**Predictive Modelling for   
ATM Cash Demand**

Rowena Kok 440256380

Charles Christopher Hyland 450411920

# Exploratory data analysis

Our analysis and modelling has been undertaken on a dataset comprising 22,000 records with the following attributes:

**Table 1**: Description of variables

|  |  |
| --- | --- |
| Variable | Description |
| *Withdraw (response)* | The total cash withdrawn in a day (1000s of local currency) |
| *Shops* | Number of shops/restaurants within a walkable distance (100s) |
| *ATMs* | Number of other ATMs within a walkable distance (10s) |
| *Downtown* | 1 if the ATM is in downtown, 0 if not |
| *Weekday* | 1 if the day is a weekday, 0 if not |
| *Center* | 1 of the ATM is located in a center (shopping, airport, etc.), 0 if not |
| *High* | 1 if the ATM has a high cash demand in the last month, 0 if not |

The data was validated to be complete and with no missing values. An initial exploration of descriptive statistics revealed the following information for the continuous variables Withdraw, Shops and ATMS, and the binary variables Downtown, Weekday, Center and High:

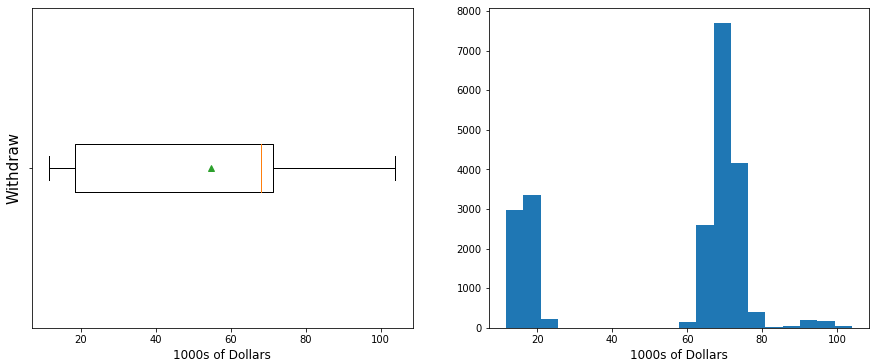
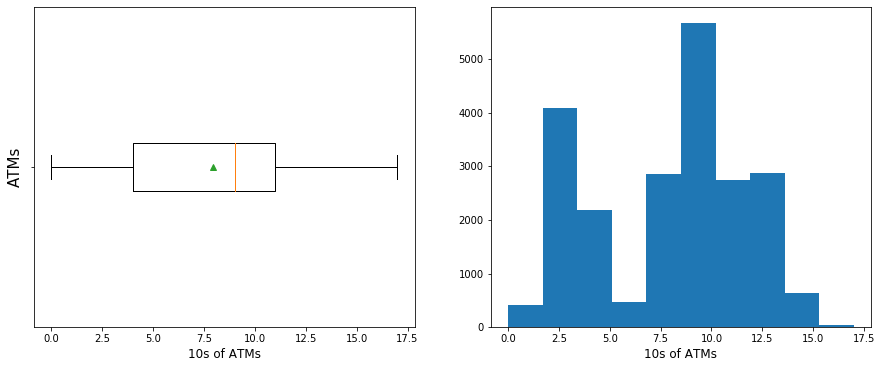
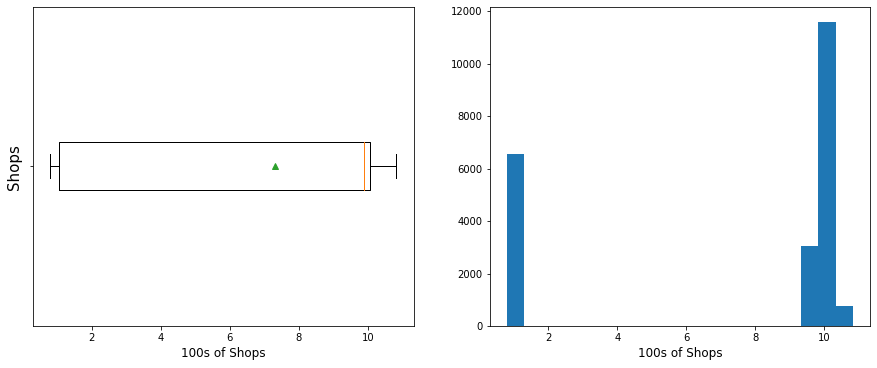
**Table 2**: Descriptive statistics of continuous variables

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Continuous Variable | Mean | Standard deviation | Median | Skew |
| *Withdraw* | 54.653 | 25.100 | 68.241 | -0.772 |
| *Shops* | 7.316 | 4.119 | 9.890 | -0.878 |
| *ATMs* | 7.937 | 3.673 | 9 | -0.319 |

**Table 3**: Descriptive statistics of binary variables

|  |  |  |
| --- | --- | --- |
| Binary Variable | Frequency of 1 (%) | Frequency of 0 (%) |
| *Downtown* | 6556 (29.8%) | 15444 (70.2%) |
| *Weekday* | 6290 (28.6%) | 15710 (71.4%) |
| *Center* | 19746 (89.9%) | 2254 (10.2%) |
| *High* | 15365 (69.8%) | 6635 (30.2%) |

This demonstrated that all continuous variables were quite left-skewed. Further analysis of the continuous variables (see Figure 1 below) revealed the distribution of the data and the need for standardisation.

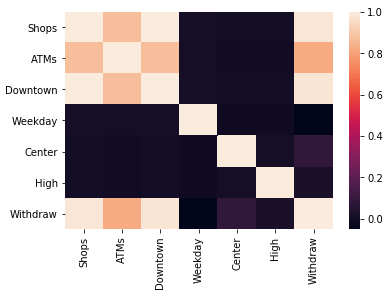


**Figure 1**: Boxplots and histograms of continuous variables

The following correlation matrix was computed:

**Table 4**: Correlation matrix of variables

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Shops | ATMs | Downtown | Weekday | Center | High | Withdraw |
| Shops | 1 | 0.873 | 0.999 | 0.013 | 0 | 0.002 | 0.986 |
| ATMs | 0.873 | 1 | 0.874 | 0.01 | -0.003 | -0.003 | 0.824 |
| Downtown | 0.999 | 0.874 | 1 | 0.013 | 0 | 0.002 | 0.984 |
| Weekday | 0.013 | 0.01 | 0.013 | 1 | -0.007 | -0.007 | -0.05 |
| Center | 0 | -0.003 | 0 | -0.007 | 1 | 0.011 | 0.088 |
| High | 0.002 | -0.003 | 0.002 | -0.007 | 0.011 | 1 | 0.021 |
| Withdraw | 0.986 | 0.824 | 0.984 | -0.05 | 0.088 | 0.021 | 1 |



**Figure 2**: Heatmap of correlation matrix of variables

Although the Shops and Downtown variables are highly correlated, they were both included in the analysis as they were both considered to feasibly influence the response Withdraw variable. However, we made particular note to include models that would account for any potential multicollinearity, such as Ridge regression and Neural Networks.

# Data pre-processing

The training dataset of 22,000 entries was split into 60% (13,200) training set, and 40% (8800) validation set (full code can be found in Analysis.ipynb).

In preparation for training the model, all variables were standardised by removing the mean and scaling to unit variance (using scikit-learn’s StandardScaler), and the response variables were separated from the predictors.

Since we are using the *keras* module for to implement our neural networks, our *pandas* DataFrames needed to be transformed in matrices. Additionally, when training the neural network model, as many of our predictors are dummy variables the presence of the zeros (0s) would result in the model not updating. Thus, to ensure the parameters are updated, a small constant (0.11) was added to all predictors in both the training and validation set.

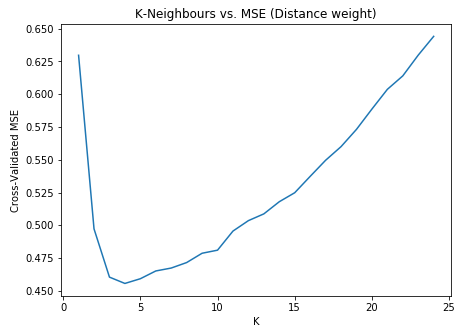
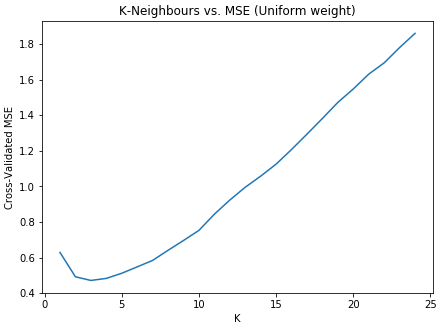
# Model selection

## K-nearest neighbours

The k-Nearest Neighbours (k-NN) method predicts new data by computing the simple average of the *k* most similar observations in the training sample. In order to choose the hyperparameter *k*, we utilise a randomised search on cross validation (*RandomizedSearchCV* from the scikit-learn package).

We evaluated the performance of a multitude of k-NN models with k ranging from 1 to 25 on the validation data using two weighting functions – ‘*uniform’*, where all points in each neighbourhood are weighted equally, and ‘*distance’*, where points are weighted by the inverse of their distance, i.e. closer neighbours have a greater influence than neighbours further away. The metric used was negative mean squared error (MSE).

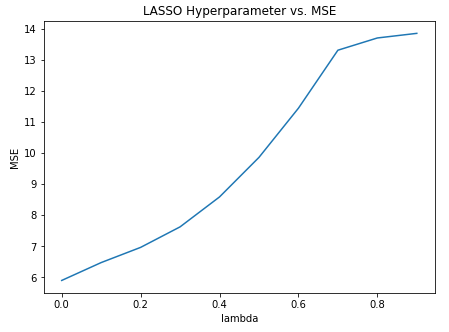
The results for this are plotted on the graphs below – the optimal k-NN model was found to be that with *k* = 9, using the distance weighting function, and with a cross-validated MSE of 0.479.



**Figures 3 and 4**: k-Neighbours vs. MSE (uniform and distance weightings)

## Lasso

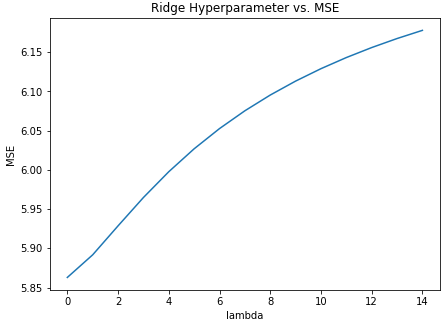
The LASSO method imposes a penalty on the norm of the parameter vector. The LASSO estimator minimises the following penalised objective function:

The hyperparameter controls the shrinkage and must be selected. The figure below graphs the chosen against the MSE, and demonstrates that zero shrinkage was the optimal LASSO model, with an MSE of 5.9. Since this is much higher than that of the optimal k-NN model, it was excluded from our consideration in further modelling.

**Figure 5**: LASSO hyperparameter tuning

## Ridge

The Ridge regression method imposes a penalty on the norm of the parameter vector. The Ridge regression estimator minimises the penalised residual sum of squares:

Again, we must select the appropriate . The figure below illustrates the results of examining a range of from 0 to 15. Again, we see that is the minimal MSE Ridge regression model, with MSE = 5.9, so the Ridge regression method was also excluded from our further modelling.

**Figure 6**: Ridge regression hyperparameter tuning

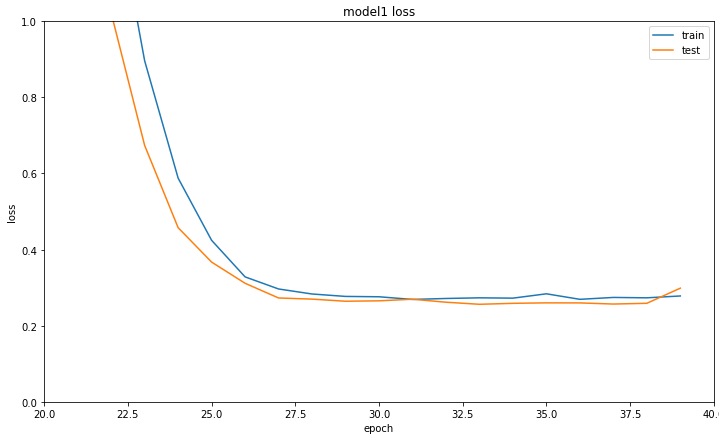
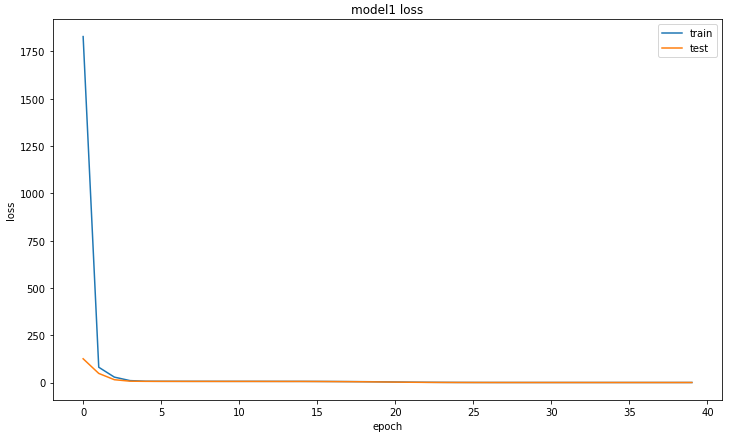
## Neural networks

Neural networks are a data representation learning technique that utilise a layered composite function to develop very flexible, non-linear methods for regression and classification.

We examined the following structures of neural networks:

### Initial sequential model

A sequential model using the arguments below was compiled using the adaptive movement (ADAM) optimiser. ADAM was utilised as it is computationally efficient, has been shown to work well in practice and outperforms other stochastic optimisation methods (Kingma and Ba, 2014).

**Table 5**: Initial sequential neural network arguments

|  |  |  |
| --- | --- | --- |
| Layer no | Layer Type | Specifications |
| 1 | Input layer | Input = 6, activation = relu |
| 2 | Dense layer | Input = 24, output = 24, activation = relu |
| 3 | Dense layer | Input = 24, output = 12, activation = relu |
| 4 | Dense layer | Input = 12, output = 1, activation = relu |

**Figures 7 and 8**: MSE vs. Epoch for Initial Sequential model (batch size = 60)

A batch size of 60 was considered appropriate, and the MSE of the model was plotted against the number of epochs from a range of 0 to 40. We can see in Figure 8 (focusing on models with an MSE less than 1) that we could consider early stopping at a number of epochs = 30. Using number of epochs = 30, and batch size = 60 resulted in an MSE of 0.2932.

Thus, using the number of epochs = 40, and batch size = 60, this resulted model in an MSE of 0.2539.

### Wide model

A wide model was also examined using the following arguments:

**Table 6**: Wide neural network arguments

|  |  |  |
| --- | --- | --- |
| Layer no | Layer Type | Specifications |
| 1 | Input layer | Input = 6, activation = relu |
| 2 | Dense layer | Input = 6, output = 10000, activation = relu |
| 3 | Dense layer | Input = 10000, output = 1000, activation = relu |
| 4 | Dense layer | Input = 1000, output = 250, activation = relu |
| 5 | Dense layer | Input = 250, output = 1, activation = relu |

With number of epochs = 40 and batch size = 60, this resulted in an MSE of 0.4710.

### Dropout model

A sequential model utilising dropout regularisation as in the arguments below was compiled using the adaptive movement (ADAM) optimiser.

**Table 7**: Dropout neural network arguments

|  |  |  |
| --- | --- | --- |
| Layer no | Layer Type | Specifications |
| 1 | Input layer | Input = 6, activation = relu |
| 2 | Dense layer | Input = 6, output = 30, activation = relu, dropout ratio = 0.1 |
| 2 | Dense layer | Input = 30, output = 30, activation = relu, dropout ratio = 0.1 |
| 3 | Dense layer | Input = 30, output = 30, activation = relu |
| 4 | Dense layer | Input = 30, output = 1, activation = linear |

With number of epochs = 40 and batch size = 60, this resulted in an MSE of 1.1286, indicating that dropout regularisation was not appropriate for this data, so it was excluded from further analysis.

### Rectangular model

A sequential rectangular model using the arguments below was compiled using the adaptive movement (ADAM) optimiser.

**Table 8**: Rectangular neural network arguments

|  |  |  |
| --- | --- | --- |
| Layer no | Layer Type | Specifications |
| 1 | Input layer | Input = 6, activation = relu |
| 2 | Dense layer | Input = 6, output = 20, activation = relu |
| 3 | Dense layer | Input = 20, output = 20, activation = relu |
| 4 | Dense layer | Input = 20, output = 20, activation = relu |
| 5 | Dense layer | Input = 20, output = 20, activation = relu |
| 6 | Dense layer | Input = 20, output = 1, activation = relu |

With number of epochs = 40 and batch size = 60, this resulted in an MSE of 0.2593.

### Right based-pyramid model

A sequential right based-pyramid model using the arguments below was compiled using the adaptive movement (ADAM) optimiser.

**Table 9**: Right based-pyramid neural network arguments

|  |  |  |
| --- | --- | --- |
| Layer no | Layer Type | Specifications |
| 1 | Input layer | Input = 6, activation = relu |
| 2 | Dense layer | Input = 6, output = 10, activation = relu |
| 3 | Dense layer | Input = 10, output = 50, activation = relu |
| 4 | Dense layer | Input = 50, output = 500, activation = relu |
| 5 | Dense layer | Input = 500, output = 1000, activation = relu |
| 6 | Dense layer | Input = 1000, output = 1, activation = relu |

With number of epochs = 40 and batch size = 60, this resulted in an MSE of 0.2852.

### Left-based pyramid

A sequential left based-pyramid model using the arguments below was compiled using the adaptive movement (ADAM) optimiser.

**Table 10**: Right based-pyramid neural network arguments

|  |  |  |
| --- | --- | --- |
| Layer no | Layer Type | Specifications |
| 1 | Input layer | Input = 6, activation = relu |
| 2 | Dense layer | Input = 6, output = 1000, activation = relu |
| 3 | Dense layer | Input = 1000, output = 500, activation = relu |
| 4 | Dense layer | Input = 500, output = 50, activation = relu |
| 5 | Dense layer | Input = 50, output = 10, activation = relu |
| 6 | Dense layer | Input = 10, output = 1, activation = relu |

With number of epochs = 40 and batch size = 60, this resulted in an MSE of 0.3466.

### Final model/summary

**Table 11**: Neural network model MSE summary

|  |  |
| --- | --- |
| Neural Network | MSE |
| Initial sequential | **0.2539** |
| Wide |  |
| Dropout | 1.1286 |
| Rectangular | 0.2593 |
| Right-based pyramid | 0.2852 |
| Left-based pyramid | 0.3466 |

# References

Kingma, D. and Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.