UNIVERSITY of STIRLING

ITNPBD6 – Data Analysis

Course Assignment

David Haveron – Student Number 2527317

David Haveron, MSc. Student
University of Stirling
Stirling FK9 4LA
Scotland UK

Dear Mr Buquetlowd,

Please find attached the report detailing the development of the regression predictive model and the performance classification model, as per your request. Please note explanations have been given where possible, however if any detail of interest has been omitted, please feel free to contact me to discuss further.

I hope this work will be beneficial to your commercial operations and will stimulate growth for the World of Bargains chain going forward.

Kind Regards,

David Haveron
MSc. Student

Contents

Introduction

A consultative service tender was won, where the service requested included the investigation and delivery of two machine learning models which aim to assist in the effective management for a chain of shops across the UK. Each of the machine learning models are to fulfil a specific requirement or function: the requirement of the first model is to ***predict*** the expected commercial income for a given shop instance with specific feature values. The second model is to ***classify*** an existing shop instance into a pre-defined performance category.

These two models aim to assist management achieve the following results:

- *Estimate* the expected (or *predicted*) commercial revenue of existing shop branches for comparison with the true commercial revenue generated by the shop branch. This process can be implemented to evaluate those chain locations which may be under-performing, and corrective action can be taken to address the under-performance. Should a given shop be achieving a higher than expected (or predicted) revenue, an evaluation of that shop branch could be advantageous to discover what shop features contribute to that high performance and if they can be applied to other chain branches.

- *Estimate* the expected commercial revenue of potential new chain branches (for a given instance of 'engineered' feature values) in an aim to identify growth opportunities substantiated by high probability of commercial success, based on historic chain performance.

- Finally, to effectively *classify* both existing shop branches and potential new shop branches into distinct categories of performance – 'Poor', 'Reasonable', 'Good' or 'Excellent'.

Both the *prediction* and *classification* models were designed using a data mining framework as illustrated in *Figure 1*, below.

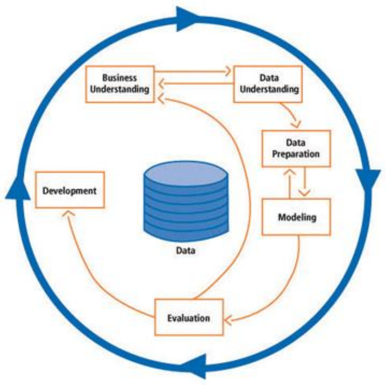


Figure 1 - The Data Mining Process (illustration owner Steven Struhl, cRoss industry standard)

Following the lifecycle of the data mining process above, the objectives of this report are to:

- Introduce the consultancy task and demonstrate understanding of the commercial operation
- Describe/interpret the dataset provided by World of Bargains
- Outline the data preparation process undertaken
- Detail the approach and techniques used in the modelling process
- Report and evaluate model results

In conclusion, a commercial strategy is suggested for further development and implementation of these models to deliver sustainable, profitable growth for the World of Bargains franchise.

Data mining can also be termed ‘knowledge discovery’ (or more recently ‘machine learning’), as it is a process of finding correlations and patterns among many features in a dataset. This discovery of knowledge from datasets can be used to increase revenue and/or reduce costs. The actual knowledge discovery is done by building mathematical models consisting of algorithms, which effectively describe a set of rules which approximate the relationship between these inputs to a specific target output.

As the primary objective of the consultancy services rendered was to develop a computer program which guides management in the choice of new shop locations (that is, to increase commercial revenue) and equips management with assessment tools (through prediction/classification) for the performance of existing branches (reduce costs) - it becomes clear the data mining process is well suited to achieve these objectives.

Data Summary

The store data provided, was received as a CSV file containing a (137x20) matrix of data. For purposes of the report, this matrix has been divided into an 'inputs' feature matrix **X** and 'outputs' matrix **y**.

> *X = [* ***Town*** *(nominal),* ***Country*** *(nom),* ***Store ID*** *(numeric),* ***Manager Name*** *(nom),* ***Staff*** *(num),* ***Floor Space*** *(num),* ***Window*** *(num),* ***Car Park*** *(nom),* ***Demographic Score*** *(num),* ***Location*** *(nom),* ***40min population*** *(num),* ***30min population*** *(num),* ***20min population*** *(num),* ***10min population*** *(num),* ***Store age*** *(num),* ***Clearance Space*** *(num),* ***Competition Score*** *(num)]*

For illustrative purposes a fictitious example instance of a shop from matrix **X** can be seen below.

> $X_{(i)}$ ***= [Stratton, UK, 1234, Frank, 5, 14634, 145, Yes, 14, Retail Park, 1288372, 2837261, 4938271, 6758371, 8, 192, 31]***

The target variables matrix **y** can be seen below, indicating for a given instance of matrix **X**, a predicted profit can be calculated *OR* the shop instance can be classified into a performance class.

y = ***[Profit (num), Performance (nom)]***

Or (illustrative example)

$Y_{(i)}$ ***= [4044051, 'Excellent']***

Many the features labels are self-explanatory, however for thoroughness the definitions for a selection of the feature labels can be seen below.

→ Floor space - measurement of shop floor area (interpretation of units is area in square feet)

→ Window space - measurement of public window area, used for advertising/marketing (interpretation of units is area in square feet)

→ Car Park - 'Yes' or 'No'

→ Demographic Score - a measure of how well a given prospect fits your target audience

→ Location - '*Shopping Centre*, '*High Street*', '*Retail Park' or 'Village'*

→ 40min, 30min, 20min, 10min – the (radial) population size for a given drive time

→ Store age – the number of years the store has been trading

→ Clearance space in store – the floor space in square feet, dedicated to reduced-price products

→ Competition number – how many competing stores are in the proximity to a store branch

→ Competition score – a value representing the competitor 'strength' associated with a given location

Data Preparation

Machine learning models yield best performance when the data they are trained on, is adequately prepared. Unfortunately, data seldom comes in this desirable format and varying degrees of preprocessing is necessary to improve model performance. This can be done 'manually' using software such as Microsoft Excel or using assorted libraries in Python/Java. The data mining software of choice, Weka, implements many of the steps internally to yield improved model performance, faster. This is advantageous as much of the preprocessing or 'heavy lifting' is done by Weka and only a small amount (if any) need to be done manually in advance to improve model performance.

The steps described below aim to walk the reader through the preprocessing of the data for ease of model replication. Steps 1 and 2 were carried out using Microsoft Excel and steps 3 onwards aim to explain some best practise preprocessing steps some of which are internally administered by Weka.

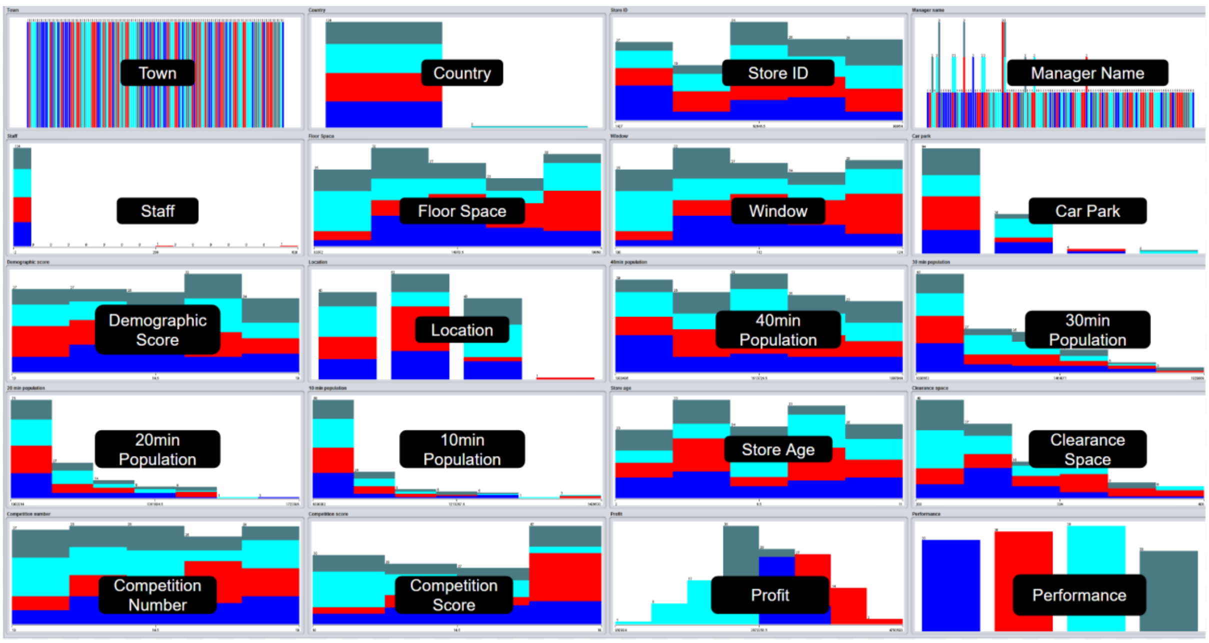


Figure 2 – Pre-processing distribution of variable values in Weka

1. Removing Rows or imputing Missing Values

One element of shaping a raw dataset into a consistent dataset is to identify data anomalies or inconsistencies. There are a few approaches to rectify these inconsistencies. Primarily the client can be approached and the data inconsistency (ies) can be identified, investigated and corrected with the client's approval. An example of a data entry error is a negative sale value where a sale should only take on a value of zero (no sale) or a positive value. Ideally the assorted data entry errors should be identified, discussed with client and appropriately corrected to ensure high quality data. However, this at times becomes infeasible and alternative methods will suffice. The first alternative is to remove the vector/row in which the data error was identified, although at a potential cost to data quality and integrity. The second alternative is to impute values (if possible) as an average value across the vector.

After an exploratory data analysis, the following data inconsistencies/errors were identified and the following action taken to rectify the errors.

- 'Country' Feature – Potential data entry error (instance 41 & 97), where the town listed was confirmed geographically to be in the UK and not in France – these data entry errors were amended to reflect towns in the UK.
- 'Staff' Feature – Potential data entry error (instances 4, 55 & 110), where the Staff counts listed were identified as potential outliers in the dataset. Instance 4 had a value of -2 and was corrected to reflect a

positive staff count of 2, Instance 55 and 110 had a Staff count of 300 and 600 respectively, significantly higher than the staff count at other stores across the country – these instances were removed from the dataset.

- '**Car Park**' Feature – Inconsistent data entry errors where the instances took values in the set { Yes, No , Y , N }. Instances 9, 12, 31, 62, 84, 115 & 121 were amended to reflect a set of consisting of 'Yes' and 'No' values
- '**Location** Feature – Shop instance 37, potential entry outlier for shop in Southwick where the shops location is listed as 'village'. The shop entry was confirmed geographically to be located on a High Street in a small fishing village. The entry has been amended to reflect its location on the High Street iteration

2. Removal of feature vectors

During the exploratory data analysis in Weka, it was noted that four columns contained data, which, have no impact on the performance of the model. These variables are labelled flat or wide variables and add no value to the model performance. More specifically, the '*Town'* feature vector consisted entirely unique entries and the '*Country'* feature vector consisted of identical entries down the vector. The assumptions made by removing these two feature vectors are that the models built, serve only to predict/classify shops trading in the United Kingdom and further that no two shops are to be trading in the same town as they would be in direct commercial conflict with each other.

In addition, the feature vector named '*Manager'* was removed as these variable values will have no impact on the model during learning. This implicitly implies that every shop is managed independently by its dedicated manager. Finally, the feature vector '*Store ID'* was removed, assuming each store would be assigned a unique number and hence would have no impact on the model during learning.

3. Training/Test Split

For an effective measure of model performance, the dataset was subdivided primarily into two sets: a training set and separate test set. This is to ensure the model built not only performs well on the training data but also generalizes well on unseen data.

Furthermore, the training set is loaded into Weka and the K-fold cross-validation method is used to evaluate model performances and optimize the hyperparameters chosen. K-fold cross-validation can be conceptually seen below in *Figure 3*, where the training dataset provided is subdivided into K subsets. In larger training datasets, K is commonly chosen as 10, however as the training dataset is relatively small, K is chosen as 5. The models are trained on the first four subsets and evaluated against the fifth subset, where an approximate error is determined. In second iteration, the models are trained on the first three subsets in addition to the fifth subset, and evaluated against the fourth subset of the data, where the approximate error again is calculated. This process repeats K (5) times until all the subsets have been used as a test reference. The errors for each iteration are summed and divided by the number of iterations to yield an accurate representation of the model performance.

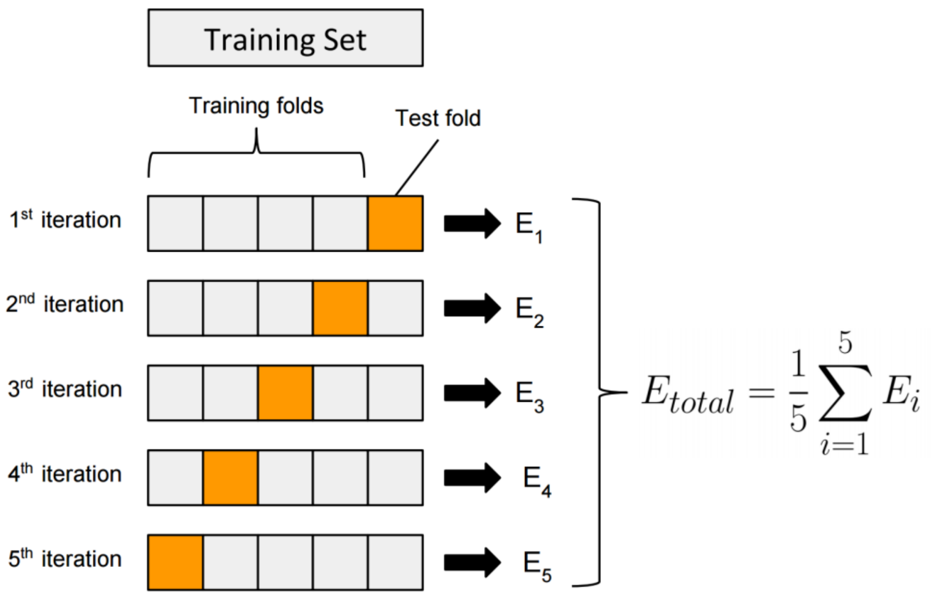


Figure 3 – K-fold cross-validation

Once an appropriate model has been selected and optimized in Weka, the final model can be evaluated with the initial test set with a high degree of confidence the model will perform well on unseen data. Finally, the model is retrained on all the data provided by World of Bargains to produce a concluding set of models, ready for commercial implementation.

Based on the dataset provided, instances 1 – 91 were loaded into Weka for model training/cross-validation. The remaining instances 92 – 134 were set aside for the final model evaluation (test set).

$X_{(1\text{-}91)}$ → Data loaded into Weka for training/K-fold cross-validation

$X_{(92\text{ - }134)}$ → Final test set

4. Encoding Class Labels

Many datasets contain data in a variety of formats, however it is good practice to create a dataset where class labels (‘*Yes*’ or ‘*No*’, for instance) are encoded as integer values (‘0’ or ‘1’) through a mapping process.

5. Feature Scaling

A core element to building effective machine learning algorithms is feature scaling, where a given feature (‘*Floor Space*’, for example) is scaled into a range [0,1] or a standard normal distribution with zero mean.

6. Dimensionality Reduction (Feature selection/extraction)

Some datasets contain highly correlated features which may imply a degree of redundancy in the data. Using a technique called dimensionality reduction, a dataset can be reduced or compressed from a high dimensional space (containing many features) to a lower dimensional subspace. This report outlines the process of feature selection in the results section below.

Modelling selection & hyperparameter tuning

Weka, an open source data mining platform, was used to train, optimize and evaluate performance of a selection of machine learning models described in this section. A high-level model framework for the multivariate regression, logistic regression, multi-layer perceptron (MLP) and decision tree algorithms have been explained below.

Each of the models described below can be configured to elicit different behaviour through selection of the model hyperparameters. This process is often called model hyperparameter optimization and is an empirical process of trial and error to choose the combination of hyperparameters which yields a desirable model performance.

Note: The derivations of these formulas are beyond the scope of this report, however these equations are provided with explanations where possible to illustrate the basic mechanics of these machine learning algorithms. The equation notations have been adopted from Andrew Ng's online program 'Machine Learning'.

Regression

Regression models are used to predict target variables which are on a continuous numeric scale, when provided with inputs. The model is initially trained to determine and estimate the coefficients for a line (univariate) or hyperplane (multivariate) which best fits the training data. Once adequately trained to a satisfactory performance, it can be used for prediction tasks.

Univariate regression is used to uncover the relationship between a single feature and a target variable, however, as the dataset provided contains multiple input features which contribute to the target variable value (profit), a more complex multivariate regression will be implemented to best model the relationships in the dataset. The general form of multi-variate regression is represented in *equation 1* below where, $h_\theta(x)$ is the function used to calculate the predicted value (profit) where θ_i represents a given weight (also called parameters/coefficients) and x_i represents the feature/input value/s:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 \qquad \text{...equation 1}$$

A more general form to represent many weights with many input values is introduced by setting x_0 to 1 in *equation 2*. This approach sums the product of the weight and the input value over the entire dataset or and can be expressed in the vectorised version of *equation 1*, below.

$$h(x) = \sum_{i=0}^{n} \theta_i x_i = \theta^T x \qquad \text{...equation 2}$$

The weights are selected (or iteratively updated) such that the sum of the squared error (between actual and predicted values) on the training data, is minimized (using the ordinary least mean squares method, LMS). The 'cost' function described by *equation 3* below illustrates this method and can be used to update the weights θ_i in the direction to minimize the cost function and hence minimize the difference between a predicted value of $h_\theta(x^{(i)})$ (profit) and the true value $y^{(i)}$ in the training dataset.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \qquad \text{...equation 3}$$

Regression models are commonly optimized (to reduce the cost function) using a technique called Gradient decent where the weights or coefficients of the model are initially set to random values. The sum of the squared error (cost) is calculated and the coefficients are updated towards the direction which minimizes the square error until a satisfactory convergence is obtained. The batch gradient decent update rule is represented by *equation 4* below, where the weights θ_i are initially set to some random value, and simultaneously updated (for all θ_i). In *equation 4* below, θ_j represents a given weight and α is the learning rate (the step size taken in the direction of convergence). The learning rate is then multiplied by the partial derivative of the cost function (described above by *equation 3*) with respect to the weight being updated (the partial derivative simply represents the rate of change with respect to that weight).

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad \text{...equation 4}$$

It should be noted that the Weka allows for the ridge regularization (L2) hyperparameter to be included in a regression model, this induces smoothness and hence allows the complexity of the model to be reduced (where necessary) to avoid overfitting the model to the training data (these regularized equations have been omitted from this report).

Once this model has been trained to a satisfactory standard after optimizing the algorithm hyperparameters, the model is represented by a set of weights (or coefficients). These weights are then multiplied by their respective variable value, to yield a final target variable value solution (a numeric value for profit, for example). This calculation is often done internally, however the model can be represented in the form of *equations 1* or *2*.

Logistic Regression

This powerful statistical model is widely used as a high-performance classification tool for nominal, discrete values. The model aims to assign an instance of data to a class based on its similarity to previous examples of other instances.

The logistic regression model uses similar ideas to the simple regression model described above. A set of weights (or co-efficient) are learned (or estimated using maximum likelihood estimation in Weka) from the training data and are then multiplied by the set of feature values. The intermediate value is passed through a logistic (or sigmoid) function which produces a calculated binary value (that is, a value between 0 and 1). *Equation 5* below defines the hypothesis function representative of the logistic regression model :

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \qquad \text{...equation 5}$$

Where *equation 6* represents the logistic/sigmoid function:

$$g(z) = \frac{1}{1 + e^{-z}} \qquad \text{...equation 6}$$

As with the regression cost function described above, the logistic regression model assumes the use of the same cost function definition, however the difference is attributed to the use of the new logistic regression hypothesis function (*equations 5,6*) when calculating cost for a given set of weights/coefficients.

$$J(\theta) = \frac{1}{2} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 \qquad \text{...equation 7}$$

As mentioned above, the logistic regression model too aims to update the model weights to minimize the computed cost associated with a set of weights. *Equation 8* can be used to describe the update of weights with stochastic gradient – similarly this differs from the regression update rule earlier as we have defined a cost function which calculated on a logistic regression hypothesis function (*Equation 5,6*).

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta) \qquad \text{...equation 8}$$

The description of logistic regression so far has been appropriate for classifying instances into binary classification (for example, a 'yes' or 'no' category represented by an output in the form of a probability mapping, [1,0] represents 'yes' and [0,1] represents 'no'). However, as the goal of this classification task is to categorise a multi-class classification (‘Poor’, ‘Reasonable’, ‘Good’ or ‘Excellent’), an approach called one-vs-all method is used. This extends the binary approach outlined above, however trains the logistic regression classifier for each class category required, to predict the probability that the output calculated, falls into a given category. This output solution can be expressed in a similar fashion as [1,0,0,0] representing ‘Poor’, [0,1,0,0] representing ‘Reasonable’, [0,0,1,0] representing ‘Good‘ and finally [0,0,0,1] representing ‘Excellent’.

The logistic regression model is represented or defined by a matrix of weights (or coefficients) - one vector of coefficients for each class (which are combined to produce the model coefficient matrix). These weights are then multiplied by their respective variable value and passed through the logistic function to yield a final target variable value (profit, for example).

Multi-layer Perceptron (MLP)

This slightly more sophisticated and powerful model, the MLP was designed to loosely mimic the learning functions of a biological brain and can be used for both prediction and classification tasks. Sometimes called neural networks (NN), the neural net topology consists of neurons arranged into layers. *Figure 2* below shows a single layer neural network with input nodes, one hidden layer and output nodes. On a high level, neurons are computational units that receive a weighted input signal and produce an output signal using an activation function.

With the use of the illustration below (*Figure 4*), the training and implementation of the NN is explained. On the left-hand side of the diagram each of the red input nodes receives a feature value for a given instance in the data and passes the value through the red node as an output (no processing done here). The path weights represented by ϴ, are randomly initialized (at the beginning of model training, to break symmetry) and are appropriately updated through back error propagation.

For each of the blue nodes (specifically ‘a_1’, ‘a_2’ & ‘a_3’), the product of the input value and the path weight are calculated. This node value is passed through an activation (sigmoid) function to yield the activation value for that node.

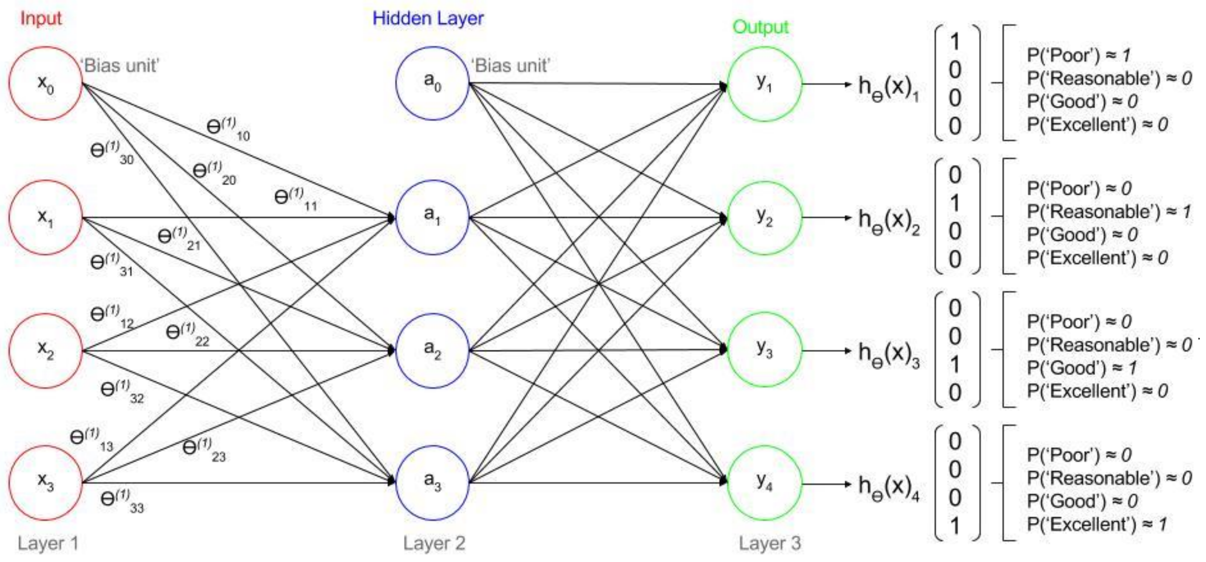

Input
Hidden Layer
Output
'Bias unit'
'Bias unit'
x_0
x_1
x_2
x_3
a_0
a_1
a_2
a_3
y_1
y_2
y_3
y_4
$\Theta^{(1)}_{10}$
$\Theta^{(1)}_{20}$
$\Theta^{(1)}_{30}$
$\Theta^{(1)}_{11}$
$\Theta^{(1)}_{21}$
$\Theta^{(1)}_{31}$
$\Theta^{(1)}_{12}$
$\Theta^{(1)}_{22}$
$\Theta^{(1)}_{32}$
$\Theta^{(1)}_{13}$
$\Theta^{(1)}_{23}$
$\Theta^{(1)}_{33}$
$h_\Theta(x)_1$ [1 0 0 0] P('Poor') ≈ 1, P('Reasonable') ≈ 0, P('Good') ≈ 0, P('Excellent') ≈ 0
$h_\Theta(x)_2$ [0 1 0 0] P('Poor') ≈ 0, P('Reasonable') ≈ 1, P('Good') ≈ 0, P('Excellent') ≈ 0
$h_\Theta(x)_3$ [0 0 1 0] P('Poor') ≈ 0, P('Reasonable') ≈ 0, P('Good') ≈ 1, P('Excellent') ≈ 0
$h_\Theta(x)_4$ [0 0 0 1] P('Poor') ≈ 0, P('Reasonable') ≈ 0, P('Good') ≈ 0, P('Excellent') ≈ 1
Layer 1
Layer 2
Layer 3

Figure 4 - General architecture of a neural network for performance category classification

$$a_1 = g(\Theta^{(1)}_{10}x_0 + \Theta^{(1)}_{11}x_1 + \Theta^{(1)}_{12}x_2 + \Theta^{(1)}_{13}x_3)$$ …equation 10

$$a_2 = g(\Theta^{(1)}_{20}x_0 + \Theta^{(1)}_{21}x_1 + \Theta^{(1)}_{22}x_2 + \Theta^{(1)}_{23}x_3)$$ …equation 9

$$a_3 = g(\Theta^{(1)}_{30}x_0 + \Theta^{(1)}_{31}x_1 + \Theta^{(1)}_{32}x_2 + \Theta^{(1)}_{33}x_3)$$ …equation 11

Here the sigmoid logistic function is represented by *equation 12* below:

$$g(\theta^T x) = \frac{1}{1+e^{-\theta^T x}} \quad \text{...equation 12}$$

To explain how we achieve a matrix of path weights which represents the training data well, we need to define a cost function similar that used in the regression and logistic regression. As a reminder, the purpose of the cost function is to show us how close our predicted values are to the true values from the training set. A high cost implies the model does not represent the training set well and the weights need be adjusted. Conversely, a low cost implies the model represents the training data well and change to the weight values may not be necessary. The cost function for the neural network is defined below, where the function contains some nested summations to account for the multiple output nodes and summing over all the training examples in the dataset. The first summation sign sums over the training example cost (the difference between the predicted value $h_\Theta(x^{(i)})$ and the actual value, $y^{(i)}$) over all the instances in the training set (represented by m). The second summation sign accounts for the multiple output nodes we have designed our neural network to have (for the classification task in this report, K = 4).

$$J(\Theta) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K}\left[y_k^{(i)}\log((h_\Theta(x^{(i)}))_k) + (1-y_k^{(i)})\log(1-(h_\Theta(x^{(i)}))_k)\right] \quad \text{...equation 13}$$

To effectively minimize the cost function above, the path weights for the network need to be appropriately chosen. This is done with back error propagation or 'backpropagation' where the error for a set of given weights is calculated and worked back through the network to update the weights in the direction which minimizes the error. Each row of the training dataset is read into the neural network and propagated forward through the network (multiplied by the random weight and passed through the sigmoid function) until it reaches the final output layer. The difference between this output and the true value in the dataset is defined as the error. This error is propagated back through the network, as the path weights are already known. This reveals the error associated with a given activation node, and when passing it back through the sigmoid activation function (that is the derivative of the activation function), the error associated with that path weight, can be obtained. This path weight/s in turn are updated to reduce the error. The weights for a neural network can be updated for each data point (stochastic gradient decent), or once for a complete pass of the data set (batch gradient decent).

MLP's (neural networks) are powerful in learning and representing the complex relationships between the training data and an output variable. And as the output depends on the choice of activation function (*equation 12*), the MLP output format can be chosen to correspond to the required prediction (linear activation functions are applied), binary classification or multi-class classification task (in which cases sigmoid functions are applied).

Although the MLP can perform well on both prediction and classification tasks, they have been known to require a high degree of optimisation and tuning for even the most basic multilayer perceptron. As the complexity of the MLP grows, so does the number of parameters to be tuned and computational power required, to effectively train and run the model.

Decision Tree

The final model to be described in this report is the decision tree, which falls under a branch of decision analysis. A decision tree commonly consists of a root node, one or more decision nodes, branches and leaf nodes, where a node represents a single variable. *Figure 5* below illustrates the framework of a decision tree, where the top node is the root node. Each level splits the data according to different attributes, where non-leaf nodes represent attributes (this is where a 'decision' is made), the leaf nodes represent the classified/predicted variable

As feature values are 'fed' into the decision tree, the input feature instance will fall down the branches of the tree and is finally classified once it reaches a leaf node. Decision trees are valued for their interpretability and can be used to effectively carry out both prediction and classification tasks.

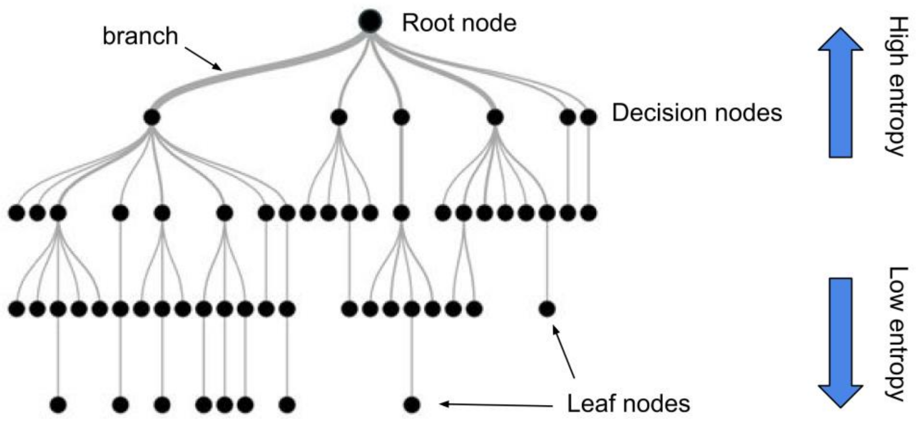


Figure 5 - General architecture of a decision tree

Many decision trees use the ID3 algorithm which uses entropy and information gain to build the decision tree and decide which attributes should be chosen to give the best split. Entropy (in the machine learning, decision tree context) is the uncertainty associated at a given point on the tree, and its this information gain concept which is frequently applied to build decision trees effectively.

Given the choice to select one of two attributes at a given decision node, the information gain is calculated for each attribute (fabricated figures, for illustration):

Information Gain for "Staff" = 0.05
Information Gain for "Floor Space" = 0.19

As the information gain associated with the attribute floor space, this will be chosen in preference of the two as to maximize the Decision Tree information gain and hence reduce the system entropy. This process is followed when evaluating all the available attributes when building or structuring the tree such that it receives inputs from input feature values which add/contribute the most information to the system, first and hence reduce the system entropy the fastest.

Results and Errors

During this consultancy investigation, all four of these models outlined above, were explored and considered as potential candidates for final solutions. As mentioned in the *Introduction* section, the two models chosen are to achieve two separate functions. The first model is to perform a prediction (regression) on the expected commercial profit, for a shop instance containing a set of feature values. As this is a prediction task, the **Multivariate Regression** and **Multilayer Perceptron** (MLP) models were assigned as candidates to fulfil the functional requirement.

The second model chosen, is to serve effectively in classification of these instances into discrete nominal categories of performance, namely 'Poor', 'Reasonable', 'Good' and 'Excellent'. Based on the functional requirement to classify an instance with variable feature values into discrete categories/classes, the **Logistic Regression**, **Multilayer Perceptron** and **Decision Tree** models were assigned as potential candidates for this to fulfil the functional requirement.

Note: all models implemented the 5-fold Cross-validation for model evaluation

Profit Prediction

The section below outlines the variation and evaluation of meaningful hyperparameters which yielded different model performances. As described earlier the predictive target variable has been defined as the expected profit a given shop makes, given a feature matrix **X** (representing all the attributes) and target variable **y** (shown below):

X = [Staff, Floor Space, Window, Car Park, Demographic Score, Location, 40min population, 30min population, 20min population, 10min population, Store age, Clearance Space, Competition Score]

y = [Profit]

Figure 2 in the *Data Preparation* section, showed the variable value distributions prior to any preprocessing. *Figure 6* below shows the variable value distributions post-processing (for the all attributes, prior to the Test/Train split) for the profit prediction task:

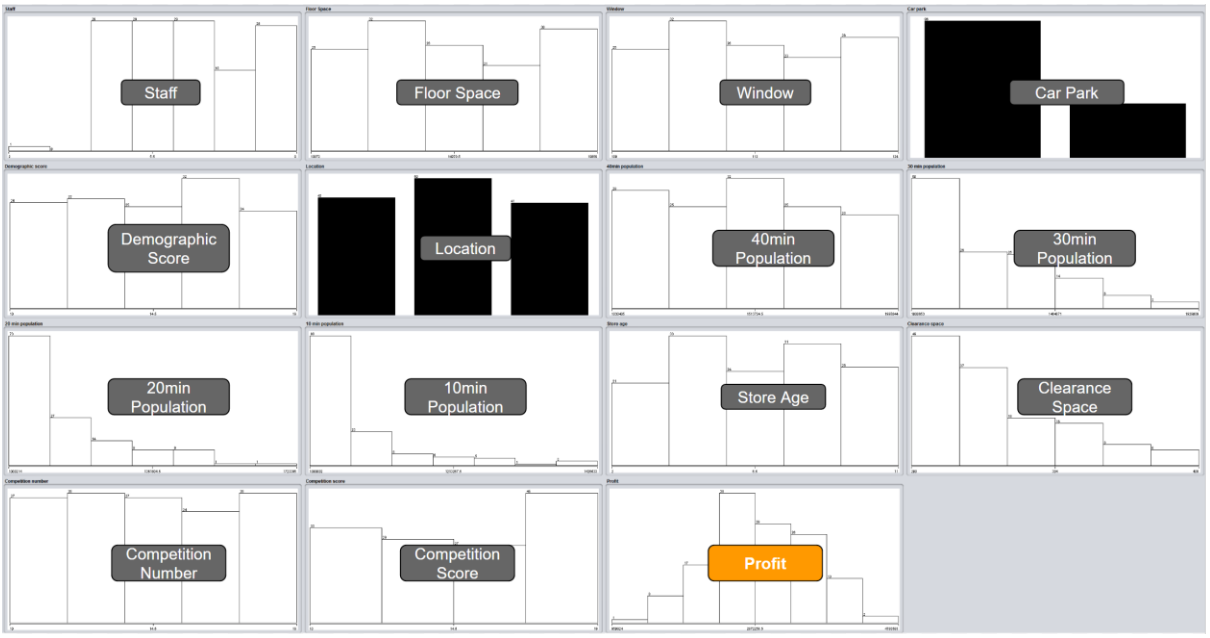
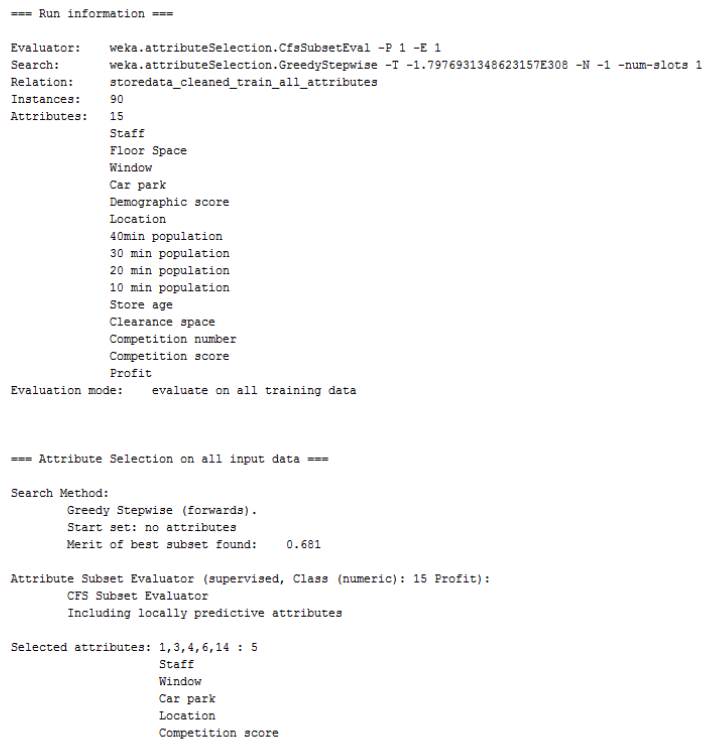


Figure 6 - Complete post-preprocessing variable value distributions for the prediction model

A feature selection algorithm was implemented on the dataset to extract a subset, of the original attributes provided (see matrix X, above), which contribute the most to the model performance. This process can be done in Weka to calculate the correlation between each attribute and the output variable (Performance). By selecting high correlation features and removing irrelevant features there are a few advantages to be gained:

- Avoids overfitting (the generalization error of the model is reduced by removing noise)
- Improves Accuracy (the model is trained without misleading data)
- Improves computational efficiency
- Reduces data acquisition and hence project cost (reducing the number of attributes required to train and use the model, will lower the data mining investment cost)

For extracting the subset of relevant features in the prediction task, the training dataset (for the prediction task) was loaded into Weka. In the 'Select attributes' tab, the Attribute Evaluator chosen was the 'CfsSubsetEval' which evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them. The search method chosen was the 'GreedyStepwise' (assessed using the full training set containing all the attributes), the results of which can be seen below:



```
=== Run information ===

Evaluator:    weka.attributeSelection.CfsSubsetEval -P 1 -E 1
Search:       weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1 -num-slots 1
Relation:     storedata_cleaned_train_all_attributes
Instances:    90
Attributes:   15
              Staff
              Floor Space
              Window
              Car park
              Demographic score
              Location
              40min population
              30 min population
              20 min population
              10 min population
              Store age
              Clearance space
              Competition number
              Competition score
              Profit
Evaluation mode:    evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
	Greedy Stepwise (forwards).
	Start set: no attributes
	Merit of best subset found:    0.681

Attribute Subset Evaluator (supervised, Class (numeric): 15 Profit):
	CFS Subset Evaluator
	Including locally predictive attributes

Selected attributes: 1,3,4,6,14 : 5
                     Staff
                     Window
                     Car park
                     Location
                     Competition score
```

Figure 7 – Feature selection algorithm results in Weka (Prediction)

The algorithm identified a subset of (more) relevant features, the chosen subset of which has been summarized in the report convention, below. This subset of features was used to evaluate the model performances in this section, in addition to the original training dataset.

X = [Staff, Window , Car Park, Location, Competition Score]

y = [Profit]

In addition, the ZeroR algorithm was implemented to determine the baseline of performance for algorithm comparison purposes. The results reflected correlation coefficient of -0.2049 and RMS error of £700,377 for both the attribute selected subset and when all the attributes in the dataset were used. The objective here is to find improvements upon the model performance produced using ZeroR by exploring and selecting the optimal hyperparameters.

For the performance summary of the *prediction* models two key metrics were taken into consideration – these have been explained below:

Correlation Coefficient → This describes how well the predicted value correlates to the actual value during the validation/test process. A value of 0 indicates no correlation and in contrast a value of 1 indicates a high correlation between a predicted value and the actual value of the output.

Root Mean Squared Error → This describes the average amount of error made on the test set in the profit variable which we are trying to predict. This allows us to understand on average, by how much we are getting a given prediction wrong when compared to the actual value given.

Regression

For evaluation of the regression models, a series of combinations were chosen to reflect the various hyperparameters for regression. The two hyperparameters explored were the 'Att. Selection Method' and the 'Ridge' Regularization parameter.

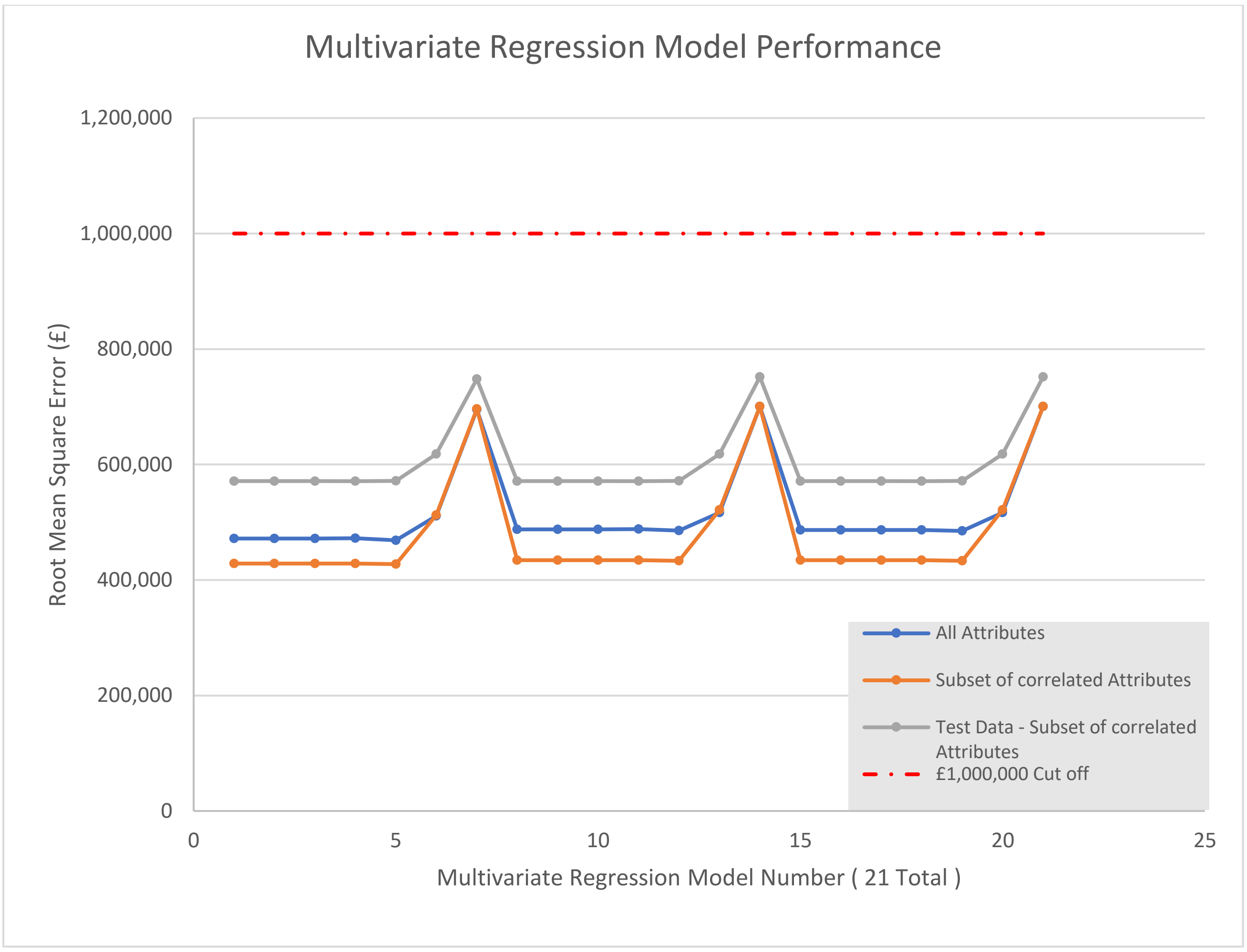
Attribute selection method → this variable defines which algorithm Weka employs to select the best subset of attributes from the dataset provided.

Ridge regularization → this value influences the extent to which the model complexity, is reduced. This is done by preventing any learned coefficient, from becoming too large.

Weka provides three algorithms for attribute selection (although this is redundant when the attribute selected subset is evaluated) and uses the ridge regularization technique to reduce the complexity of the model which reduces overfitting. The table below illustrates the model performance for a given set of hyperparameters.

Model Number	Hyperparameters		Training Performance (All Attributes)		Training Performance (Subset of correlated Attributes)		Test Performance (Subset of correlated Attributes)	
	Att. Selection Method.	Ridge	Correlation Coefficient	RMS Error (£)	Correlation Coefficient	RMS Error (£)	Correlation Coefficient	RMS Error (£)
1	None	1.0E-8	0.7487	471,841	0.7878	428,460	0.6792	571,370
2	None	1.0E-6	0.7487	471,841	0.7878	428,460	0.6792	571,370
3	None	1.0E-4	0.7487	471,840	0.7878	428,460	0.6792	571,370
4	None	1.0E-2	0.7483	472,323	0.7878	428,450	0.6792	571,371
5	None	1.0E-0	0.7497	468,786	0.7881	427,505	0.6788	571,518
6	None	1.0E2	0.7417	510,784	0.785	512,715	0.6634	618,315
7	None	1.0E4	-0.0541	695,031	-0.0961	696,442	0.6546	748,038
8	M5	1.0E-8	0.733	487,884	0.7816	434,232	0.6792	571,370
9	M5	1.0E-6	0.733	487,884	0.7816	434,232	0.6792	571,370
10	M5	1.0E-4	0.733	487,880	0.7816	434,232	0.6792	571,370
11	M5	1.0E-2	0.7325	488,252	0.7816	434,222	0.6792	571,371
12	M5	1.0E-0	0.7333	485,452	0.7818	433,309	0.6788	571,518
13	M5	1.0E2	0.7309	516,890	0.7675	521,563	0.6634	618,315
14	M5	1.0E4	-0.2049	700,377	-0.2049	700,378	0.000	751,532
15	Greedy	1.0E-8	0.7344	486,652	0.7816	434,232	0.6792	571,370
16	Greedy	1.0E-6	0.7344	486,652	0.7816	434,232	0.6792	571,370
17	Greedy	1.0E-4	0.7344	486,651	0.7816	434,232	0.6792	571,370
18	Greedy	1.0E-2	0.7343	486,707	0.7816	434,222	0.6792	571,371
19	Greedy	1.0E-0	0.7334	485,310	0.7818	433,309	0.6788	571,518
20	Greedy	1.0E2	0.7309	516,890	0.7675	521,563	0.6634	618,315
21	Greedy	1.0E4	-0.2049	700,377	-0.2049	700,378	0.000	751,532

Table 1 - Multivariate Regression Model Results



Graph 1 - Multivariate Regression Model Performance

Multilayer Perceptron (MLP)

A similar approach was used to evaluate the MLP model performance, for given combinations of hyperparameters. The hyperparameters evaluated for the MLP were the 'Number of hidden layers', the 'Learning rate' and the 'Momentum'.

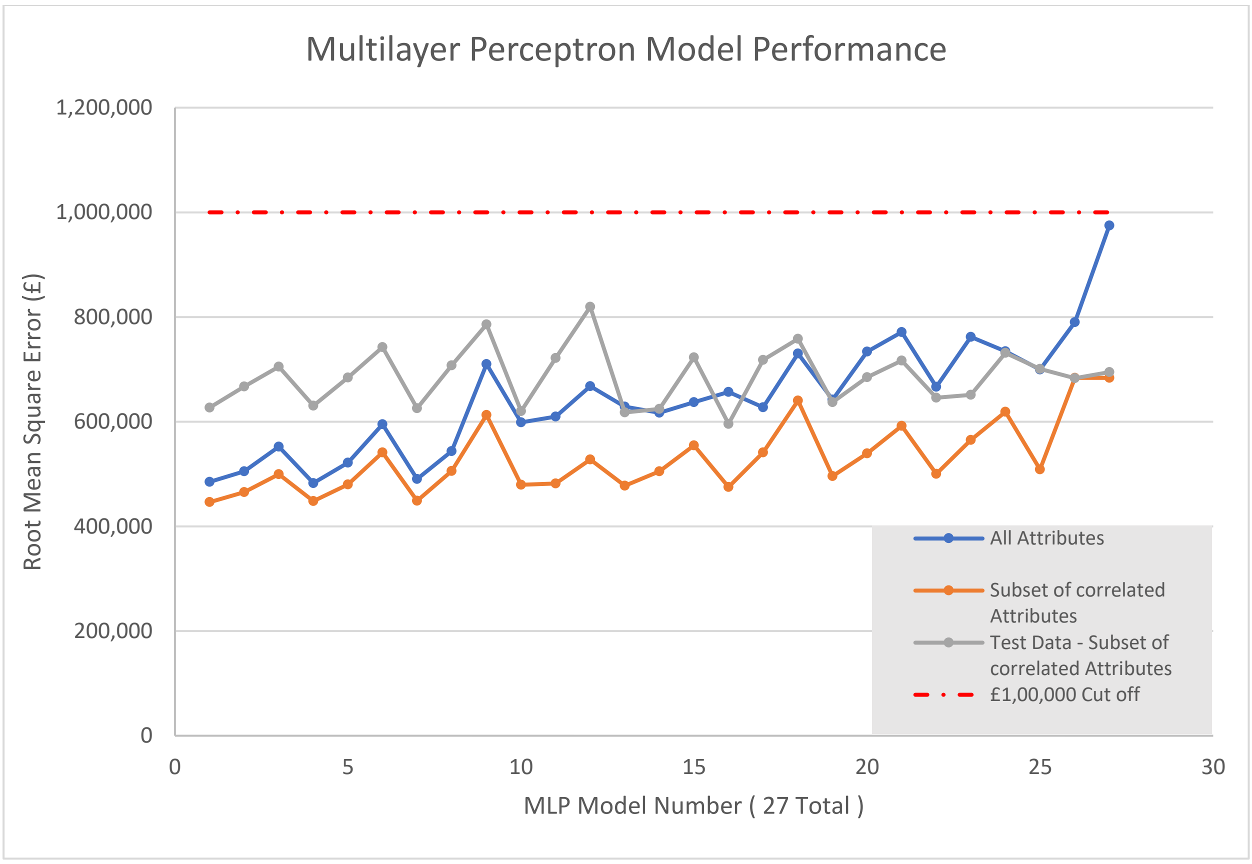
Number of hidden layers → this variable defines how many hidden layers are used to discover relationships in the data. A single hidden layer MLP can often be appropriate to model simple relationships in the data, however for more complex relationships more hidden layers might be necessary to adequately represent the relationships in the dataset.

Learning rate → this value influences how quickly the network learns, a low learning rate will result in slow (or time consuming) learning of weights while a high learning rate makes the weights and objective function diverge resulting in no progressive learning.

Momentum → this value influences the extent to which the model can escape local minima during backpropagation (get stuck in a set of suboptimal weights).

Model Number	Hyperparameters			Training Performance (All Attributes)		Training Performance (Subset of correlated Attributes)		Test Performance (Subset of correlated Attributes)	
	No. of hidden layers	Learning Rate (0.1-0.3)	Momentum	Correlation Coefficient	RMS Error (£)	Correlation Coefficient	RMS Error (£)	Correlation Coefficient	RMS Error (£)
1	1	0.1	0.2	0.7368	484,782	0.7718	446,224	0.6906	626,996
2	1	0.2	0.2	0.7199	505,027	0.755	465,217	0.6873	667,440
3	1	0.3	0.2	0.6736	552,539	0.7215	499,695	0.6805	705,317
4	1	0.1	0.4	0.7391	482,782	0.7697	448,531	0.6912	630,800
5	1	0.2	0.4	0.7044	521,386	0.7409	479,912	0.6883	684,374
6	1	0.3	0.4	0.631	595,167	0.6846	541,142	0.681	742,616
7	1	0.1	0.6	0.7296	490,617	0.7682	449,157	0.6911	625,794
8	1	0.2	0.6	0.6826	543,442	0.7123	505,867	0.6903	707,431
9	1	0.3	0.6	0.5182	710,250	0.6176	612,804	0.6766	786,106
10	2	0.1	0.2	0.6072	598,572	0.7433	479,212	0.7091	620,013
11	2	0.2	0.2	0.6078	609,532	0.7502	481,867	0.6162	721,592
12	2	0.3	0.2	0.5478	667,997	0.696	527,637	0.5334	819,532
13	2	0.1	0.4	0.5782	628,831	0.7465	477,394	0.7102	617,637
14	2	0.2	0.4	0.5928	616,936	0.7243	504,993	0.7138	624,289
15	2	0.3	0.4	0.6269	637,542	0.6633	554,441	0.6413	722,619
16	2	0.1	0.6	0.5597	656,655	0.7481	475,449	0.7115	595,906
17	2	0.2	0.6	0.5825	627,469	0.6836	541,452	0.6226	717,886
18	2	0.3	0.6	0.5212	730,155	0.622	640,481	0.463	758,237
19	3	0.1	0.2	0.6171	642,228	0.72	495,974	0.6668	637,496
20	3	0.2	0.2	0.5003	733,792	0.6992	539,487	0.657	685,065
21	3	0.3	0.2	0.5971	770,879	0.6641	592,203	0.6476	716,931
22	3	0.1	0.4	0.5805	666,739	0.7178	500,215	0.6602	645,510
23	3	0.2	0.4	0.5498	761,912	0.6774	565,391	0.6658	651,041
24	3	0.3	0.4	0.6094	734,459	0.6533	618,758	0.6178	731,191
25	3	0.1	0.6	0.5908	699,577	0.713	509,080	0.6163	700,888
26	3	0.2	0.6	0.5475	790,098	0.6222	683,573	0.5941	682,974
27	3	0.3	0.6	0.4741	974,737	0.6306	683,745	0.6278	694,881

Table 2 - Multilayer Perceptron Model Results



Graph 2 - Multilayer Perceptron Model Performance

Performance Classification

The section below outlines the dataset used and variation of relevant hyperparameters which yielded different model performances for the classification task. Here, the classification target variable has been defined as the set of possible performance categories, namely *'Poor'*, *'Reasonable'*, *'Good'*, *'Excellent'*, where the initial feature matrix X and target variable y has been shown below:

X = [Staff, Floor Space, Window, Car Park, Demographic Score, Location, 40min population, 30min population, 20min population, 10min population, Store age, Clearance Space, Competition Score]

y = [Performance] $\in$ {'Poor', 'Reasonable', 'Good', 'Excellent'}

Figure 8 below shows the variable value distribution for the complete performance classification dataset:

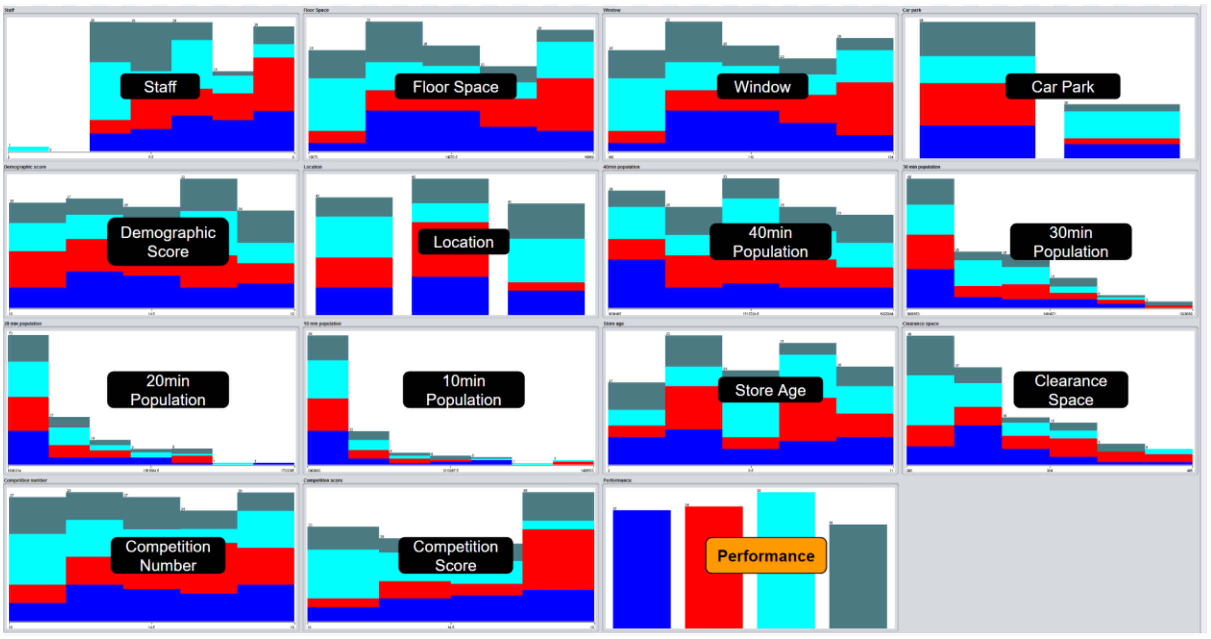
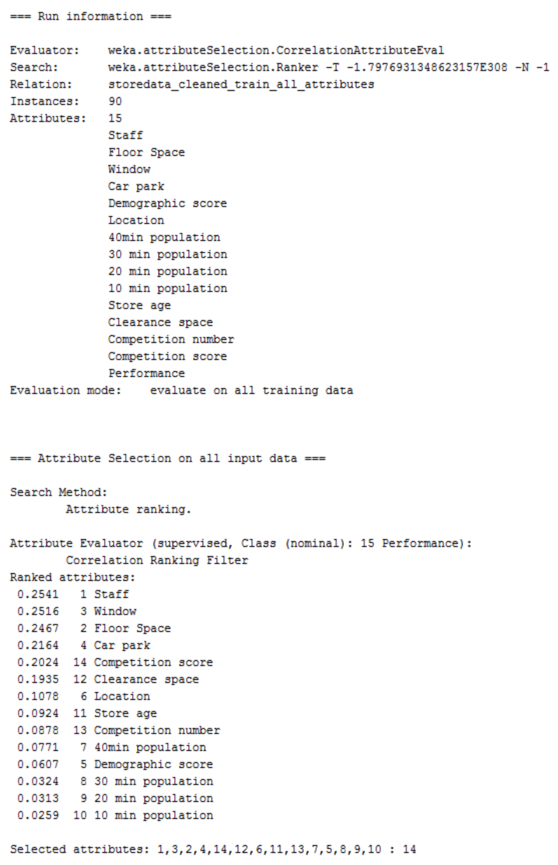


Figure 8 - Complete variable value distributions for the classification model

As with the feature selection process in the prediction task above, the 'Attribute Evaluator' function was used to perform a feature selection for the classification task. Weka uses Pearson's correlation coefficient to determine of the available attributes, those which have a positive (or negative) correlation to the output variable. The results in the following sections compare the model performance using all the attributes available to the model performance for models trained on the selected subset of 'meaningful' features. The results from the correlation based feature selection can be seen below (*Figure 9*), where the cut off value was decided to be 0.1 for relevant attributes:



```
=== Run information ===

Evaluator:    weka.attributeSelection.CorrelationAttributeEval
Search:       weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     storedata_cleaned_train_all_attributes
Instances:    90
Attributes:   15
              Staff
              Floor Space
              Window
              Car park
              Demographic score
              Location
              40min population
              30 min population
              20 min population
              10 min population
              Store age
              Clearance space
              Competition number
              Competition score
              Performance
Evaluation mode:    evaluate on all training data



=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 15 Performance):
        Correlation Ranking Filter
Ranked attributes:
 0.2541   1 Staff
 0.2516   3 Window
 0.2467   2 Floor Space
 0.2164   4 Car park
 0.2024  14 Competition score
 0.1935  12 Clearance space
 0.1078   6 Location
 0.0924  11 Store age
 0.0878  13 Competition number
 0.0771   7 40min population
 0.0607   5 Demographic score
 0.0324   8 30 min population
 0.0313   9 20 min population
 0.0259  10 10 min population

Selected attributes: 1,3,2,4,14,12,6,11,13,7,5,8,9,10 : 14
```

Figure 9 – Feature selection algorithm results in Weka (Classification)

The 'trimmed' dataset on which the models were trained on, can be seen below with the order shown indicates the relative relevance to the output variable, Performance):

X = [Staff, Window, Floor Space, Car Park, Competition Score, Clearance Space, Location]

y = [Performance] $\in$ *{'Poor', 'Reasonable', 'Good', 'Excellent'}*

The metrics used to evaluate model performance were defined to be the 'Classification accuracy' and the 'Total Misclassified Errors'

Classification accuracy → Given as a percentage, this metric value represents the ratio of correctly classified instances from the total of number of instance in the training set.

Total number of misclassified errors → the number of misclassified instances (erroneous classifications).

As with the prediction (regression) task, the ZeroR algorithm was implemented to determine the baseline of performance for algorithm comparison purposes. The results of which reflected a classification accuracy of **27.78%** (with 65 of 90 instances misclassified) for both the attribute selected subset and when all the attributes were used.

Logistic Regression

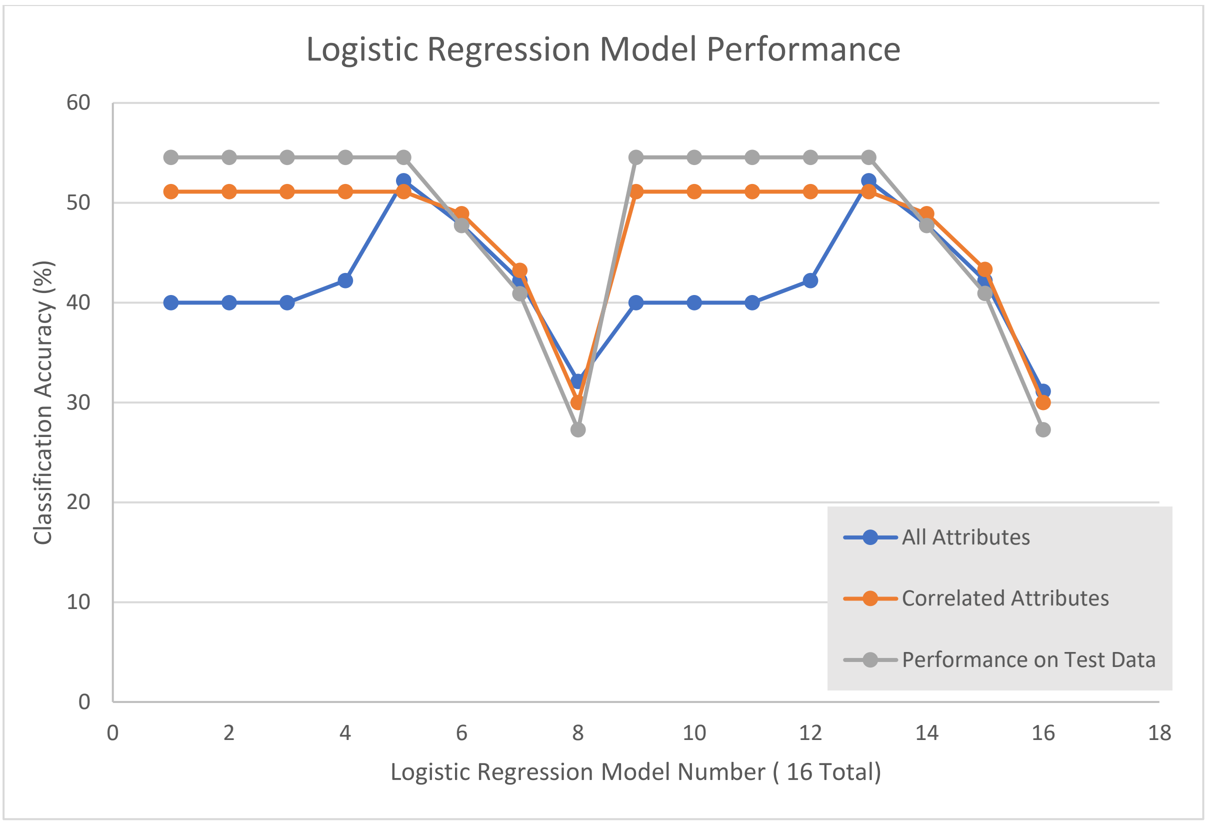
For evaluation of the Logistic Regression models, the hyperparameters considered for variation were the 'Conjugate Gradient Decent' and 'Ridge' Regularization parameter.

Conjugate Gradient Decent → this option defines whether the conjugate gradient decent algorithm was applied during learning or not. This algorithm is used to more effectively/directly find local minima by automatically updating the learning rate as the model is trained.

Ridge regularization → this value influences the extent to which the model complexity, is reduced. This is done by preventing any learned coefficient, from becoming too large.

Model Number	Hyperparameters		Training Performance (All Attributes)		Training Performance (Subset of correlated Attributes)		Test Performance (Subset of correlated Attributes)	
	Conjugate Gradient Decent	Ridge	Classification Accuracy (%)	Total Misclassified Errors (90 total)	Classification Accuracy (%)	Total Misclassified Errors (90 total)	Classification Accuracy (%)	Total Misclassified Errors (44 total)
1	False	1.0E-10	40.00	54	51.11	44	54.55	20
2	False	1.0E-8	40.00	54	51.11	44	54.55	20
3	False	1.0E-6	40.00	54	51.11	44	54.55	20
4	False	1.0E-4	42.22	52	51.11	44	54.55	20
5	False	1.0E-2	52.22	43	51.11	44	54.55	20
6	False	1.0E-0	47.78	47	48.89	46	47.73	23
7	False	1.0E2	42.22	52	43.22	51	40.90	26
8	False	1.0E4	32.11	62	30.00	63	27.27	32
9	True	1.0E-10	40.00	54	51.11	44	54.55	20
10	True	1.0E-8	40.00	54	51.11	44	54.55	20
11	True	1.0E-6	40.00	54	51.11	44	54.55	20
12	True	1.0E-4	42.22	52	51.11	44	54.55	20
13	True	1.0E-2	52.22	43	51.11	44	54.55	20
14	True	1.0E-0	47.78	47	48.89	46	47.72	23
15	True	1.0E2	42.22	52	43.33	51	40.91	26
16	True	1.0E4	31.11	62	30.00	63	27.27	32

Table 3 – Logistic Regression Model Results



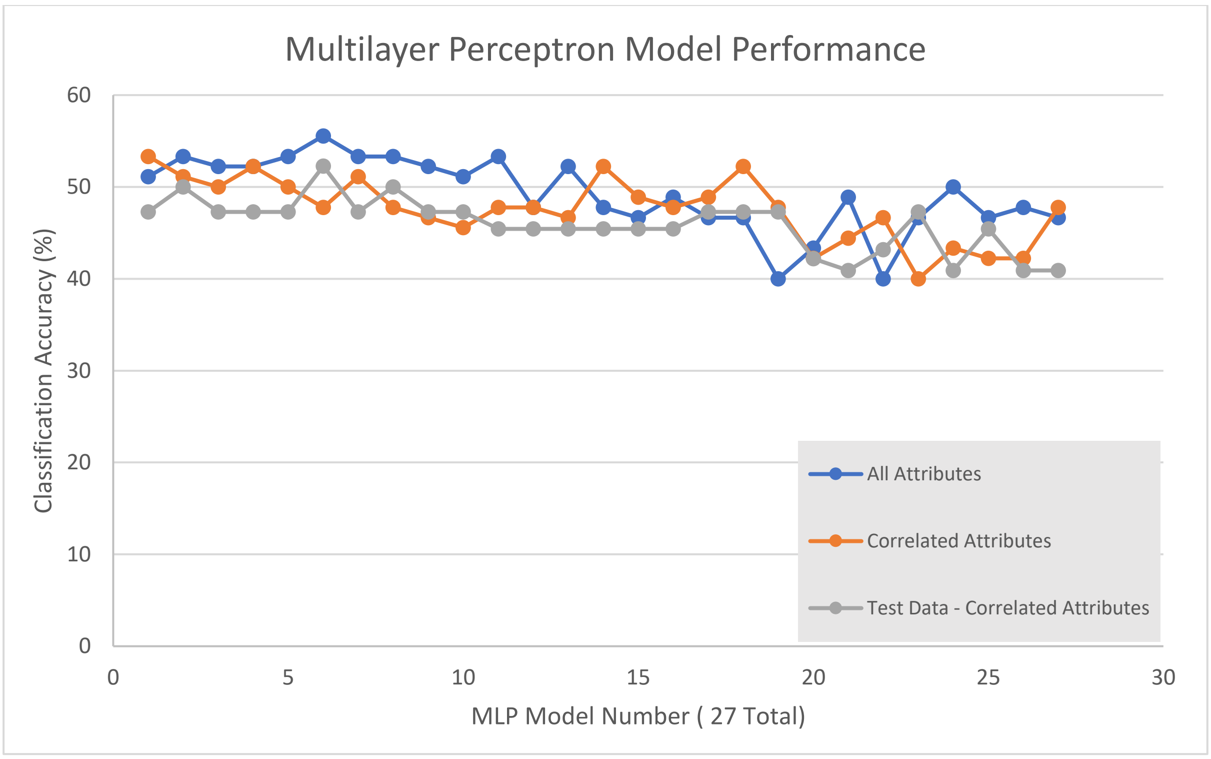
Graph 3 – Logistic Regression Model Performance

Multilayer Perceptron

Similarly, the same hyperparameters described in the prediction MLP, were used to evaluate the MLP model performance, for given combinations of hyperparameters.

Model Number	Hyperparameters			Training Performance (All Attributes)		Training Performance (Subset of correlated Attributes)		Test Performance (Subset of correlated Attributes)	
	No. of hidden layers	Learning Rate (0.1-0.3)	Momentum	Classification Accuracy (%)	Total Misclassified Errors (90 total)	Classification Accuracy (%)	Total Misclassified Errors (90 total)	Classification Accuracy (%)	Total Misclassified Errors (44 total)
1	1	0.1	0.2	51.11	44	53.33	42	47.27	23
2	1	0.2	0.2	53.33	42	51.11	44	50.00	22
3	1	0.3	0.2	52.22	43	50.00	45	47.27	23
4	1	0.1	0.4	52.22	43	52.22	43	47.27	23
5	1	0.2	0.4	53.33	42	50.00	45	47.27	23
6	1	0.3	0.4	55.56	40	47.78	47	52.27	21
7	1	0.1	0.6	53.33	42	51.11	44	47.27	23
8	1	0.2	0.6	53.33	42	47.78	47	50.00	22
9	1	0.3	0.6	52.22	43	46.67	48	47.27	23
10	2	0.1	0.2	51.11	44	45.56	49	47.27	23
11	2	0.2	0.2	53.33	42	47.78	47	45.45	24
12	2	0.3	0.2	47.78	47	47.78	47	45.45	24
13	2	0.1	0.4	52.22	43	46.67	48	45.45	24
14	2	0.2	0.4	47.78	47	52.22	43	45.45	24
15	2	0.3	0.4	46.67	48	48.89	46	45.45	24
16	2	0.1	0.6	48.89	46	47.78	47	45.45	24
17	2	0.2	0.6	46.67	48	48.89	46	47.27	23
18	2	0.3	0.6	46.67	48	52.22	43	47.27	23
19	3	0.1	0.2	40	54	47.78	47	47.27	23
20	3	0.2	0.2	43.33	51	42.22	52	42.18	35
21	3	0.3	0.2	48.89	46	44.44	50	40.91	26
22	3	0.1	0.4	40.00	54	46.67	48	43.18	25
23	3	0.2	0.4	46.67	48	40.00	54	47.27	23
24	3	0.3	0.4	50.00	45	43.33	51	40.91	26
25	3	0.1	0.6	46.67	48	42.22	52	45.45	24
26	3	0.2	0.6	47.78	47	42.22	51	40.91	26
27	3	0.3	0.6	46.67	48	47.78	47	40.91	26

Table 4 – Multilayer Perceptron Model Results



Graph 4 – Multilayer Perceptron Model Performance

Decision Tree

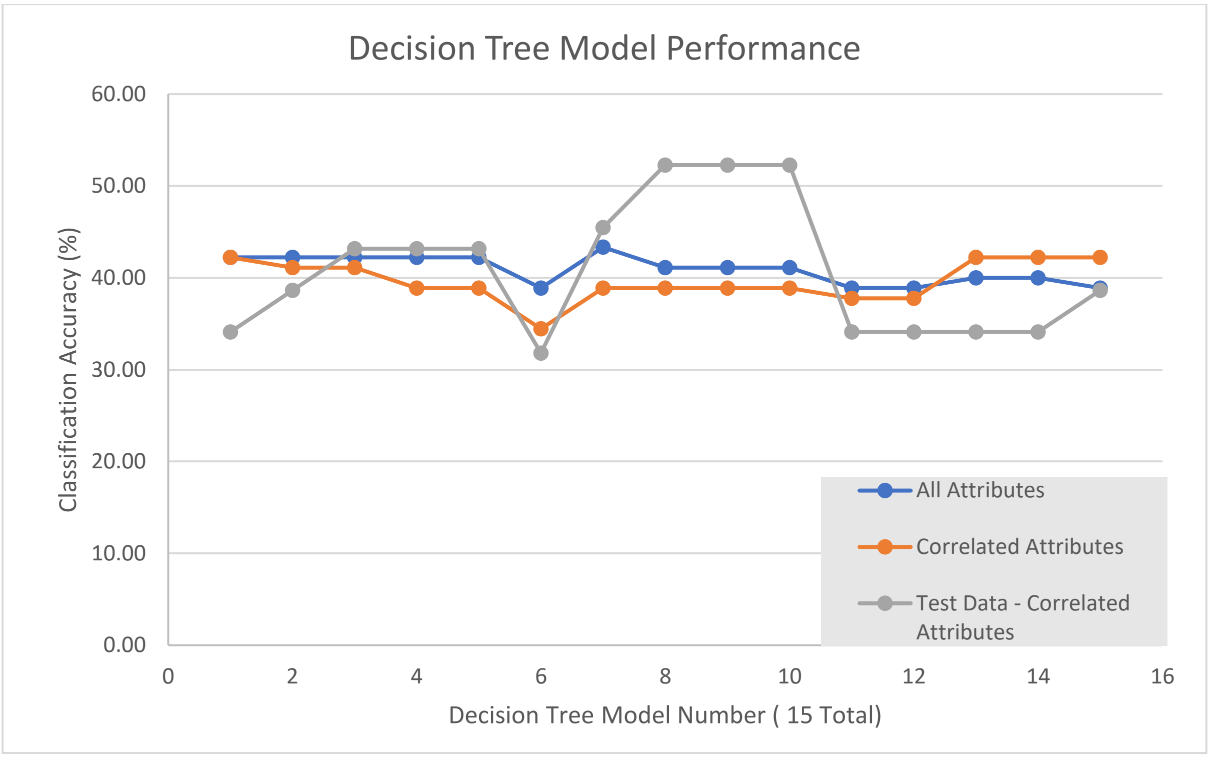
Finally, the table below represents the performance results from the decision tree algorithm. Weka's J48 decision tree was used to build a classification model The hyperparameters evaluated were 'unpruned and 'minNumObj'.

confidenceFactor → a small confidence factor will force a high degree of pruning (pruning is used to reduce the size of the decision tree by removing parts of the tree which add little contribution to classification of an instance)

minNumObj → The minimum number of instances per leaf

Model Number	Hyperparameters		Training Performance (All Attributes)		Training Performance (Subset of correlated Attributes)		Test Performance (Subset of correlated Attributes)	
	confidenceFactor	minNumObj'	Classification Accuracy (%)	Total Misclassified Errors (90 total)	Classification Accuracy (%)	Total Misclassified Errors (90 total)	Classification Accuracy (%)	Total Misclassified Errors (44 total)
1	0.1	1	42.22	52	42.22	52	34.09	29
2	0.2	1	42.22	52	41.11	53	38.64	27
3	0.3	1	42.22	52	41.11	53	43.18	25
4	0.4	1	42.22	52	38.89	55	43.18	25
5	0.5	1	42.22	52	38.89	55	43.18	25
6	0.1	2	38.89	55	34.44	59	31.82	30
7	0.2	2	43.33	51	38.89	55	45.45	24
8	0.3	2	41.11	53	38.89	55	52.27	21
9	0.4	2	41.11	52	38.89	55	52.27	21
10	0.5	2	41.11	52	38.89	55	52.27	21
11	0.1	3	38.89	55	37.78	56	34.09	29
12	0.2	3	38.89	55	37.78	56	34.09	29
13	0.3	3	40.00	54	42.22	52	34.09	29
14	0.4	3	40.00	54	42.22	52	34.09	29
15	0.5	3	38.89	55	42.22	52	38.64	27

Table 5 – Decision Tree Model Results



Graph 5 – Decision Tree Model Performance

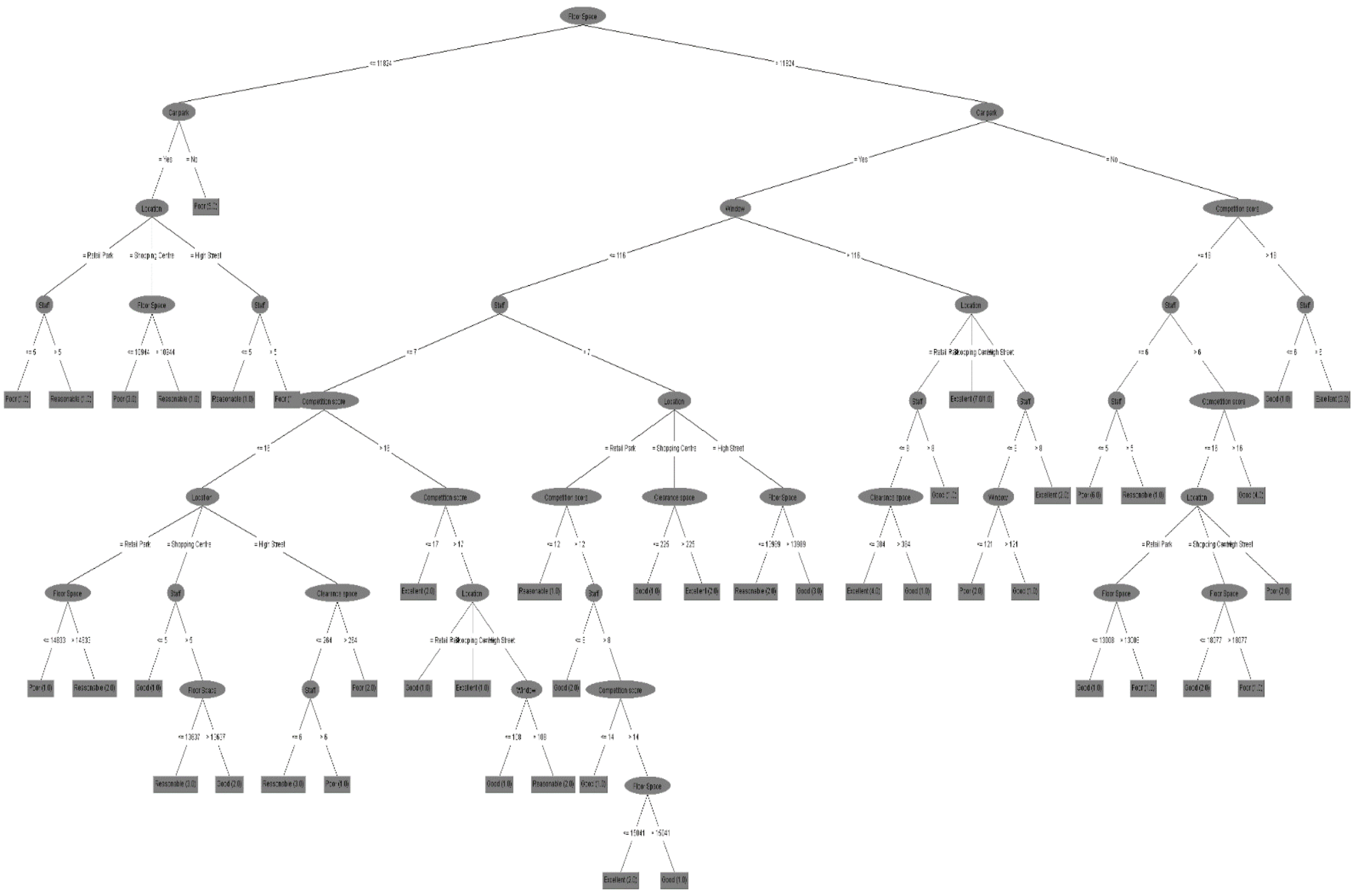


Figure 10 – The Decision Tree Visualized in Weka

Recommendations

It should be noted that the machine learning prediction models explored in this report can be built and implemented to yield *estimates* for predicting the commercial profit/revenue generated for a set of given store features. And, the machine learning classification models can be produced to *estimate* a classification performance category for a shop with given features. They are statistical models and no guarantee can be made they will predict or classify with complete certainty, for the model performances depend heavily on the data on which they have been trained. In addition, external forces may influence the performance of a given shop and these too should be taken into consideration when exploring growth opportunities or evaluating shop performance.

The feature selection algorithms were implemented on both the datasets for the *prediction* and *classification* tasks, to determine those attributes from the dataset which had higher influence and (hence higher correlation coefficient) on the target variable. The general model performance using all the attributes showed trivial improvement (reduction in root mean square error in GBP, or classification accuracy as a percentage) when compared to the model performance using a subset of selected features. Conscious of the cost implications associated with the data collection, it is suggested the subsets of attributes identified below be used to make predictions/classifications, therein minimizing the associated data collection costs.

Based on the findings in this report, two final models are suggested for commercial implementation based on the evaluated performances:

For the *prediction* task, the Multivariate Regression and Multilayer Perceptron models were trained and evaluated to fulfil the functional requirement. Based on the findings outlined in this report, *Model 7* from the Multilayer Perceptron models built, is suggested for implementation for profit prediction tasks *Model 7* represented a suitably low Root Mean Square Error on both the cross-validation and test performances (an average of £537,475, suitably below the cut off of £1,000,000) and this suggests the model adequately describes the relationship, between the input variables and the target profit value.

X = [Staff, Window , Car Park, Location, Competition Score]

y = [Profit]

The classification task was addressed by building and evaluating a Logistic Regression model, a Multilayer Perceptron model and a Decision Tree model. The final model chosen to fulfil the classification task is the Logistic Regression *Model 4* (an average classification accuracy, based on the cross-validation and test accuracy, of 47.76%) . The motivation stems from the consistency of the Logistic Regression model performance in the cross-validation training and test evaluations, comparatively to the unstable model performances yielded by the MLP and Decision Tree models. The final suggested attributes to use for classification are shown below:

X = [Staff, Window, Floor Space, Car Park, Competition Score, Clearance Space, Location]

y = [Performance] ∈ *{'Poor', 'Reasonable', 'Good', 'Excellent'}*

Although the models described in this report adequately perform the prediction and classification tasks from individual machine learning models, two improvements are suggested for further review:

1. There is an alternative, more appropriate, elegant solution to identifying the category performance of a shop which can be written in a small program. The program will read in the profit a given shop has made, and, based on that value will use a series of 'if' and 'else' statements to categorize the shops profit into one of four bins: 'Poor', 'Reasonable', 'Good' or 'Excellent'. This approach will give absolute certainty for classification of the profitability of a shop.
2. A more robust approach of ensemble (or majority voting), boosting or bagging methods are suggested for review for the prediction task, prior to final implementation. The ensemble approach

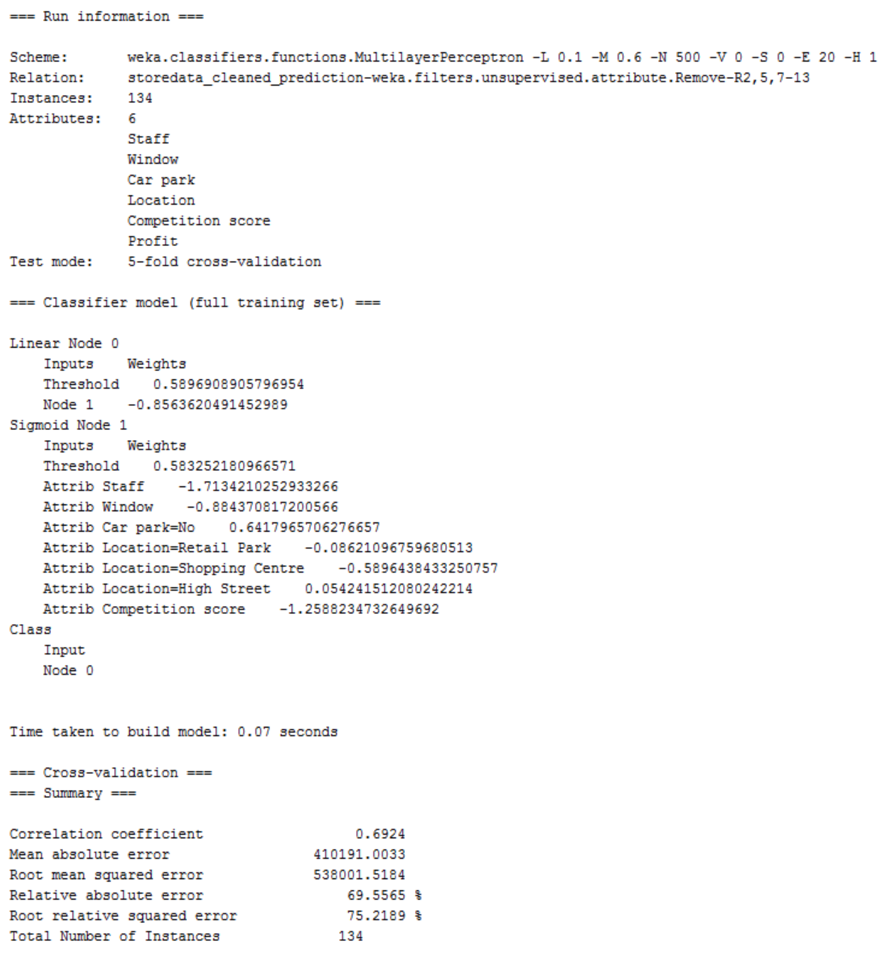
(for example) uses a set of models to each predict the profit, given the same input variable values and the average (or majority) result is taken.

Using these more sophisticated and robust approaches will deliver a better prediction or classification performance when combining individual model strengths to produce a collective, superior model performance.

In conclusion, the models have been re-trained on the selected subset of (prediction/classification) attributes, for the combined training and test data splits. The final model definitions have been included below in the *Appendix* section, for further commercial implementation.

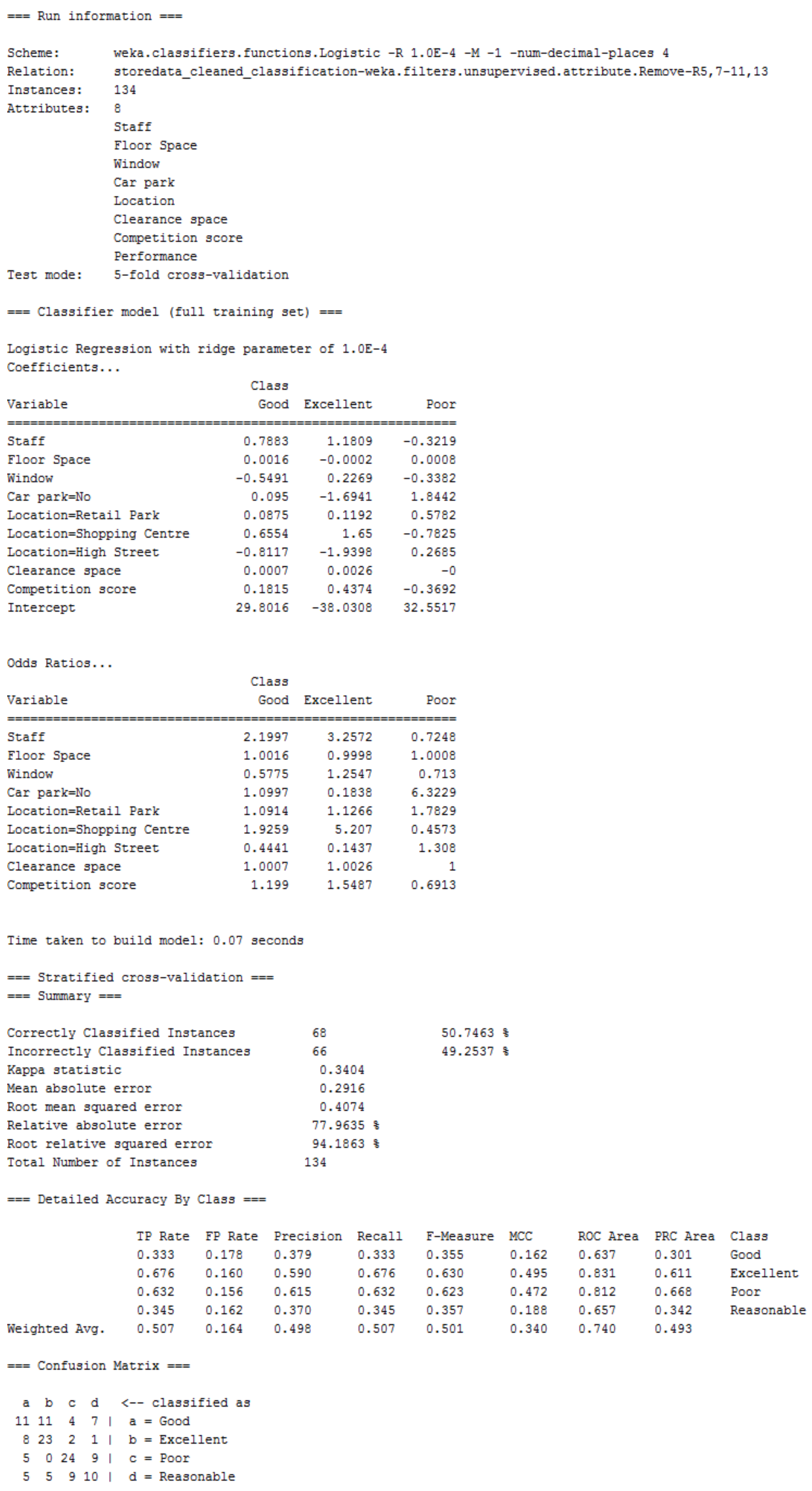
Appendix

Predictive Model



```
=== Run information ===

Scheme:       weka.classifiers.functions.MultilayerPerceptron -L 0.1 -M 0.6 -N 500 -V 0 -S 0 -E 20 -H 1
Relation:     storedata_cleaned_prediction-weka.filters.unsupervised.attribute.Remove-R2,5,7-13
Instances:    134
Attributes:   6
              Staff
              Window
              Car park
              Location
              Competition score
              Profit
Test mode:    5-fold cross-validation

=== Classifier model (full training set) ===

Linear Node 0
    Inputs    Weights
    Threshold    0.5896908905796954
    Node 1    -0.8563620491452989
Sigmoid Node 1
    Inputs    Weights
    Threshold    0.583252180966571
    Attrib Staff    -1.7134210252933266
    Attrib Window    -0.884370817200566
    Attrib Car park=No    0.6417965706276657
    Attrib Location=Retail Park    -0.08621096759680513
    Attrib Location=Shopping Centre    -0.5896438433250757
    Attrib Location=High Street    0.054241512080242214
    Attrib Competition score    -1.2588234732649692
Class
    Input
    Node 0


Time taken to build model: 0.07 seconds

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

Correlation coefficient                  0.6924
Mean absolute error                 410191.0033
Root mean squared error             538001.5184
Relative absolute error                 69.5565 %
Root relative squared error             75.2189 %
Total Number of Instances              134
```

Classification Model



```
=== Run information ===

Scheme:       weka.classifiers.functions.Logistic -R 1.0E-4 -M -1 -num-decimal-places 4
Relation:     storedata_cleaned_classification-weka.filters.unsupervised.attribute.Remove-R5,7-11,13
Instances:    134
Attributes:   8
              Staff
              Floor Space
              Window
              Car park
              Location
              Clearance space
              Competition score
              Performance
Test mode:    5-fold cross-validation

=== Classifier model (full training set) ===

Logistic Regression with ridge parameter of 1.0E-4
Coefficients...
                                Class
Variable                         Good  Excellent       Poor
===========================================================
Staff                          0.7883     1.1809    -0.3219
Floor Space                    0.0016    -0.0002     0.0008
Window                        -0.5491     0.2269    -0.3382
Car park=No                     0.095    -1.6941     1.8442
Location=Retail Park           0.0875     0.1192     0.5782
Location=Shopping Centre       0.6554       1.65    -0.7825
Location=High Street          -0.8117    -1.9398     0.2685
Clearance space                0.0007     0.0026         -0
Competition score              0.1815     0.4374    -0.3692
Intercept                     29.8016   -38.0308    32.5517


Odds Ratios...
                                Class
Variable                         Good  Excellent       Poor
===========================================================
Staff                          2.1997     3.2572     0.7248
Floor Space                    1.0016     0.9998     1.0008
Window                         0.5775     1.2547      0.713
Car park=No                    1.0997     0.1838     6.3229
Location=Retail Park           1.0914     1.1266     1.7829
Location=Shopping Centre       1.9259      5.207     0.4573
Location=High Street           0.4441     0.1437      1.308
Clearance space                1.0007     1.0026          1
Competition score               1.199     1.5487     0.6913


Time taken to build model: 0.07 seconds

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

Correctly Classified Instances          68               50.7463 %
Incorrectly Classified Instances        66               49.2537 %
Kappa statistic                          0.3404
Mean absolute error                      0.2916
Root mean squared error                  0.4074
Relative absolute error                 77.9635 %
Root relative squared error             94.1863 %
Total Number of Instances              134

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.333    0.178    0.379      0.333    0.355      0.162    0.637     0.301     Good
                 0.676    0.160    0.590      0.676    0.630      0.495    0.831     0.611     Excellent
                 0.632    0.156    0.615      0.632    0.623      0.472    0.812     0.668     Poor
                 0.345    0.162    0.370      0.345    0.357      0.188    0.657     0.342     Reasonable
Weighted Avg.    0.507    0.164    0.498      0.507    0.501      0.340    0.740     0.493

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 11 11  4  7 |  a = Good
  8 23  2  1 |  b = Excellent
  5  0 24  9 |  c = Poor
  5  5  9 10 |  d = Reasonable
```