable
```

Figure 2.2 – Result's summary of a J48 decision tree model on Weka.

Evidently, running the J48 decision tree algorithm in Weka, also provides a summary of the results. Among the displayed information is the number of both correctly and incorrectly classified instances, followed by several error measurements and lastly accuracy measures, including the *ROC area*. A *Receiver Operating Characteristic* or *ROC* curve is used to identify the number of false positives and true positives for each possible classification.
A perfect test consists of a ROC area with a value of 1, whereas values below 0.5 indicate an ineffective test. The following point system provides guidance on classifying the accuracy of such a test:

0.90-1 = Excellent test
0.80-0.90 Good test
0.70-0.80 = Fair test
0.60-0.70 = Poor test
0.50-0.60 = Failed test

Further analysis of the confusion matrix, accuracy and error measurements, as well as the attribute selection methodology and cross-validation, is carried out on the **Model Training** section of the report.

Multilayer Perceptron – Neural Network

While decision trees are widely used for the classification of data, the multilayer perceptron technique is suitable for prediction. Whilst a classification could be perceived as a prediction of classes, classification values are discreet, whereas the values of a prediction tend to be continuous. Nonetheless, the essence of a prediction model lies within the concepts of predictors and predicted values. Much like decision tree techniques, the multilayer perceptron is heavily based on various mathematical models, which require to be thoroughly explained, in order to fully understand the concept of neural networks.

The multilayer perceptron or otherwise known as MLP, is a particular type of neural network, that has the ability to learn a function between the inputs and the outcome of a certain dataset. It operates by creating a function out of multiple smaller ones, joined together by weighted connections.

While, “standard perceptrons calculate a discontinuous function, due to technical reasons neurons in MLPs calculate a smoothed variant of this” [7], evident in **Figure 2.3**.

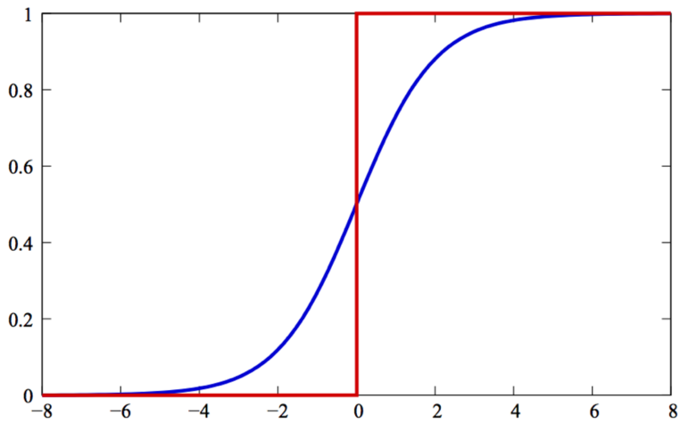


Figure 2.3 – Logistic sigmoid and step discontinuous activation functions (Machine Learning: Multi Layer Perceptrons Prof. Dr. Martin Riedmiller, Albert-Ludwigs-University Freiburg, p. 4/61)

This logistic activation function, $f(a) = \frac{1}{1+e^{-a}}$, is the most fundamental function for any input, where α is the weighted sum of the inputs. The input data of a neural network is used to modify the weighted connections between all of its functions, until it is able to predict the data accurately. This is referred to as *training* of the neural network.

Before analysing the training process of a multilayer perceptron, it is vital to understand the basic structure of the MLP. First of all, this type of neural network is a finite acyclic graph, which means that all neurons can be organized in layers. Each node is a neuron with logistic activation. Nodes that proceed all other connections are called *input neurons*. On the contrary, “nodes that are no source of any connection are called *output neurons*.” [8] It is possible to have more than one output neuron for any given MLP. For the purposes of this task however, there is only one output (*Profit*). The rest of the nodes, that are neither part of the input layer, nor the output layer, are called *hidden neurons*. **Figure 2.4** demonstrates the basic architecture of the MLP structure that has been described, containing the three aforementioned layers.

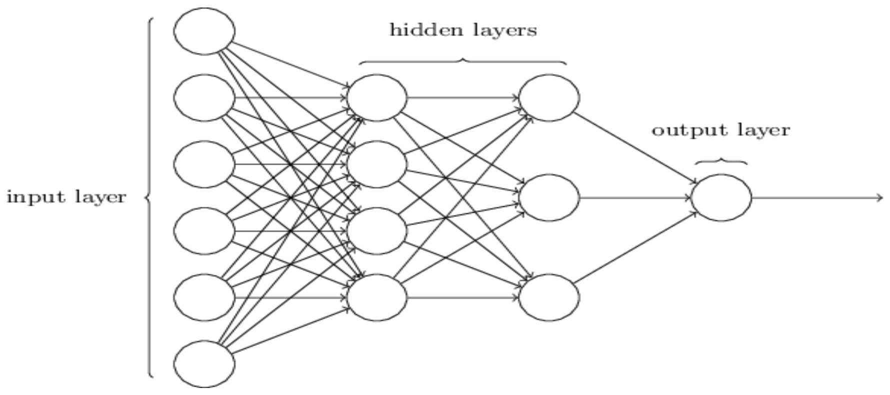


Figure 2.4 - An Example of Multilayer Perceptron Architecture. Image taken from: https://dzone.com/articles/deep-learning-via-multilayer-perceptron-classifier

[7] Prof. Dr. Riedmiller, Martin. *‘Machine Learning: Multi Layer Perceptrons’*. Lecture Slides, http://ml.informatik.uni-freiburg.de/former/_media/documents/teaching/ss09/ml/mlps.pdf, p. 4/61

[8] Ibid: p. 6/61

As previously mentioned, predicting the data accurately is part of the training process of the neural network. Once data is prepared, it is split into two different sets: the test set and the training set. The MLP reads though the data file, presenting the predictors as inputs and the predicted values as the target output. Therefore, making a prediction involves comparing the value given by the neural network, to the target value. Learning is the process which aims to reduce the error measurements between the networks actual outputs and the desired outputs. This suggests a propagation through the network and thus, algorithm learning is otherwise known as back-propagation. The concept of back-propagation is to minimize the test error, in relation to the connection weights. Initially, the multilayer perceptron starts with random weights. Then the weighted inputs are propagated forward through the network and the *output signal* of each neuron from the hidden layer is calculated:

$$O_j = f(\sum_i w_{ij} \ O_i)$$

where *j* is the neuron from the hidden layer and *f* is the sigmoid activation function. Once the output signal y_n is calculated for each node *n*, this process is repeated for the output layer neuron using the output y_n from the hidden layer nodes as inputs. At this point, the real output *y* is compared to the desired target output value *z*, found in the training data set. The difference is called error signal *δ* of the output layer neuron: $\delta = z - y$. The error signal of the output neuron is then propagated back to all the nodes, and the error signal for each individual neuron is computed, $\delta_i = \sum w_{ij} \, \delta_j$, resulting to the adjustment and update of weights in the network. The mathematical formula used to calculate the modified weight is:

$$w'_{ij} = w_{ij} + \eta \delta_\iota \frac{df_{i(e)}}{de} y_i$$

where *i* is the current node, *j* is the hidden layer node it points to, *η* is the learning rate, *df(e)/de* is the derivative of the sigmoid activation function *f* and *y* is the output signal. Then a new iteration of forward propagation begins, and the process is repeated until the error no longer reduces.

The manner in which the multilayer perceptron technique represents a solution, differs in the representation present in the J48 decision tree algorithm. While both techniques, contain almost the same error measurements, neural networks concern the prediction of data. A numerical measure closely tied to the MLP is the correlation coefficient. The correlation coefficient, measures the linear relationship between two variables and in this case, it concerns the relationship between the predicted value and the target value. The accepted range of values for this measure, stands from -1 to 1. A value of 0 indicates no relation between variables, whilst a value of -1 or 1 illustrates perfect negative or perfect positive correlation respectively, as **Figure 2.5** suggests. Nonetheless, negative values of the correlation coefficient are not present in the given task.

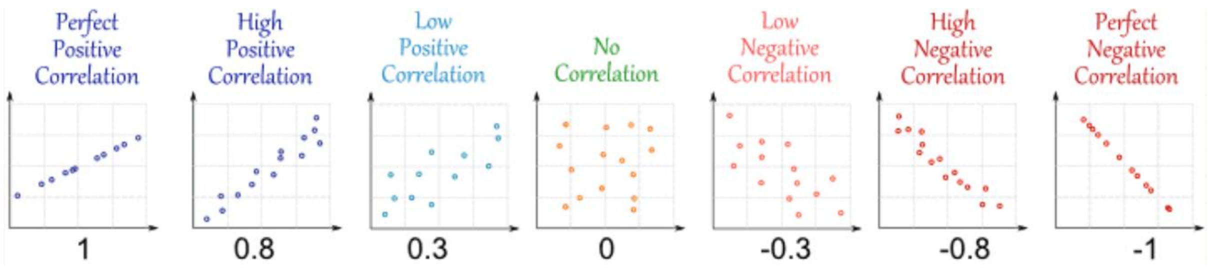


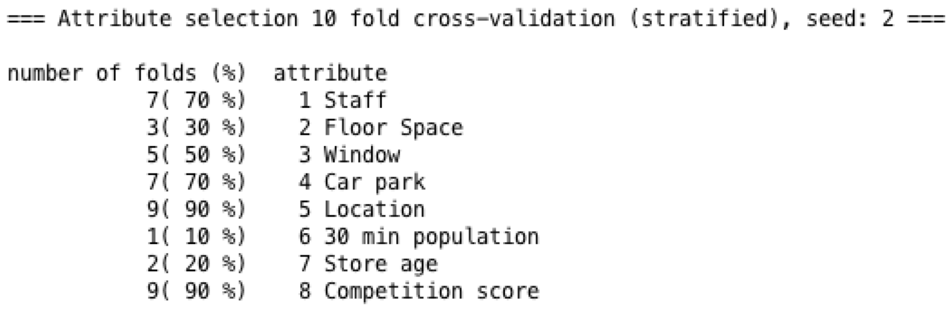
Figure 2.5 – Illustrated correlation value range

Variable Selection

Now that both of the desired techniques have been thoroughly explained, appropriate attribute selection is required for successful model training. Weka provides different methods for attribute selection. The following variable selection, utilizes two of these algorithms for the decision tree and MLP technique respectively. As required by the client, the population related attributes are among the investigated variables, in order to clarify if they are sufficient for the task. Further, attribute selection relies on proper data pre-processing, hence, as previously explained, the *Town*, *Store ID* and *Manager name* variables should be considered already removed from the initial data set.

Decision Tree Attribute Selection

First of all, in order to provide valid results, the *Profit* attribute is immediately removed from the dataset, since the decision tree J48 algorithm can only classify nominal attributes. At the Select attributes section of Weka, the *WrapperSubsetEval* attribute evaluator is used. The default parameters need to be modified and therefore, the J48 algorithm is selected and the threshold is set to -1, which means that the mean standard deviation repeat limit is set automatically. The search method used is *BestFirst* and its direction parameter is also changed to *Backwards*, meaning that the starting set contains all the attributes, which are then removed one by one respectively. Using the full training set, this results to the eight following attributes, excluding the *Performance* attribute: **Staff, Floor Space, Window, Car park, Location, 30 min population, Store age, Competition score**. It is immediately clear that three of the population attributes are not needed. Next, the remaining attributes from the data set are removed and the same attribute selection methods are used, with ten-fold cross validation. Performing cross-validation using different random seeds, elucidates that the 30 min population attribute among others, is encountered the least amount of times in the folds, as shown by **Figure 3.1**. Removing the attributes that hold the smallest percentage, leaves the data set with the following variables: **Staff, Car Park, Location** and **Competition score**. Carrying out further cross validation tests, concludes that this is the optimum minimum attribute set, to be used by the J48 decision tree algorithm.



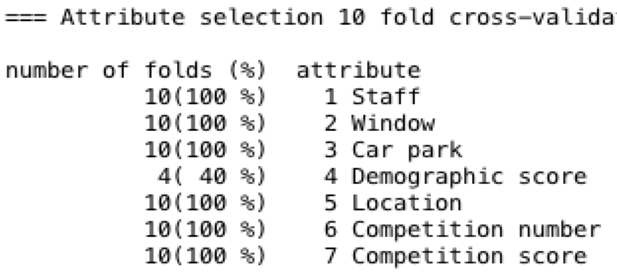
```
=== Attribute selection 10 fold cross-validation (stratified), seed: 2 ===

number of folds (%)  attribute
           7( 70 %)    1 Staff
           3( 30 %)    2 Floor Space
           5( 50 %)    3 Window
           7( 70 %)    4 Car park
           9( 90 %)    5 Location
           1( 10 %)    6 30 min population
           2( 20 %)    7 Store age
           9( 90 %)    8 Competition score
```

Figure 3.1 – 10-fold cross validation attribute selection results for the J48 algorithm.

Multilayer Perceptron Attribute Selection

In the case of the MLP, since the goal of the task is to estimate the annual profit, the *Performance* attribute is no longer needed and thus removed from the initial data set. This time, the chosen attribute evaluator is the *CfsSubsetEval* in its default settings. The searching method and its parameters will be identical to the previous attribute selection process. Using the full training set, results to the following seven variables, excluding the *Profit* attribute: **Staff, Window, Car park, Demographic score, Location, Competition number** and

Competition score. As before, the rest of the attributes are removed and multiple 10-fold cross-validation tests using different seeds are performed, to check for inconsistencies. Summarizing the results of such tests, all attributes but one are necessary, as **Figure 3.2** clearly proves. Hence, it is established that the best variable subset for the MLP, excluding the *Profit,* is the following: **Staff, Window, Car park, Location, Competition number** and **Competition score**.



```
=== Attribute selection 10 fold cross-validation seed: 3 ===

number of folds (%)  attribute
          10(100 %)    1 Staff
          10(100 %)    2 Window
          10(100 %)    3 Car park
           4( 40 %)    4 Demographic score
          10(100 %)    5 Location
          10(100 %)    6 Competition number
          10(100 %)    7 Competition score
```

Figure 3.2 - 10-fold cross validation attribute selection results for the MLP algorithm.

The verdict obtained by performing both attribute selection methods, is sufficient to draw the conclusion that most of the attributes provided are actually not needed for prediction nor classification. Evidently, among the unnecessary variables, is the population data which the client requested particular focus on.

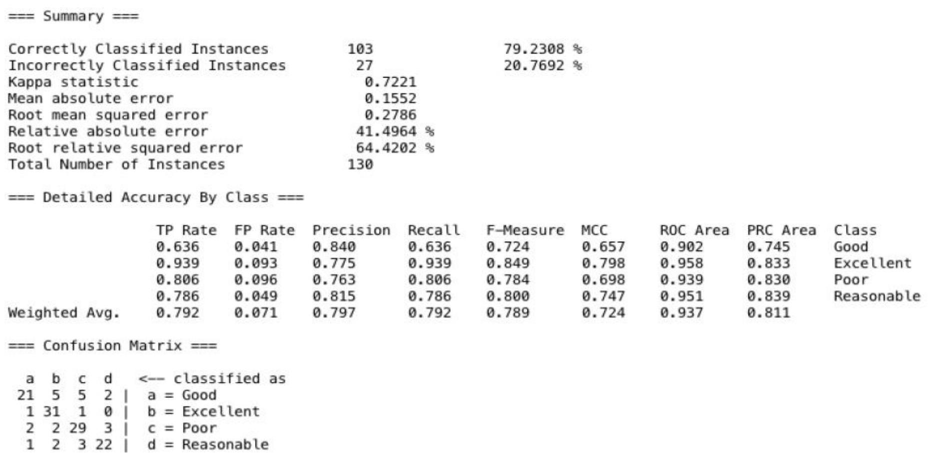
Model Training

Having completed the selection of attributes, the training of models using the two data mining techniques can proceed. The goal of model training, is not just to provide results for a given data set, but centres around applying each technique to the given task and adjusting the algorithms in order to perform better. Analysing the exact error measurements that Weka provides, is necessary to access the technique results and make recommendations.

Decision Tree Model Training

Initial data set

To begin with, training the model on the initial data set will provide a first glimpse on how well the J48 algorithm can perform, when given the full dataset as input. For the first attempt of the model training procedure, the J48 classifier with its default settings is chosen and the test option used is the training set. The result of this experiment, as **Figure 4.1** suggests, seems to be oddly positive. The percentage of the correctly classified instances is at 79.2% and the weighted average of the ROC area almost reaches 1. However, evaluating on the training set causes a phenomenon known as overfitting, which produces misleading optimistic performance results. Overfitting occurs when a model learns the noise in the training set to such extent that the performance of new data is massively affected. Therefore, performance on the training set does not guarantee equally good performance on an independent test set. To avoid this phenomenon, the data set must be split into two parts, one for training and one for testing. To get reliable evaluation results the training data needs to be different to the testing data. In most cases, this is achieved by using 1/3 of the data for testing and the rest for training. The process of using certain amount of data for training and testing is known as *holdout*.

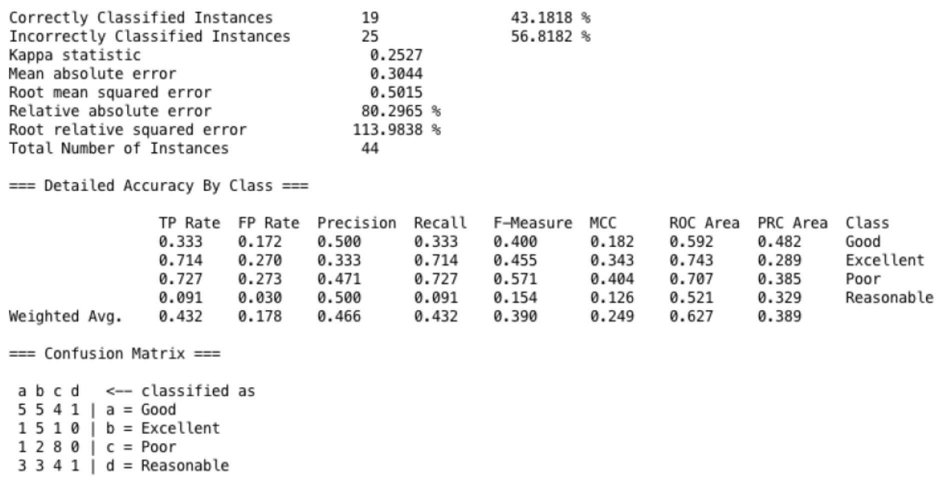


```
=== Summary ===

Correctly Classified Instances         103               79.2308 %
Incorrectly Classified Instances        27               20.7692 %
Kappa statistic                          0.7221
Mean absolute error                      0.1552
Root mean squared error                  0.2786
Relative absolute error                 41.4964 %
Root relative squared error             64.4202 %
Total Number of Instances              130

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.636    0.041    0.840      0.636    0.724      0.657    0.902     0.745     Good
                 0.939    0.093    0.775      0.939    0.849      0.798    0.958     0.833     Excellent
                 0.806    0.096    0.763      0.806    0.784      0.698    0.939     0.830     Poor
                 0.786    0.049    0.815      0.786    0.800      0.747    0.951     0.839     Reasonable
Weighted Avg.    0.792    0.071    0.797      0.792    0.789      0.724    0.937     0.811

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 21  5  5  2 |  a = Good
  1 31  1  0 |  b = Excellent
  2  2 29  3 |  c = Poor
  1  2  3 22 |  d = Reasonable
```

Figure 4.1 – J48 algorithm result summary tested on the training set, without parameter adjustment

First subset of variables

The second attempt on training the model will use the eight attributes, that were initially selected from the first *WrapperSubsetEval* algorithm. However, in order to avoid overfitting, the testing option used, is the percentage split set at the default 66%, meaning that 2/3 of the data are used for training. Also, the J48 parameter *confidenceFactor* which represents the confidence threshold for pruning, is set to 0.6. Running this model, provides significantly worse results than the initial experiment. Within this attempt, the percentage of correctly classified instances is reduced to 43.2%. It should be clarified that the total number of instances in this scenario is 44 since only 1/3 of the data set is actually used for testing. In this case the sample used for training or testing may not be representative. This is not only evident by the increase of the *mean squared error* and the rest of the error measurements, but also by the dramatic decrease of the kappa statistic to 0.25. The kappa statistic, only given when the class value is nominal, is a "chance-corrected measure of agreement between the classifications and the true classes." [9] When the kappa value, nears zero, the possibility of the result occurring by chance is more likely. Moreover, correctly classifying *Reasonable* performance, has utterly failed as made clear by the low ROC area of the specific class. A closer look to the confusion matrix, displayed in **Figure 4.2** proves that only one out of the eleven instances have been correctly classified as *Reasonable*.



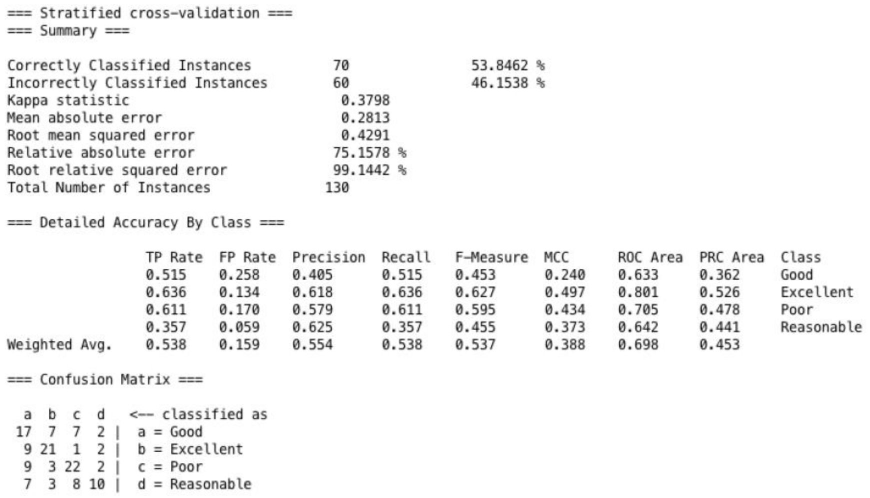
```
Correctly Classified Instances          19               43.1818 %
Incorrectly Classified Instances        25               56.8182 %
Kappa statistic                          0.2527
Mean absolute error                      0.3044
Root mean squared error                  0.5015
Relative absolute error                 80.2965 %
Root relative squared error            113.9838 %
Total Number of Instances               44

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.333    0.172    0.500      0.333    0.400      0.182    0.592     0.482     Good
                 0.714    0.270    0.333      0.714    0.455      0.343    0.743     0.289     Excellent
                 0.727    0.273    0.471      0.727    0.571      0.404    0.707     0.385     Poor
                 0.091    0.030    0.500      0.091    0.154      0.126    0.521     0.329     Reasonable
Weighted Avg.    0.432    0.178    0.466      0.432    0.390      0.249    0.627     0.389

=== Confusion Matrix ===

 a b c d   <-- classified as
 5 5 4 1 | a = Good
 1 5 1 0 | b = Excellent
 1 2 8 0 | c = Poor
 3 3 4 1 | d = Reasonable
```

Figure 4.2 - J48 algorithm result summary tested using percentage split, with parameter adjustment.

[9] Montana Tech, '*Weka Error Measurements*', CSCI347, Fall 2017, https://katie.mtech.edu/classes/csci347/Resources/Weka_error_measurements.pdf

Second subset of variables

The final result of the attribute selection process has shown that four attributes can be used to for the best classification of data, using the J48 decision tree algorithm. While the holdout procedure of error rate estimation, performed on the previous subset of variables overcomes the problems imposed by overfitting, it is not enough to produce valid results. The standard method employed in order to predict the error rate of a learning algorithm, is known as stratified ten-fold cross-validation. This process involves the division of a single fixed sample of data, into ten random parts. Therefore, "the class is represented in approximately the same proportions as in the full dataset."[10] This method reduces the risk of getting a lucky test set by using nine of the ten subsets for training and the tenth one for testing. Considering this, the J48 classifier is used again, changing the confidenceFactor parameter to 0.75 and choosing 10-fold cross-validation as the test option. It is immediately obvious that, the results are undoubtedly better. As **Figure 4.3** suggests, 70 out of the 130 instances are correctly classified. The kappa statistic has increased, and the mean squared error is relatively low. In comparison with the previous results, it is evident that this model training is the most reliable and accurate of any other. The result summary is examined on the **Results** section, in which a detailed analysis of some key accuracy measures is provided.



```
=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances          70               53.8462 %
Incorrectly Classified Instances        60               46.1538 %
Kappa statistic                          0.3798
Mean absolute error                      0.2813
Root mean squared error                  0.4291
Relative absolute error                 75.1578 %
Root relative squared error             99.1442 %
Total Number of Instances              130

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.515    0.258    0.405      0.515    0.453      0.240    0.633     0.362     Good
                 0.636    0.134    0.618      0.636    0.627      0.497    0.801     0.526     Excellent
                 0.611    0.170    0.579      0.611    0.595      0.434    0.705     0.478     Poor
                 0.357    0.059    0.625      0.357    0.455      0.373    0.642     0.441     Reasonable
Weighted Avg.    0.538    0.159    0.554      0.538    0.537      0.388    0.698     0.453

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 17  7  7  2 |  a = Good
  9 21  1  2 |  b = Excellent
  9  3 22  2 |  c = Poor
  7  3  8 10 |  d = Reasonable
```

Figure 4.3 - J48 algorithm result summary tested using 10-fold cross-validation, with parameter adjustment

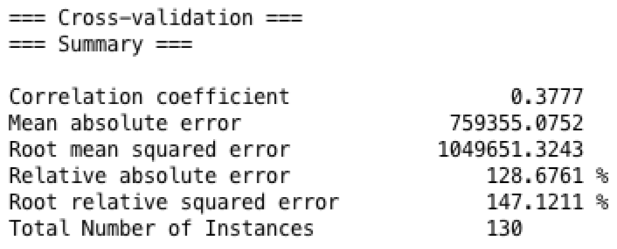
MLP Model Training

Initial data set

Following the same procedure as the decision tree model training, the MLP is chosen as the classifier and the initial data set is used, using the *Profit* attribute as the desired value to be predicted. Similarly, the default parameters are not altered at this point. However, unlike the previous model training, 10-fold cross validation is going to be the preferred testing option, to get accurate results. That is because, overfitting can occur in either of the data mining techniques. **Figure 4.4** displays the error measurements summary of the constructed model. Evidently the correlation coefficient is low at 0.37 and the mean absolute error is considerably high. The MAE is "the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal

[10] Witten, Ian. H. & Frank, Eibe. '*Data Mining Practical Machine Leaning Tools and Techniques'*. San Francisco, Elsevier, 2005, p.150

weight."[11] Simply put, the absolute error shows how much the results deviate from the real values. Another error measure present in the summary, is the Root mean squared error or MSE. A further analysis of the MSE is provided in the **Results** section. Moreover, both measures are negatively-oriented scores, meaning that lower values are better. It is clear that in this model, the MSE is significantly large, with a value of more than one million. The client has stated that the predicting model would only be of use, if the prediction error is less than half a million pounds. The process of learning in a neural network, is also highly related to minimizing the MSE.



```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.3777
Mean absolute error                 759355.0752
Root mean squared error            1049651.3243
Relative absolute error                128.6761 %
Root relative squared error            147.1211 %
Total Number of Instances              130
```

Figure 4.4 - 10-fold cross validation result for the MLP algorithm. All attributes used with no parameter adjustment

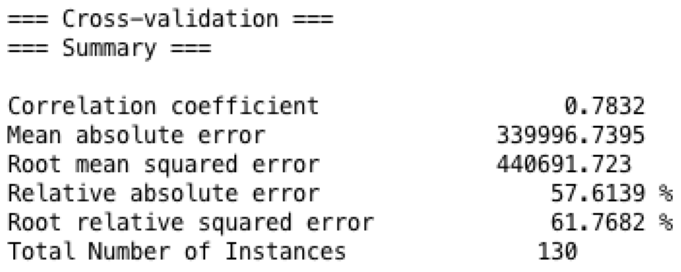
First subset of variables

Since the first attempt of the model training was unsuccessful, the six variables from the attribute selection are utilized to provide better results. In order to build a more successful model, some of the MLP parameters need to be altered. "A decay parameter causes the learning rate to decrease with time: it divides the starting value by the epoch number to obtain the current rate."[12] Setting the decay value to *True*, can therefore improve the performance of the neural network and prevent it from diverging. In addition, the *hiddenLayers* parameter defines the hidden layers present and how many nodes each one contains. After multiple testing, it is established that the optimum value of nodes is 14, in one hidden layer. Further, apart from configuring the structure of the MLP, the *learningRate* and *momentum* parameters can be adjusted, to boost the performance of the neural network. The *learningRate* is essentially a small value which is multiplied by the derivate of the loss function with respect to every weight, each time the weights are updated during the backpropagation process. Its value is set to 0.2. On the other hand, momentum concerns the addition of a small proportion of the update value from the previous iteration, to the new weight. According to Ian H. Witten, "this smooths the search process by making changes in direction less abrupt."[13] The value of this parameter is set to 0.1. **Figure 4.5** contains the summary of this particular model and the results are significantly better than the previous attempt. The correlation coefficient is proportionally higher, and both the MSE and MAE values are satisfactorily smaller than the half a million pounds threshold.

[11] JJ. 'MAE and RMSE – Which Metric is Better?', *Human in a Machine World, Medium.* March 23, 2016, https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

[12] Witten, Ian. H. & Frank, Eibe. '*Data Mining Practical Machine Leaning Tools and Techniques'*. San Francisco, Elsevier, 2005, p. 413

[13] Witten, Ian. H. & Frank, Eibe. '*Data Mining Practical Machine Leaning Tools and Techniques'*. San Francisco, Elsevier, 2005, p. 233

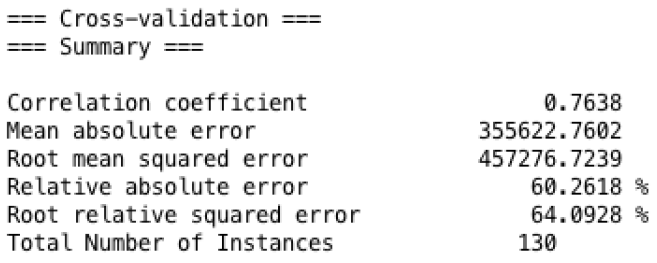


```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.7832
Mean absolute error                 339996.7395
Root mean squared error             440691.723
Relative absolute error                 57.6139 %
Root relative squared error             61.7682 %
Total Number of Instances              130
```

Figure 4.5 - 10-fold cross validation for the MLP algorithm, including parameter adjustment and using six variables.

Second subset of variables

In order to perform a well-rounded model training, another subset of variables is tested using the exact same methodology as the pervious experiment. However, to satisfy the needs of the given task, one of the attributes is removed to check how well the model performs. It has been established by the decision tree attribute selection, that the *Competition number* variable is not as necessary as the *Competition score*. Therefore, during this model training the former attribute is removed and the total number of variables used, is reduced down to five. Most parameters remain the same as before, although the number of epochs is changed. An *epoch* is a single complete pass through the entire training set. The *trainingTime* parameter sets the number of epochs to train through the neural network. Setting its value to 70 produces a result that is only marginally less efficient than the previous one. A further analysis of the results is provided in the next section of the report.



```
=== Cross-validation ===
=== Summary ===

Correlation coefficient                  0.7638
Mean absolute error                 355622.7602
Root mean squared error             457276.7239
Relative absolute error                 60.2618 %
Root relative squared error             64.0928 %
Total Number of Instances              130
```

Figure 4.5 - 10-fold cross validation results for alternative attribute subset of the MLP algorithm, including parameter adjustment.

Results

This section focuses on analysing the results of the two best models, that each data mining technique produced. The second subset of variables for the J48 and MLP algorithms are chosen respectively, as the best model after taking into consideration both performance and minimum number of attributes. Lastly, a comparison of the two models is provided in order to establish which model proves to be more accurate.

Decision tree best model

As shown by **Figure 4.3** the percentage of correctly classified instances for the specific model is higher than 53%, meaning that more than half of the instances in the dataset are successfully classified. Another accuracy measure however, that can provide a more in-depth indication of the model’s performance is known as *ROC curve*. The ROC curve is a plot of TP (true positive) rates against FP (false positives) rates. A model that performs better than chance will lie above the diagonal, known as chance line where TP rates = FP rates. As a general rule, the best systems tend to approach the top left corner of the diagram as much as possible. As far as the task is concerned, the importance of the ROC curves lies within the *Poor* and *Excellent* performance classifier. The closer the ROC area for these classes is to 1, the more reliable the results can be, as stated previously in the technical description of decision trees. The cost of classifying a *Poor* performance store as *Excellent* and vice versa can be significant to the client, thus such probabilities need to be minimized. **Figure 5.1** visualises the ROC curves for the *Excellent* and *Poor* classifications, including the chance diagonal.

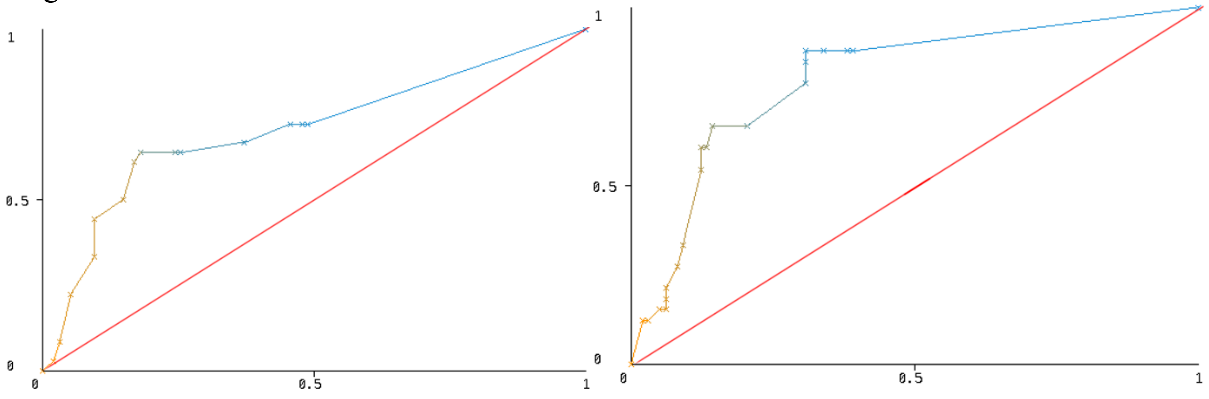


Figure 5.1 – Left chart: Poor performance ROC curve. Right chart: Excellent performance ROC curve. X axis depicts false positives, Y axis depicts true positives. Red line: Chance diagonal

Multilayer perceptron best model

On this model, the first to notice is the correlation coefficient. **Figure 4.6** indicates high positive correlation, which is most certainly desired. However, the most significant statistics displayed in the model's summary, are the MAE (mean absolute error) and the RMSE (root mean squared error). The latter measure of accuracy represents the standard deviation of the differences between the target values and the predicted values. The lower both measures are, the more accurate the prediction. In this case, the minimum target for either measure is the 0.5 million threshold, which evidently both measures effectively satisfy. Moreover, visualizing the classifier errors, is essential in evaluating the prediction error of the model. **Figure 5.2** presents the distribution of error.

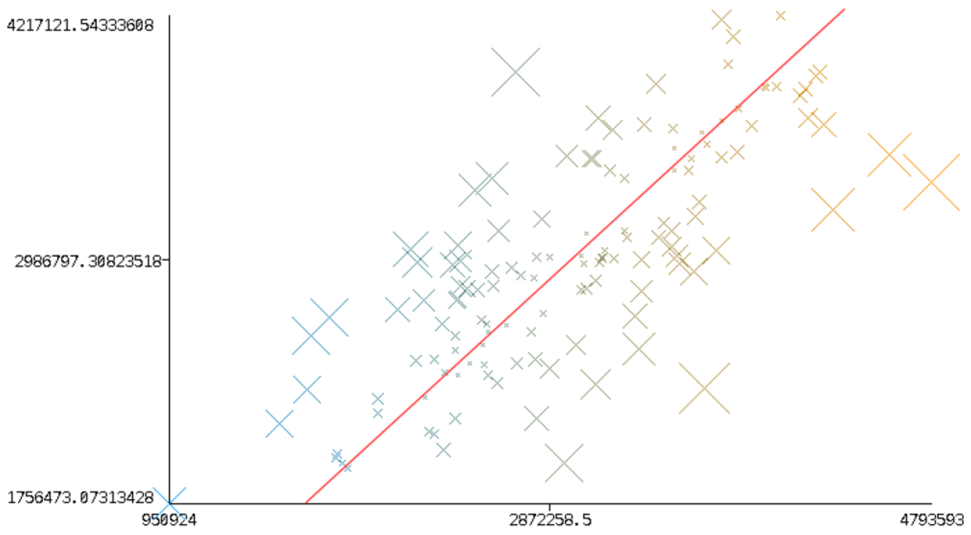


Figure 5.2 – Classification error distribution plot. X axis depicts actual profit. Y axis depicts predicted profit

The red line through the data minimizes the distance between all the points and the line. This distance is the error of the model, known as Mean Squared Error:

$$MSE = \frac{1}{n}\sum_{e}^{n}(\bar{Y} - Y)^2$$

Model comparison

For accurate comparison of the two models, only the same error measurements can be compared to establish which model performs best. When the class value is nominal the kappa statistic is evaluated, whereas when the class value is numeric the correlation coefficient is provided. Even though the accuracy measures are not present in both models, they have the same range of values. For either the correlation coefficient or the kappa statistic, the closer their value is to 1, the more accurate the prediction. In the case of the neural network, the correlation coefficient reaches high positive correlation, with a value of 0.76. However, the J48 decision tree algorithm produces a kappa statistic of 0.38, meaning that the agreement between the classification and the true classes is poor. Furthermore, the relative absolute error and the root relative squared error are examined, since both measures of errors are evaluated by the two models. The RAE of the MLP model is almost 15% lower than the one produced by the J48 decision tree model. Similarly, the RRSE of the latter model is about 35% higher than the MLP's. Lastly, the aim of the given task is to predict a numerical value rather than classifying a nominal attribute, so it is clear that the MLP is the best performing model and simultaneously the most suitable for the given task.

Recommendations

As explicitly stated from the client, the collection of variables is a rather expensive task and therefore it advised to use as few attributes as possible for the predictor models. Although, only one technique is ultimately recommended, a variable suggestion for both techniques is provided.

Variable recommendations

Regarding the decision tree technique, the following four attributes are sufficient for the given task and provide the best possible results: **Staff**, **Car park**, **Location**, **Competition score**. This choice is supported by the results of the J48 trained model as well as the attribute selection process of Weka.

As far as the as the multilayer perceptron technique is concerned, after careful consideration of the two aforementioned subset of variables, the following five attributes are recommended: **Staff**, **Window**, **Car park**, **Location**, **Competition score**. As previously highlighted, the MSE and MAE are marginally higher than the first suggested variable subset. However, taking into consideration the cost of collecting each attribute, it is safe to say that this subset is sufficient for the purposes of the given task.

Technique recommendation

The aim of the specific task, is to create a predictor that can estimate the annual profit of a store. Whist both techniques seem to produce fairly good results, the decision tree algorithm can only classify nominal attributes. Neural networks unlike decision tree algorithms, like the J48, are able to handle multiple numerical and categorical variables and can cope with non-linear relationships. It is also evident from the model comparison performed in the **Results** section, that the MLP model has the smallest value for each error measure. The MLP and the majority of neural networks pose difficulties with regards to understanding how predictions are made. While, impactful errors can occur if not trained properly, neural networks can generalize to data that has not been seen before. Training the MLP model, using the proposed six variables is also in scope of the half a million-pound goal, required by the client. In conclusion, it has been clear through the technical description of each technique that decision trees are widely used for classification of data rather than prediction, thus it is apparent the MLP is the most efficient and suitable choice for recommendation.

References

Works cited

Witten, Ian. H. & Frank, Eibe. '*Data Mining Practical Machine Leaning Tools and Techniques*'. San Francisco, Elsevier, 2005

JJ. 'MAE and RMSE – Which Metric is Better?', *Human in a Machine World, Medium.* March 23, 2016, https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

Montana Tech, '*Weka Error Measurements*', CSCI347, Fall 2017, https://katie.mtech.edu/classes/csci347/Resources/Weka_error_measurements.pdf

Prof. Dr. Riedmiller, Martin. *'Machine Learning: Multi Layer Perceptrons'*. Lecture Slides, http://ml.informatik.uni-freiburg.de/former/_media/documents/teaching/ss09/ml/mlps.pdf,

Dr Sayad, Saed. *Decision Tree – Classification,* http://www.saedsayad.com/decision_tree.htm

Tan, Pang-Ning. Steinbach, Michael. Kumar, Vipin. '*Introduction to Data Mining (First Edition)*', Boston MA, USA, Addison-Wesley Longman Publishing Co. Inc., 2005

Works consulted

'Multilayer Perceptrons and Neural Networks', Technical University of Sofia, http://www.wseas.us/e-library/transactions/circuits/2009/29-485.pdf

'Improved J48 Classification Algorithm for the Prediction of Diabetes', International Journal of Computer Applications (0975 – 8887) Volume 98 – No.22, July 2014, http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.678.9273&rep=rep1&type=pdf

Brownlee, Jason. *'Overfitting and Underfitting With Machine Learning Algorithms*, Machine Leaning Mastery, March 21, 2016, https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/

Ratner, Bruce. Journal of Targeting, Measurement and Analysis for Marketing June 2009, Volume 17, Issue 2, pp 139–142, *'The correlation coefficient: Its values range between +1/−1, or do they?*', May 18, 2009, https://link.springer.com/article/10.1057/jt.2009.5