Data Mining Assignment

dar00056

February 2020

Contents

1 Introduction

This report outlines the processes and techniques within data mining that have been used to identify patterns and company behaviours using the supplied World of Bargains dataset. The task presented is to determine which variables are useful in predicting business performance, as well as predicting profit. To achieve this, common but proven data mining techniques must be used alongside the supplied data to create different models and test several possible solutions, to finally evaluate and produce an optimal solution in terms of accuracy and validity, which will indefinitely help maximise profit, increase efficiency and keep error to a minimum.

1.1 Why Data Mining?

Data mining is a method in which we extract useful patterns from data sets to help us make predictions in the future, this is why it is ideal for this task, business performance is determined by lots of factors(features), and if identifiable relationships can give us an insight into what needs to change (even if these relationships where previously hidden) to improve performance, it helps save money through determining what data is useful, which data isn't and can help increase profit margins and forecast for the future. Furthermore, we have sufficient enough data to train and test models appropriately.

Two techniques in particular were to be used for this task as per request, a full technical description will be provided in this report [see fig 1 fig 2] later, however for a brief introduction to these, see the following:

- **Multi-Layer-Perceptron** - The MLP is a type of feed forward neural network, it is made up of several *perceptrons* and layers; the input layer, hidden layer(s) and output layer. MLP are mostly applied to supervised learning tasks, they are trained using input-output pairs and then model the dependencies between the given inputs and outputs, producing an identifiable pattern. Certain parameters can be adjusted during training to try and improve performance, this will be expanded on later.

- **Decision Tree** - A Decision Tree is a rule discovery technique that can use a divide and conquer algorithm to eventually reach a classification, this technique works best on nominal attributes. A decision tree is comprised of nodes, and these can be split into there categories: Root nodes, internal nodes and end nodes (leaf nodes). A decision tree works on a top down basis, starting at the root and eventually reaching a classification at the leaf node.

2 Variables

NOTE: All distributions shown in this section are using the performance class, as can be seen from the colouring used in the histograms.

Below are the variables supplied with the original dataset(see left), and the new subset with unused variables omitted (see Right):

Variable	Type	Value(s)
Town	Nominal	Unique
Country	Nominal	Distinct
Store ID	Nominal	Unique
Manager Name	Nominal	Unique
Staff	Numeric	Continuous
FloorSpace	Numeric	Continuous
Window	Numeric	Discrete
CarPark	Nominal	Distinct
DemScore	Numeric	Continuous
Location	Nominal	Discrete
40minPop	Numeric	Continuous
30minPop	Numeric	Continuous
20minPop	Numeric	Continuous
10minPop	Numeric	Continuous
StoreAge	Numeric	Continuous
Clearance	Numeric	Continuous
CompNum	Numeric	Continuous
CompScore	Numeric	Continuous
Profit	Numeric	Continuous
Performance	Nominal	Distinct

Town
The town name variable has been omitted as it offers no conclusive affect on the profit or performance of the company as a whole, due to it being a unique value.
Country
The country variable has been omitted as all stores are located in the UK, and it produces no conclusive affects on performance or dependencies as all stores are located in the same country. The subset of "country", contained one obvious error, as can be seen below.
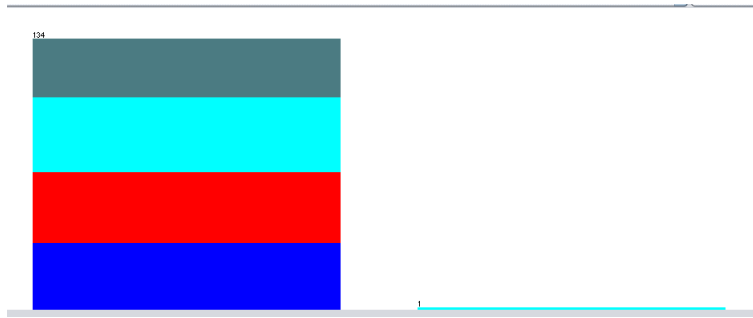
Figure 1: Country distribution with error.

Store ID

The storeID variable is a unique identifier, and an inconclusive variable when recognising patterns in data. Therefore it has been omitted.

Manager Name

The Manager Name variable is a unique identifier, and an inconclusive variable when recognising patterns in data. Therefore it has been omitted.

Staff

The variable staff is a continuous measure of how many staff are working at each store, and will be very useful when predicting performance and profit. It contains continuous numerical values, that can be compared with other variables for further insight into the performance of a business. e.g. staff numbers vs performance
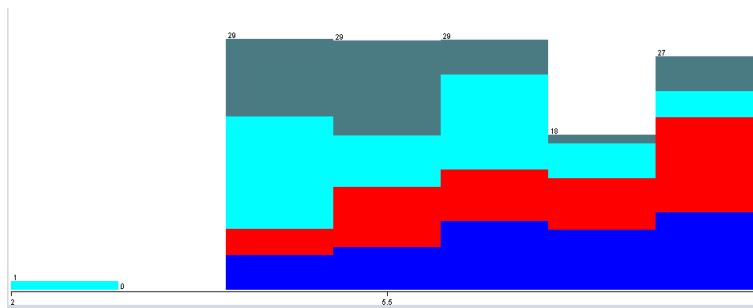

Figure 2: Staff distribution with error.

FloorSpace

The variable FloorSpace is a continuous measure of the size of a store. This could be useful when classifying performance as one could hypothesize that with greater space equals more goods = greater profit. It has been omitted from subset 3.

Window

The variable window is a continuous numerical measure of how much window space is available in stores. This has no direct affect on performance and profit and therefore has been omitted.

CarPark

The variable CarPark contains nominal discrete values of Yes or No. This variable is useful as customers who travel from longer distances are more likely to use a carpark [see below], hence bringing in a larger population range would likely increase profits.It has been omitted from subset 3.
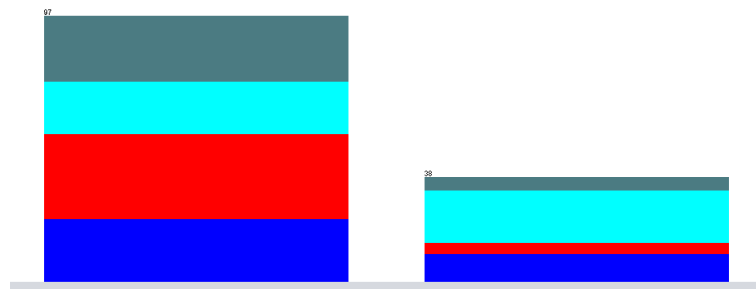

Figure 3: CarPark distribution.

DemScore

The variable DemScore contains continuous numeric values. This has been omitted in subset 1 but used in subsets 2 and 3.

Location

The variable Location contains nominal discrete values of the type of area a store is located. This is particularly useful as a difference in location could be affecting performance and profit. The subset of "location" contained an outlier, which has been omitted, as can be seen later in the data cleaning section of this report.
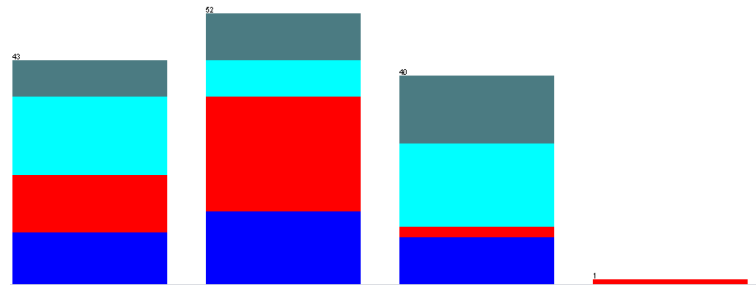

Figure 4: Location distribution with outlier.

Population

The variables 40minPop, 30minPop, 20minPop and 10minPop are continuous numeric values that measure the population of four increasing surrounding areas. This data is expensive and it is unnecessary to use all four variables here. As can be seen from the distributions below, the 10min, 20min and 30min population distributions are also unbalanced, therefore using the 40min population balanced distribution data will not only provide a larger population scope, but also gives us an even spread of population numbers in the greater surrounding area.
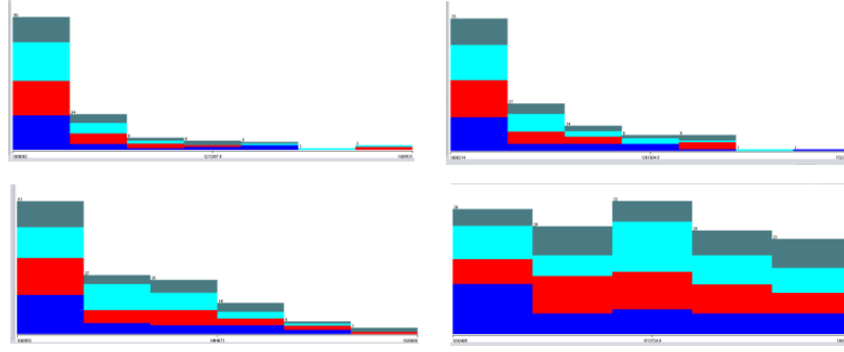

Figure 5: Top: 10min, 20min Bottom: 30min, 40min

StoreAge

The variable StoreAge contains continuous numeric values. It could have an affect on performance and profit if a store has been established for a long time. This variable has been omitted from subsets 2 and 3.
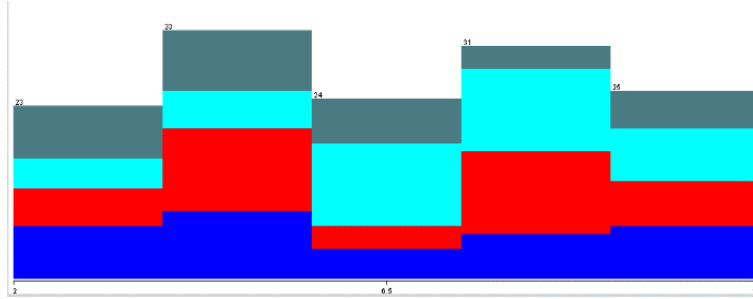

Figure 6: StoreAge Distribution

Clearance

The variable Clearance has been omitted as it is not directly related to performance or profit. The variable distribution is also unbalanced.
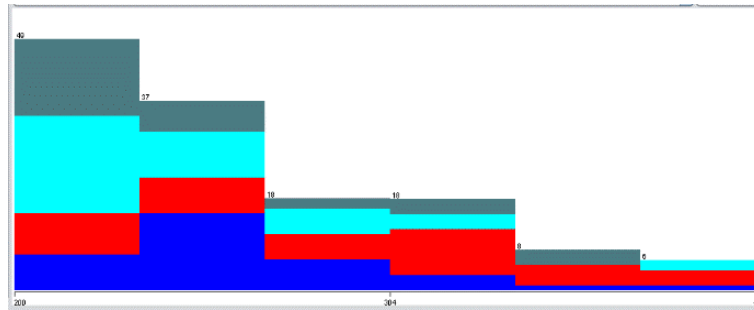

Figure 7: Clearance Distribution

CompNum

The variable "Competition Number" contains the number of competing stores around World of Bargains stores. This will be particularly useful as an investigation into whether competitor numbers are stifling sales could yield invaluable information gain. This distribution is balanced.

CompScore

The variable "Competition Score" contains the measure of performance of the competing scores in the area, this could also be useful for the same reason as outlined above. This distribution is balanced.

Profit

The variable profit will be the variable predicted by the prediction model. It is also directly linked to store performance[see below].
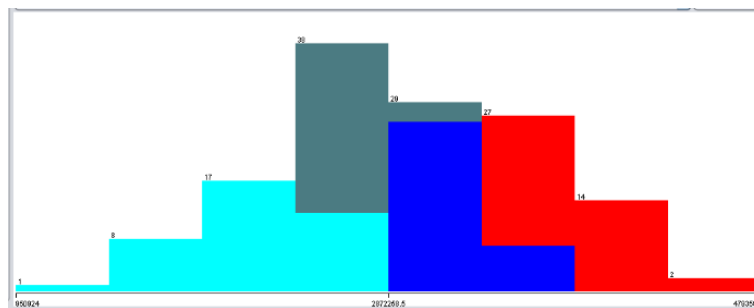

Figure 8: Profits Distribution

Performance

The variable performance is a nominally distinct variable and can be used to classify the performance of stores based on the other variables within the dataset.
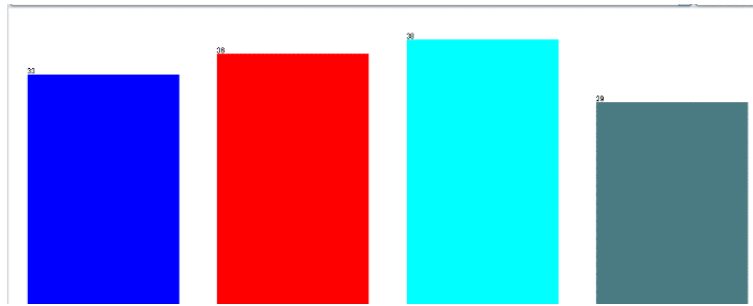

Figure 9: Performance Distribution

Subsets

I have chosen to use three different subsets in the models as to help further determine which variables are of use and which are less appropriate

Variable	Type	Value(s)	Variable	Type	Value(s)
Staff	Numeric	Continuous	Staff	Numeric	Continuous
FloorSpace	Numeric	Continuous	FloorSpace	Numeric	Continuous
CarPark	Nominal	Distinct	CarPark	Nominal	Distinct
Location	Nominal	Discrete	Location	Nominal	Discrete
40minPop	Numeric	Continuous	40minPop	Numeric	Continuous
StoreAge	Numeric	Continuous	DemScore	Numeric	Continuous
CompNum	Numeric	Continuous	CompNum	Numeric	Continuous
CompScore	Numeric	Continuous	CompScore	Numeric	Continuous
Profit	Numeric	Continuous	Profit	Numeric	Continuous
Performance	Nominal	Distinct	Performance	Nominal	Distinct

Variable	Type	Value(s)
Staff	Numeric	Continuous
Location	Nominal	Discrete
40minPop	Numeric	Continuous
DemScore	Numeric	Continuous
CompNum	Numeric	Continuous
CompScore	Numeric	Continuous
Profit	Numeric	Continuous
Performance	Nominal	Distinct

3   Methodology

The processes carried out to complete this task follows the CRISP-DM industry standard to produce the optimum and most accurate solution.
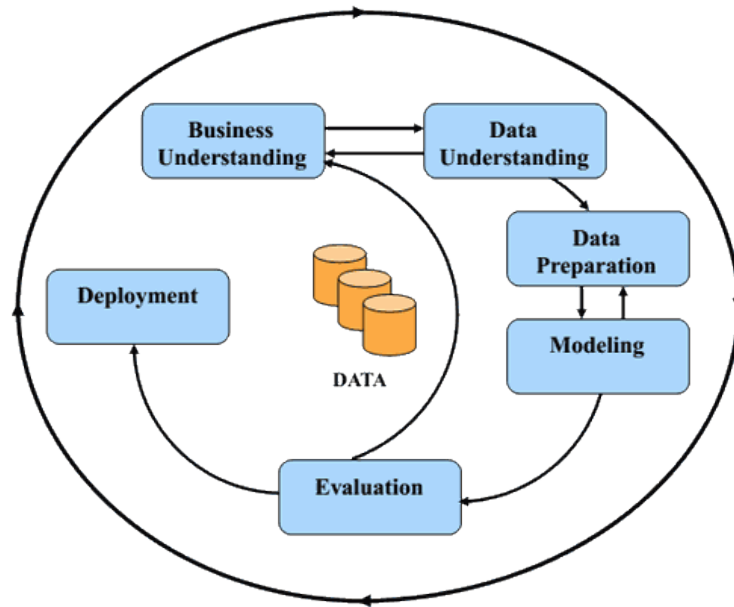

Figure 10: CRISP-DM process.

3.1 Pre-processing

These are the steps I carried out during the pre-processing stage on the data provided.

Cleaning the data

First and foremost during the pre-processing stage is to clean the data to be used in the models.

This involves:

- Removing rows with missing values or with obvious data entry errors, as to minimise inconsistency, avoid compromising the models and the reliability of the results.[see figure 1]

Figure 11: An obvious entry error (Staff)

- Recoding any obvious data entry inconsistencies, such as Y/N as opposed to Yes/No [see figure 2]

Figure 12: Y being equivalent to Yes (Car Park)

- Removing rows with minority values - Minority values are values that appear infrequently, this gives each variable a more balanced distribution and will aid in producing more accurate results. [see figure 3]

Yes	16	High Stree	1518005
Yes	17	Retail Parl	1609510
Yes	13	Village	1697206
Yes	10	Retail Parl	1486821
No	12	Retail Parl	1555172
Yes	17	Shopping	1408324

Figure 13: Highlighted row is only entry with "village"(Location)

3.2 Techniques

These are the models I used to output the predictions for the test data...

3.2.1 Multi-Layer-Perceptron(MLP)

Firstly, to help fully understand what a MLP is, a short explanation of a standard perceptron follows.
A perceptron is a logical argument comprised of four things: an input layer, weights and bias values, the net sum, and an activation function.
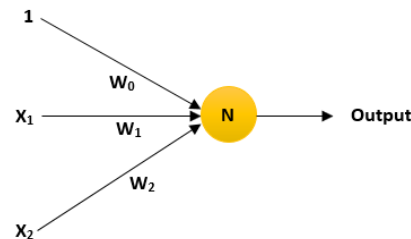
Figure 14: A standard perceptron

Given inputs x in the input layer, and the associated connection weights w:

$$w = \begin{pmatrix} w_0 \\ w_1 \\ w_2 \end{pmatrix} \qquad x = \begin{pmatrix} 1 \\ x_1 \\ x_2 \end{pmatrix}$$

Figure 15: Weights and inputs.

we can then produce the weighted sum of all inputs and associated weights:

$$\{\, w_0 + x_1 w_1 + x_2 w_2 \,\}$$

Figure 16: Weighted sum

furthermore, because the standard perceptron is a binary classifier; meaning it classifies outputs as one thing or another (e.g. 1 or 0), if the weighted sum > 0 it can be classified as 1, and if ≤ 0 the it can be classified as 0.
The activation function is a value (in binary terms, between 0 1) that acts as a threshold that must be met before the neuron is activated.

A Multi Layer Perceptron is a network of perceptrons(or linear classifiers).They are comprised of an input layer, hidden layer(s) and an output layer[see figure]. Like a standard perceptron an MLP each connection between these nodes has a weight associated with it(a number). Each of these nodes in the MLP carries out a weighted sum of its inputs and produces a result.
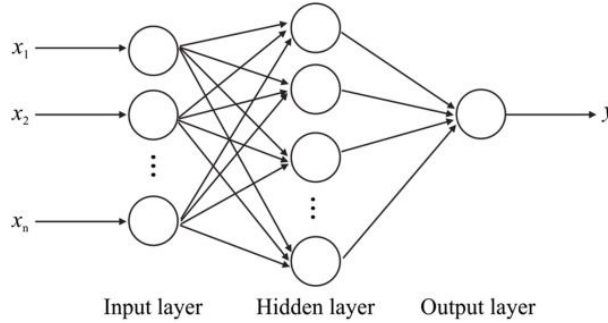

Figure 17: A three layer Multi-Layer-Perceptron

Like a standard perceptron, in a MLP a neuron is given an activation value, for simplicity in this explanation, this is a value between 0 and 1 with 0 being one classification and 1 the other, with a range in between. When these neurons take in an input, their value will correspond to the value of whatever variable value that the neuron is processing. Furthermore the activation's in the input layer, determine the activation's in the next layer.[4] Like a standard perceptron, the connections between these layers are called weights, to refresh; weights are numbers which can be used to tweak and change how the neural network behaves, for example, we can assign a positive weight to neuron connections with an activation value closer to 1, and a negative weight to an activation value closer to 0, with the weights differing depending where they correspond to each neurons activation value on the scale between 0 and 1. Therefore the final activation function of each neuron is how positive the sum of the relative weighted value is. This calculation is carried out on every neuron in the network, which gives us the value in which the neurons will activate. This is called the activation function, like in a standard perceptron the product of each neurons weight and activation is taken, and then added together in series[10]:

$$(a_1 w_1 + a_2 w_2 + ... a_n w_n)$$

However, when calculating a weighted sum, we want to be able to condense it down into our range of 0 to 1, this is achieved using the Sigmoid function.

In a Sigmoid function, negative inputs are closer to 0, and more positive inputs are closer to 1, with a steady positive increase around the input of 0:
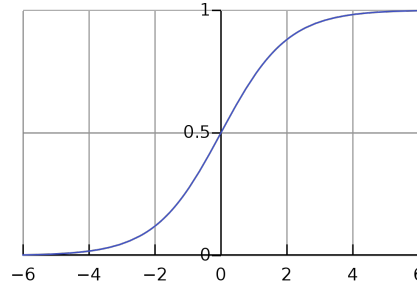
$$\sigma = \frac{1}{1 + e^{-}(x)}$$

Figure 18: Sigmoid equation and its graphical form.[C]

We can use this function to determine that the activation value is just a measure of how positive the relevant weighted sum is, which when coupled with the relevant series of weight and activation products, produces:

$$\sigma(a_1 w_1 + a_2 w_2 + ...a_n w_n)$$

However for the purpose of tuning a neural network, one may not want a neuron to activate if its value is only bigger than 0, to change this, we use a Bias, which can be any given value, to amend the function and cause the activation to occur when the result is larger(or smaller) than the new Bias inherent value:

$$\sigma((a_1 w_1 + a_2 w_2 + ...a_n w_n + B)$$

This expression can also be written as a set of vectors and a matrix,however in this example it shall be condensed down, for ease of use when applying it to real world applications:

$$A^1 = \sigma(W a^0 + B)$$

Therefore, in conclusion, the weights of each connection between neurons indicate which variable value a given neuron is processing, and the Bias gives the principle value that the neuron is required to reach before it activates. Each connection between neurons, has its own weight associated with it, and each neuron has its own Bias.[4][10] Both of these variables are *hyperparameters*, large sets of parameters that if organised a certain way yield the best results, there is no formula to achieve this ideal state. When training a neural network, its "learning" is essentially the process of finding the best weights and Bias that correspond with a higher accuracy rate.

3.2.2 Decision Tree

3.3 Other Techniques

3.4 Optimal solution

Here is the best performing process that I carried out on the data....

3.4.1 Cross-validation

Cross validation allows us to compare different models to develop an understanding of how they will work in practice, and reduces to risk of just outputting a "lucky" test.

4 Results Analysis

An analysis of the best results that I could find..

5 Recommendations

I can recommend that these changes be incorporated in the future...