CSCU9T6 Report

Student no: 2519302

March 16th 2018

Contents

Introduction

The task provided is to understand how one can estimate the revenue a store should produce based off data that describes aspects of each store in this chain. The dataset provided regarding all the stores in question is fairly large, as such the most appropriate course of action would be to employ data mining techniques as a method of analyzing the data provided to provide a meaningful prediction. In order to predict values, we need to find a correlation between a set of variables and the variable we intend to track, upon finding this correlation, we would need to find the function that creates this aforementioned correlation in the data. Therefore, the most efficient manner of getting this information is to use data mining techniques such as Multilayer Perceptrons or Decision Trees.

Since revenue is a value that is dependent on more than one variable, reliably predicting values for that requires finding out which of the variables provided show strong correlation to it.

This report will aim to find the most cost-effective variables that should be collected for profit prediction purposes and explain the process of finding these aforementioned variables.

Pre-processing

In order to prepare the data set for analysis by any machine learning technique, the data set needs to be pre-processed. In this process, we will be removing any outlier variables and variables which are considered flat and wide (appear sparsely) in the dataset. Since, it is not possible to gather additional statistics on these outlying entries, the only option is to remove them from the dataset as a whole since they would skew the resultant output of the models.

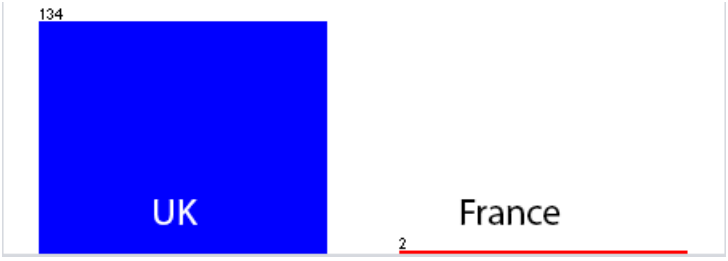


Figure 1.1 Distribution of countries

In this dataset, one of the outliers we can easily find are the entries listing France as the country where the stores are located. In this specific dataset, we are told that the chain of stores owned are in the UK, as a result, we are inclined to remove these entries since they would be considered erroneous.

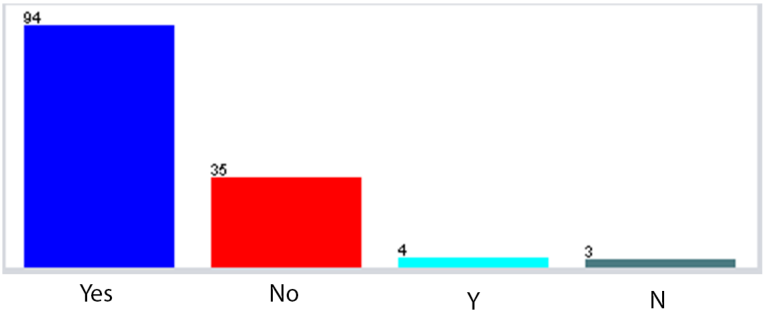


Figure 1.2 Distribution of answers to car park

If we look at the column listing whether or not there are carparks at the stores, we can see that there are some entries where, instead of a Yes or No, we have entered a Y or N. These values must be rectified to fit with the general format of the data. In this particular case, as shown above, we can see that the general format of this particular column is mostly Yes or No answers. As such, we will be changing those Y / N values to Yes / No respectively.

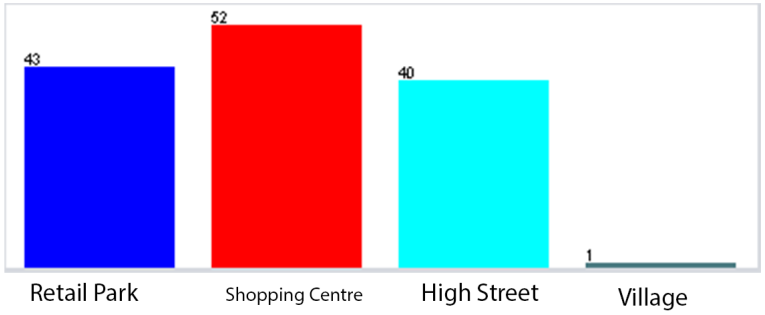


Figure 1.3 Distribution of locations as listed

Other outliers were entries listing Village as a store location. The graph above shows that this entry is listed only once, therefore, we can safely exclude this from the data set.

As for flat and wide variables, Town, Country, Store ID and Manager Name are variables that fit that classification. As we can see from the figures below, Town is unique to each location, Store ID is unique to each store and Manager Name has only a few data points where more than one person shares a name.

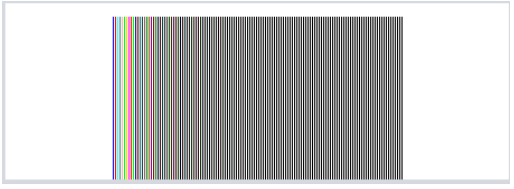


Figure 1.4 Distribution of towns as listed

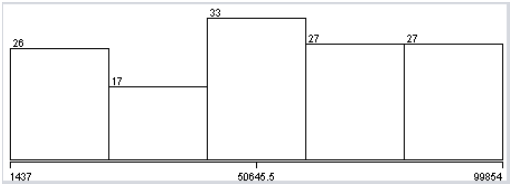


Figure 1.5 Distribution of store IDs as listed

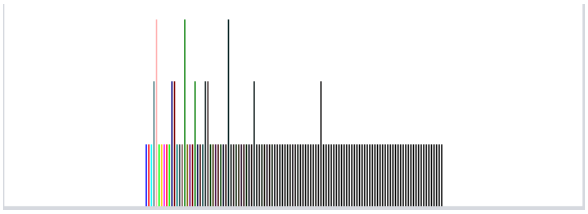


Figure 1.6 Distribution of manager names as listed

Country has only one value, UK, as can be observed from Fig 1.1, thus, we can safely exclude all the mentioned flat and wide variables from this data set.

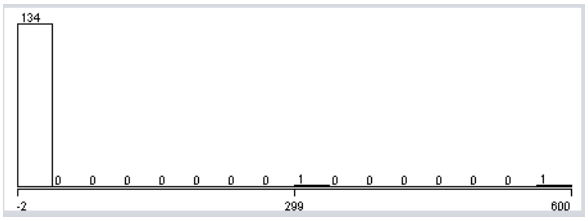


Figure 1.7 Frequency of staff headcounts as listed

Finally, entries listing their staff headcounts as -2, 300 and 600 respectively, were the last of the outliers to be found and removed from the data set. These entries are removed since, having a headcount as -2 is impossible and the other headcounts were extremely high with respect to all the other headcounts for stores listed.

Techniques

The techniques recommended to be used are the Multilayer Perceptron and the Decision Tree. Both of these methods of analysis and prediction function quite differently from each other. We shall first talk about the Multilayer Perceptron and the way it operates then we will look into the Decision Tree and how it works.

Multilayer Perceptron

A multilayer perceptron (MLP) is an artificial neural network that can find a function that finds a correlation between the inputs it receives, the output we intend to track and predict values that output can take within a certain margin of error.

The general structure of an MLP is for it to have an input layer, a hidden layer, and an output layer. Within these layers, there are neurons that compute an output using an activation function (also known as a sigmoid function) and have weighted connections to neurons in the following layer, as shown in the diagram below.

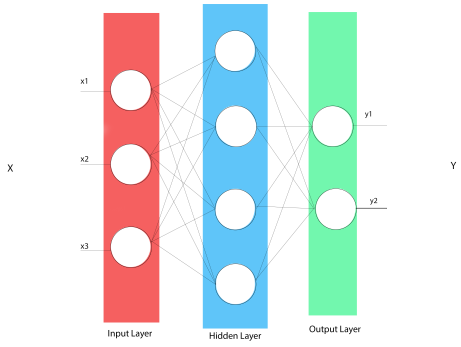


Figure 2.1 Basic structure of a MLP

The weights associated with each neuron's connection is randomly assigned at the start of the model training and is then refined by looping the training process until the error of the output from the model compared to the desired output can no longer be reduced or is at 0.

Each neuron in the MLP is composed of two parts:

1. Adder output signal / Summing function
2. Activation function (also known as the sigmoid function)

The adder output signal uses the function below to calculate the weighted sums of the inputs to a neuron:

$$O_j = f(\sum w_{ij} \cdot O_i) \tag{1}$$

Here, O_j is the weighted sum of inputs, w is the weight of the connection, i is the network input and j is the neuron in the layer.

The result of the aforementioned function is then passed to the activation function, shown below, for the neuron to process:

$$f(x) = \frac{1}{1 + e^x} \tag{2}$$

Here, x is equivalent to O_j from the adder output signal and e is Euler's number.

The training process the MLP goes through is as such:

1. Calculate the output of the neural network
2. Compare the generated output and the target output to find the training error
3. Adjust the weights of the connections based on the error value produced
4. Repeat the above steps until the error produced can no longer be reduced or is 0.

The aforementioned training process, how the model learns, is called, *error backpropagation*. This is how the model reduces the error between the output it generates to the target output.

Decision Trees

Decision trees are a classification method that mathematically decides which variables will branch off into nodes, therefore automatically classifying variables from a given data set.

In general decision trees can be made using the *divide-and-conquer* method, where a root node is chosen by calculating the information gain of the variables and picking the one with the most gain. This process is then repeated to find the leaf nodes down from the root node. This process is complete and the tree is complete when the information gain is zero or when all leaf nodes are considered pure, or have the same classification.

The specific tree used in this analysis is the **J48 Decision Tree**. J48 is an open-source

Java implementation based on the **C4.5 algorithm** which is an extension of the **ID3 Algorithm**. The primary difference between the ID3 algorithm and the C4.5 algorithm is how it deals with classifying variables after the root node is chosen. Both algorithms recursively check the remaining variables entropy to continue building the tree, however, the C4.5 algorithm is optimized to prune the tree after creation to remove any branches that do not help reduce training error and to evaluate for continuous (numerical) variables by splitting them into bins based off a threshold value.

In order to explain the formula used to find the entropy of each node, we must first look at the formula for calculating the information of an event occurring. Information, being the measure of the uncertainty of the event occurring to be specific. This is calculated with the formula:

$$I(e) = -log_2(p_e) \tag{3}$$

Therefore, the information of an event *e* is equivalent to p_e, the probability of the event *e* occurring, multiplied by the negative base two logarithm.

For example, imagine a regular deck of 52 playing cards. From these cards, we are looking for the information value of the probability of getting an ace. There are 4 aces in a deck of cards, thus the probability of drawing an ace is $\frac{1}{13}$.

Now, the information we would obtain from that by plugging in our variables to the equation, is -$\log_2(\frac{1}{13})$, which evaluates to approximately 3.7. This shows that there is little uncertainty to the event occurring. This is to show that, since it is a probability with a low chance, we can be sure that it is most likely to not occur. Contrarily, if we computed the information for picking a black card from the deck, a probability of $\frac{1}{2}$, the associated information value is evaluated as 1. This gives us little certainty on whether this event will or will not occur.

Now, **entropy**, the measure of uncertainty (or impurity), of a given variable ***X***. This is calculated using the equation:

$$H(X) = \sum P(x_i)I(x_i) = -\sum P(x_i)log_2(P(x_i)) \tag{4}$$

That is, the sum of the information of each possible event multiplied by the probability of each possible event. The value computed as a result of this equation allows us to choose the best possible variable with which to create a root node and proceed with the further building of the tree.

In order to split variables from the root node, information gain of each subsequent variable is calculated. The variable with the highest gain will be evaluated as the next leaf on the tree. The formula for evaluating information gain is as such:

$$IG(Outcome, Input) = H(Outcome) - H(Outcome|Input) \tag{5}$$

This equation describes getting the information gain *IG* evaluated as the conditional entropy of the outcome given the input subtracted from the entropy of the outcome.

Since the values we intend to predict is profit, a numerical value, the divide and conquer method will not be suitable this task. The C4.5 algorithm however, is optimized to

evaluating for continuous (numerical) variables. In this data set, we are given **Performance** which effectively classifies different profit values with a certain nominal value. This allows the C4.5 algorithm to predict the next possible profit values with a decent level of accuracy.

Process

In this section, we will discuss what variables were selected for the model training and why they were selected. This will include the variables selected for the most optimal results and also the most cost-efficient variables with the best result possible. We will also be looking at how the models were trained with the aforementioned variables.

Variable Selection

After pre-processing the data provided, the next step was to run WEKA's built-in attribute selection functions related to each model in order to start narrowing down the variables that are relevant to the task at hand. First, we will discuss narrowing down variables for the Multilayer Perceptron. Initially the Attribute Evaluator *CfsSubsetEval* was used in conjunction with the *BestFirst* search method. The output of the Attribute Evaluator with the following functions produced these results (Fig 4.1):

```
=== Attribute selection 10 fold cross-validation seed: 1 ===

number of folds (%)  attribute
          10(100 %)    1 Staff
           0(  0 %)    2 Floor Space
           8( 80 %)    3 Window
          10(100 %)    4 Car park
           5( 50 %)    5 Demographic score
          10(100 %)    6 Location
           0(  0 %)    7 40min population
           0(  0 %)    8 30 min population
           0(  0 %)    9 20 min population
           0(  0 %)   10 10 min population
           0(  0 %)   11 Store age
           2( 20 %)   12 Clearance space
          10(100 %)   13 Competition number
          10(100 %)   14 Competition score
```

Figure 3.1 Attributes generated

The results chosen were the attributes with 50% or higher folds. This excluded clearance space from the attribute list to be used.

These results were then passed through the model to see what results will be obtained. These were the generated results from the model:

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

Correlation coefficient                  0.7836
Mean absolute error                337735.0302
Root mean squared error            440353.3025
Relative absolute error                57.2307 %
Root relative squared error            61.7207 %
Total Number of Instances                130
```

Figure 3.2 Results generated with attributes from Fig 3.1

The reason behind using the aforementioned attribute evaluator and search method is due to the fact that an MLP would be looking for a subset of attributes with the most correlation to our desired output variable. This specific evaluator evaluates a subset of attributes by considering the individual predictive ability of each attribute as well as the redundancy factor between them. Subsets which show high correlation to the target attribute whilst having low intercorrelation are desired. The search method used along with the evaluator ensures that it is looking for the best attributes first, therefore providing a list of best to worst attributes with respect to the aforementioned evaluator.

Now, moving on to the selecting variables for the decision tree, as mentioned in the description for decision trees, we cannot evaluate variables to a numerical variable, therefore, we will be evaluating the data to a nominal value **performance**. Keeping this in mind, we run WEKA's attribute selector for *InfoGainAttributeEval* as its attribute evaluator and *Ranker* as its search method on the entire data set. This produces this set of variables (Fig 3.3):

```
average merit      average rank  attribute
 0.15  +- 0.009     1.1 +- 0.3    14 Competition score
0.131 +- 0.012     2.2 +- 0.75    6 Location
0.071 +- 0.014     3.3 +- 0.64    4 Car park
0     +- 0         4.7 +- 1.42    5 Demographic score
0.015 +- 0.044     4.9 +- 1.14    2 Floor Space
0     +- 0         6.4 +- 1.2    13 Competition number
0.015 +- 0.044     6.5 +- 1.2     3 Window
0     +- 0         8.6 +- 1.8     7 40min population
0     +- 0         9.4 +- 1.2     8 30 min population
0     +- 0        10.2 +- 0.6     9 20 min population
0     +- 0        10.8 +- 0.6    12 Clearance space
0     +- 0        11.3 +- 2.1    11 Store age
0     +- 0        12.8 +- 0.6    10 10 min population
0.013 +- 0.039    12.8 +- 3.6     1 Staff
```

Figure 3.3 Attributes generated

The values chosen from this result are the ones with *average merit* listed. Any variables with no merit listed will be removed from the list of variables. This narrows the variables to:

- Competition Score
- Location
- Car park
- Floor Space
- Window
- Staff

The results this subset of variables provides is as such (Fig 3.4):

```
=== Summary ===

Correctly Classified Instances          59               45.3846 %
Incorrectly Classified Instances        71               54.6154 %
Kappa statistic                          0.2669
Mean absolute error                      0.3067
Root mean squared error                  0.4717
Relative absolute error                 81.9364 %
Root relative squared error            109.0046 %
Total Number of Instances              130

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.394    0.247    0.351      0.394    0.371      0.141    0.539     0.301     Good
               0.576    0.175    0.528      0.576    0.551      0.390    0.724     0.441     Excellent
               0.528    0.202    0.500      0.528    0.514      0.320    0.683     0.393     Poor
               0.286    0.108    0.421      0.286    0.340      0.207    0.540     0.300     Reasonable
Weighted Avg.  0.454    0.186    0.452      0.454    0.450      0.268    0.626     0.362

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 13  8  8  4 |  a = Good
  7 19  5  2 |  b = Excellent
  8  4 19  5 |  c = Poor
  9  5  6  8 |  d = Reasonable
```

Figure 3.4 Results of J48 Decision Tree

Keeping in mind that each variable costs money to collect, we would like to reduce the number of variables to the minimum that would need to be collected while still having a model with results within the error margin stipulated. The first thing looked at was the common variables that were found in the both model's variable list.

This narrowed down the variables from 20 attributes to 6 attributes. These 6 attributes are as follows:

- Staff
- Window
- Car Park
- Location
- Competition Score
- Profit

Additionally, some trial and error was used with the each model's variable sets to check if there were indeed any less variables that could be used. This proved unsuccessful and all the results found were over the error margins by a significant amount.

Model Training

Given that the variables for each model has been selected and refined down to a subset common to both (with the exception of the J48 model using Performance instead of Profit). The next step is to run the variables through both models and provide results that were satisfactory to the task.

Multilayer Perceptron

Beginning with the MLP, we obtain these results when passing the variables through with no changes to the settings on the model (Fig 5.1):

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

Correlation coefficient                  0.646
Mean absolute error                473786.1946
Root mean squared error            607900.5293
Relative absolute error                80.2852 %
Root relative squared error            85.2045 %
Total Number of Instances              130
```

Figure 4.1 Initial results without any modification

Therefore, it is evident that some settings needed to be changed in order for the model to not skew the results it generated. Here, one of the main settings to be changed was whether or not the model used *decay* when in the error backpropagation step. Decay causes the learning rate to decrease. It does this by dividing the learning rate by the epoch number in order to determine what the current learning rate of the model should be. This helps keep the model from diverging from the target output (in our case, an error less than half a million pounds). Other settings changed within the model settings was the learning rate, from 0.3 to 0.22, the momentum from 0.2 to 0.001, the number of hidden layers from automatic to 4 and the training time (also known as the number of *epochs*) to 350 from 500. The changes to momentum, learning rate and decay allowed the MLP to make more gradual changes to the weights during the training process, thus reducing any potential errors gained solely from the training process. The hidden layers were changed by way of trial and error by looking at the outputs received with different numbers of hidden layers and comparing them to the output of WEKA'S automatically generated number of hidden layers as well as comparing all the outputs against each other. The best results obtained from this model with the variables reduced to the most cost efficient are as such:

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

Correlation coefficient                  0.7639
Mean absolute error                355334.5602
Root mean squared error            457330.4224
Relative absolute error                60.213  %
Root relative squared error            64.1003 %
Total Number of Instances              130
```

Figure 4.2 Multilayer Perceptron final output

We can see that the model shows clear improvement from looking at the model's respective MEAN ABSOLUTE ERROR (MAE) and ROOT MEAN SQUARED ERROR (RMSE). MAE is a measure of the average magnitude of errors in a set of predictions and RMSE is a measure of the average magnitude of errors as well, however, it gives a larger weight to large error outliers since the errors are squared before being averaged. Thus, we can observe that in *Fig 4.2* there are clearly less large errors present within the model than in *Fig 4.1*.

Decision Tree

Model training for the decision tree proceeded somewhat like the training for the MLP. Initially, passing the values through the model produced these results:

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

Correctly Classified Instances          64               49.2308 %
Incorrectly Classified Instances        66               50.7692 %
Kappa statistic                          0.3174
Mean absolute error                      0.2978
Root mean squared error                  0.4565
Relative absolute error                 79.5455 %
Root relative squared error            105.4896 %
Total Number of Instances              130

=== Detailed Accuracy By Class ===

               TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
               0.424    0.237    0.378      0.424    0.400      0.180    0.552     0.310     Good
               0.636    0.165    0.568      0.636    0.600      0.455    0.767     0.501     Excellent
               0.583    0.202    0.525      0.583    0.553      0.370    0.668     0.390     Poor
               0.286    0.078    0.500      0.286    0.364      0.259    0.597     0.364     Reasonable
Weighted Avg.  0.492    0.175    0.493      0.492    0.485      0.319    0.649     0.392

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 14  8  6  5 |  a = Good
  5 21  6  1 |  b = Excellent
  9  4 21  2 |  c = Poor
  9  4  7  8 |  d = Reasonable
```

Figure 4.3 Decision Tree initial results

Thus, certain settings in the model needed to be changed along with any variables that need not be part of the model. The first thing that had to be changed was the number of variables it used. This gave the most significant change in the model's output. This is due to the fact that tree building algorithms tend to prefer having less variables since introducing a larger data set introduces more potential errors due to the variables themselves which affects the model's capacity to efficiently classify attributes. The attribute that ended up being removed was *window*. Thus reducing the number of variables used in the decision tree to 5. This variable was removed after a series of trial and error attribute removals from the model. Following that change, changing the *confidence factor* also assisted in making the tree classify attributes better since a higher confidence factor means that the variables have to exhibit a strong correlation to their respective classification. Other than those settings and variables changing, other settings did not need to be changed.

These were the final results obtained from the decision tree that were deemed the best possible:

```
=== 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 4.4 Decision Tree final output

Model Analysis

In this section we will be taking a look at both models and analyzing their outputs, how accurate they are and where their outputs are erroneous.

Multilayer Perceptron

Firstly, looking at both MLP models, there is a clear difference between the outputs of the model. We can see this difference in the correlation coefficients of both models.

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

Correlation coefficient                  0.7836
Mean absolute error                 337735.0302
Root mean squared error             440353.3025
Relative absolute error                 57.2307 %
Root relative squared error             61.7207 %
Total Number of Instances              130
```

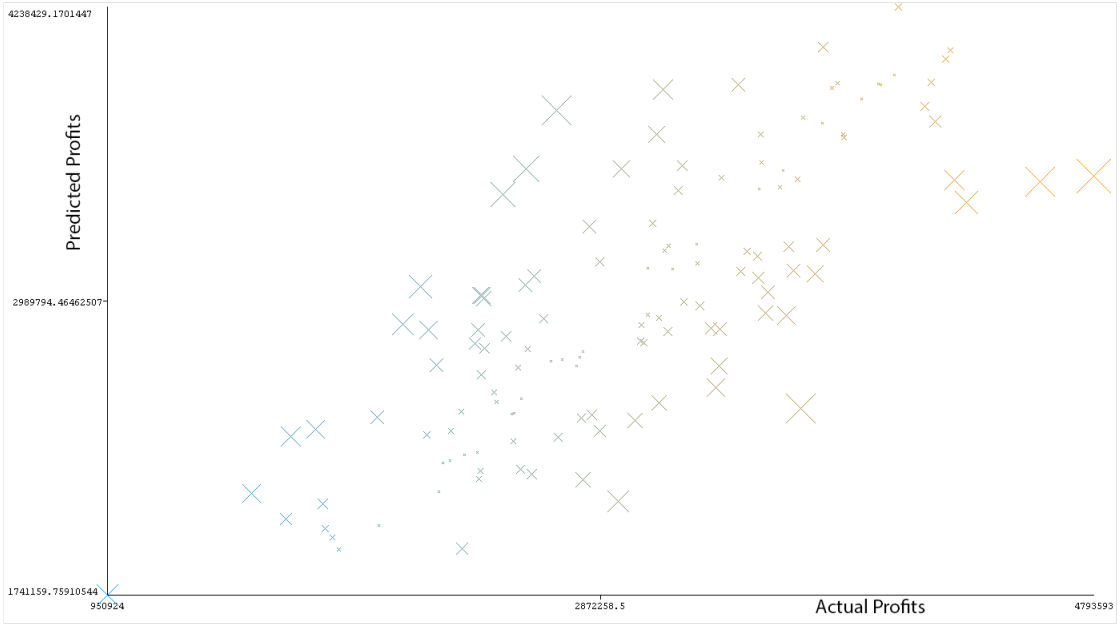
(a) MLP with best results

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

Correlation coefficient                  0.7639
Mean absolute error                 355334.5602
Root mean squared error             457330.4224
Relative absolute error                 60.213  %
Root relative squared error             64.1003 %
Total Number of Instances              130
```

(b) MLP with best result using cost efficient variables

Figure 5.1 Outputs of both models (a) and (b)

Model (a) has a better coefficient with 0.7836 to Model (b)'s coefficient of 0.7639. This is also revealed in the model's respective MAE and RMSE. Furthermore, we can see in the **classifier error charts** below that the points on the graph are more tightly clustered for model (a) with a much more visible trend towards predicted profits being near equivalent to the actual profits from each store.



(a) Classifer Error graph for **model (a)**

(b) Classifier Error graph for **model (b)**

`Figure 5.2 Classifier Errors comparison`

However, we can also see in Fig 5.2 from the classifier error graph for model (b) that, although the clustering is not as strong, it is very similar to the clustering observed in the graph for (a). However, it is clear to see that model (a) is clearly more accurate since model (b) can be observed to having more erroneous predictions than the latter. Thus, we can conclude that model (a) is superior.

Decision Tree

Now, moving on to the J48 models, we will be able to see a difference in the models by first looking at their respective outputs.

```
=== 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
```

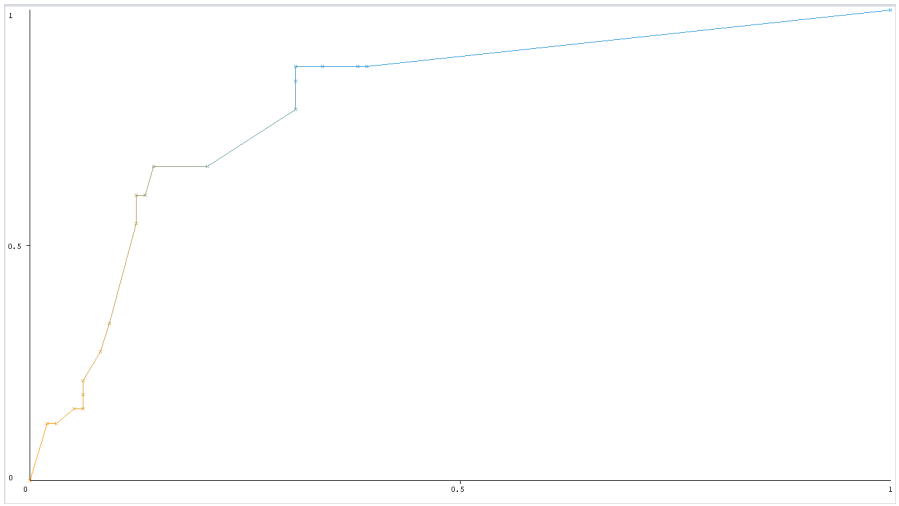
(a) J48 with best results

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

Correctly Classified Instances          64               49.2308 %
Incorrectly Classified Instances        66               50.7692 %
Kappa statistic                          0.3174
Mean absolute error                      0.2978
Root mean squared error                  0.4565
Relative absolute error                 79.5455 %
Root relative squared error            105.4896 %
Total Number of Instances              130

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.424    0.237    0.378      0.424    0.400      0.180    0.552     0.310     Good
                 0.636    0.165    0.568      0.636    0.600      0.455    0.767     0.501     Excellent
                 0.583    0.202    0.525      0.583    0.553      0.370    0.668     0.390     Poor
                 0.286    0.078    0.500      0.286    0.364      0.259    0.597     0.364     Reasonable
Weighted Avg.    0.492    0.175    0.493      0.492    0.485      0.319    0.649     0.392

=== Confusion Matrix ===

  a  b  c  d   <-- classified as
 14  8  6  5 |  a = Good
  5 21  6  1 |  b = Excellent
  9  4 21  2 |  c = Poor
  9  4  7  8 |  d = Reasonable
```

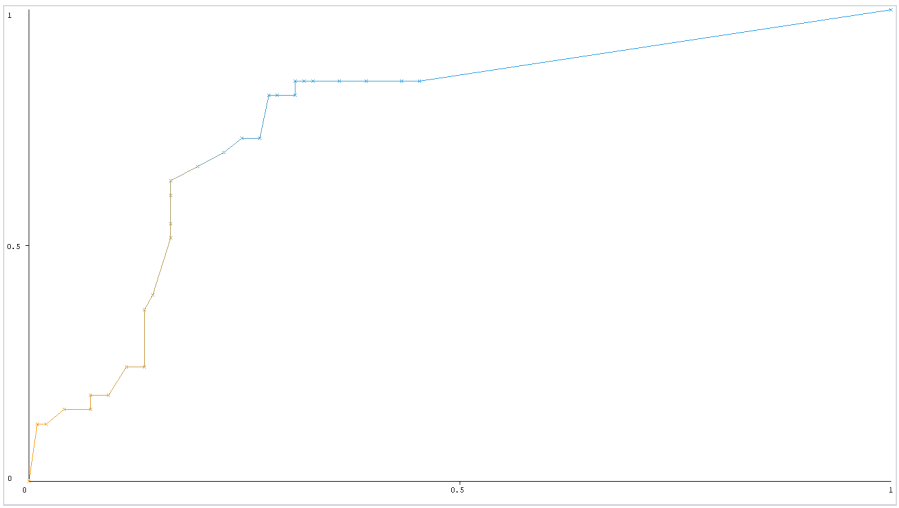
(b) J48 with initial results

We can see from the output of correctly classified instances across both models that model (a) is clearly more accurate at classifying instances than (b). Furthermore, we can look at the KAPPA STATISTIC of both model outputs along with their Receiver Operating Characteristic (ROC) curve in order to confirm which of the two models is the more accurate.

We can see that the KAPPA STATISTIC of (a) is 0.3798 compared to the 0.3174 of (b). The kappa statistic is a measure of precision. Thus we can observe that (a) is more precise than (b), albeit by a factor of 0.0624.



(a) ROC for *model (a)*



(b) ROC for *model (b)*

Looking at the ROC curves for both models also shows that (a) has a steeper graph which indicates it is the more accurate test. Along with the steepness of the graph, we can see from the area under the graph of both graphs, graph (a) has an area of 0.8015 and graph (b) has an area of 0.7674. The area under the graph is also used to measure the accuracy of the model. The closer the area is to 1, the more accurate it is, with an area of 1 being a perfect model. Thus we can conclude that, of the decision trees, model (a) is the more accurate one.

Something to note is that a kappa statistic of 0.4 indicates "fair agreement". This has an implication discussed further in the forthcoming section.

Recommended method

In this section, a recommendation of method used to analyze the data will be made and along with it a recommendation of the variables that should be collected will be provided.

As mentioned in the previous section, we can observe that even the best J48 tree did not generate results anywhere near as accurate as the results generated by any of the MLP models. Therefore, the recommended model for predicting profits for this company will be the MLP.

With that recommendation, the variables that should be collected, to generate the most accurate results while maintaining the cost efficiency of the data collection will be this subset:s

1. Staff
2. Window
3. Car Park
4. Location
5. Competition Score
6. Profit

Here, profit is included as a statistic to continue collecting since the model must have an attribute with which values are to be calculated and predicted. The population data was excluded from the final subset and the testing subset since throughout the variable selection process, it was evident that the population data showed either poor or no correlation whatsoever to the profits generated by the store.