# QBUS3820 Group Assignment: Building a predictive model for ATM withdrawal amounts


## Summary

Optimizing cash management and efficiently maintaining and reloading ATM networks to best serve customers is an important task for banks. The goal of this analysis is to build a predictive model for total daily ATM cash withdrawals. Exploratory data analysis was conducted to better understand the distribution of the data, relationships between variables, and to determine important features that were used for both parametric and nonparametric approaches. Emphasis is given to the predictive accuracy of the model, and the metric used for evaluating this is the mean squared error (MSE) over the partitioned test data. The analysis narrowed down to two very different models: a neural network and a linear regression with various interaction terms. The simpler linear regression model was slightly favored over the neural network because the errors were about the same, but the linear regression with interactions was overall simpler and enough to capture many of the nonlinear relationships in the data.


## Data Understanding

The data used for this analysis is summarized in **Table 1**.

| Variable | Description |
|---|---|
| Shops | Number of shops/restaurants within walkable distance (in 100s) |
| ATMs | Number of other ATMs within a walkable distance (in 10s) |
| Downtown | If the ATM is downtown (1 = Yes, 0 = No) |
| Weekday | If the day is a weekday (1 = Yes, 0 = No) |
| Center | If the ATM is located in a center (shopping, airport, etc.) (1 = Yes, 0 = No) |
| High | If the ATM had a high cash demand in the last month (1 = Yes, 0 = No) |
| Withdraw | Total cash withdrawn a day (in 1000s) |

**Table 1. Description of dataset**

The first thing to check is the completeness of the data, that is there are no missing or unexpected values. A simple analysis of the covariates reveals that out of the 22,000 data points, there are no missing values and the variables all have sensible inputs (ie the binary

variables only contain 0 and 1, and the Shops and ATMs variables all have non-negative values).

The next check is to identify outliers. The presence of outliers can have a detrimental effect on the model since it could cause overfitting as the model tries to adjust for the outliers. This would then lead to lower prediction accuracy. For this analysis, an outlier is simply defined as any value outside 3 standard deviations of the mean. Some key metrics of the data to look at outliers are listed in **Table 2**. Only the variables *Shops*, *ATMs* and *Withdraw* are examined as the rest are binary.

| Variable | Mean | St. Dev. | Min | Max | Lower Bound | Upper Bound |
|---|---|---|---|---|---|---|
| Shops | 7.316 | 4.119 | 0.800 | 10.830 | -5.040 | 19.672 |
| ATMs | 7.937 | 3.673 | 0 | 17 | -3.083 | 18.958 |
| Withdraw | 54.653 | 25.010 | 11.668 | 103.964 | -20.646 | 129.952 |

**Table 2. Setting outlier boundaries**

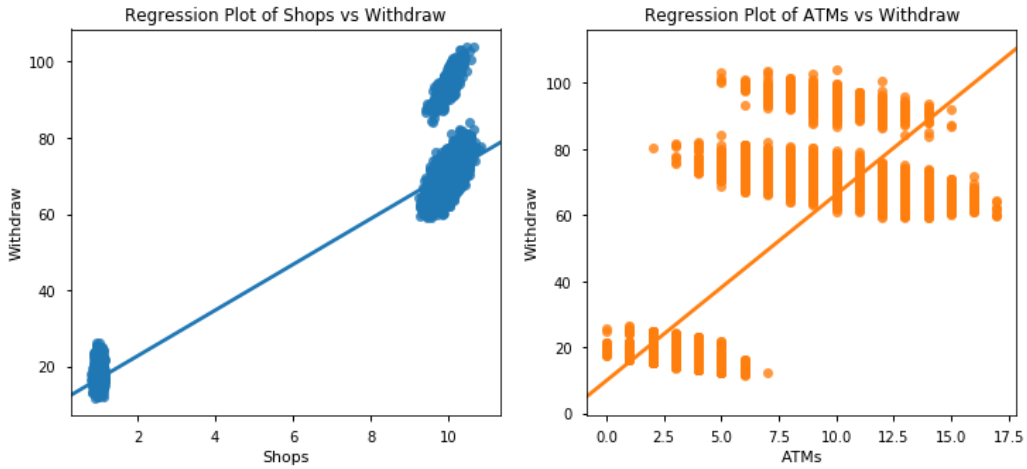
By this definition, all the data points are well inside the outlier boundaries. It is a point to note the standard deviations are so high relative to the means for each of the variables, which is why the outlier boundaries are as extreme as they are. This has potential to be an issue as the high standard deviations could be a direct result of the outliers themselves, which would warrant different methods of outlier detection. This could also be because many of the points are pocketed or grouped in the upper and lower ranges of the data, so for now all the values are left in to be further examined the the EDA phase.

# 3 Exploratory Data Analysis

Exploratory data analysis is conducted to develop a better understanding of the structure and distribution of the data. The goal is to identify any patterns or characteristics that could aide in the modelling process.

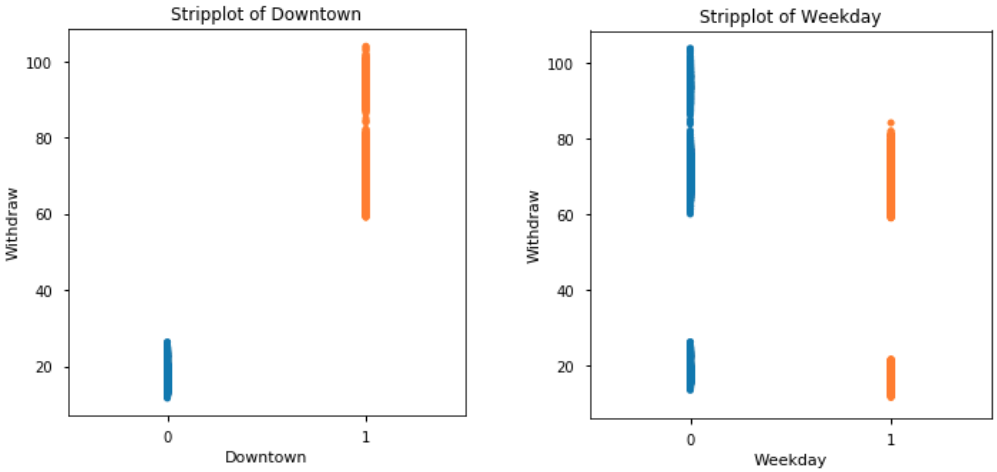
Scatterplots of *Shops* and *ATMs* against the response variable in **Plots 1 and 2** show the relationships between these variables and *Withdraw* is highly non-linear. *Shops* in particular shows a clear divide in the distribution of those data points, where every observation has either less than 200, or more than 800 shops/restaurants, but nothing in between. The *Shops* plot also clearly shows that overall, a higher number of shops/restaurants within walking distance corresponds to higher withdrawal amounts, which makes intuitive sense. For ATMs it is a bit more nuanced. A higher number of ATMs nearby does not necessarily correspond to higher withdrawal amounts, though there is a point where after about 7 ATMs, the withdrawal amounts are all around 60 thousand and above, hence the generally upward trend. It is also clear that within the three distinct clusters, there is a slight downward trend as the number of ATMs increase, which is also to be expected. These observations provide a rationale for creating and

trying to implement additional terms such as an interaction between number of ATMs nearby and the location.

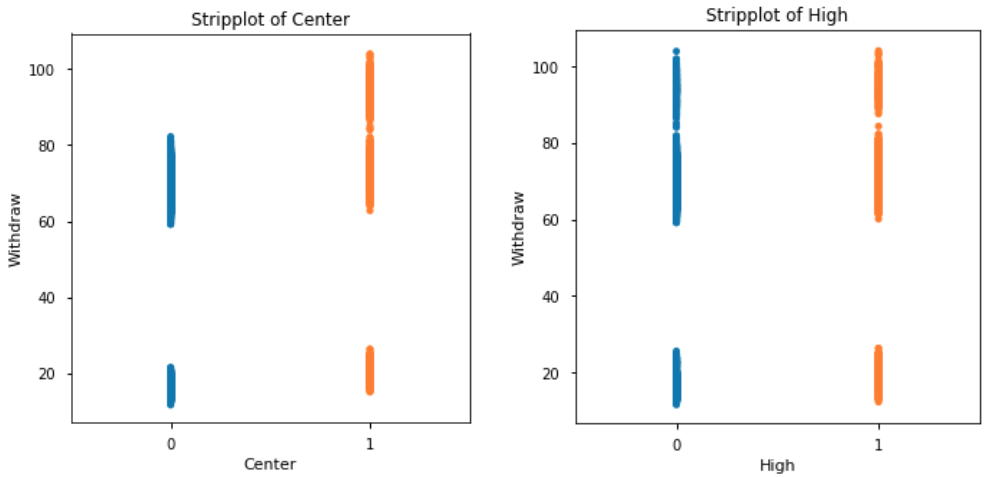



**Plots 1 and 2. *Shops* and *ATMs* relationship with *Withdraw***

Stripplots are drawn to visualize the relationships between the binary variables and the daily withdrawal amounts. These are shown in **Plots 3 - 6**.




**Plots 3 and 4. Stripplots of *Downtown* and *Weekday***




**Plots 5 and 6. Stripplots of *Center* and *High***

While the stripplots help to understand the distribution and how the binary classes might influence withdrawal amount, it is hard to extract any information on the category frequencies, so a breakdown of the proportion of ATMs that are 0 and 1 for each binary variable is in **Table 3**.

| Variable | Proportion = 1 | Proportion = 0 |
|---|---|---|
| Downtown | .702 | .298 |
| Weekday | .714 | .286 |
| Center | .102 | .898 |
| High | .302 | .698 |

**Table 3. Proportion breakdown of binary classes**

There's a few things to note from **Table 3 and Plots 3 - 6.** First is the only variable that looks to have some obvious relationship with withdrawal amount is whether or not the ATM is downtown. ATMs that are downtown on average have much more cash withdrawn on a daily basis than those further out. This makes sense as there are far more shops/restaurants in a downtown area and therefore more people in need of cash. *Center* and *Weekday* together show that the highest cash withdrawals, over about 82 thousand in a day, happen exclusively on the weekend (Weekday = 0) in centers (Center = 1), and that the majority of the data points for those two variables lie in the opposite category. This could indicate an interaction between the two variables would benefit the model because they together have an effect on the withdrawal amount. The last binary variable, *High*, doesn't look to have any noteworthy relationship with *Withdraw*.

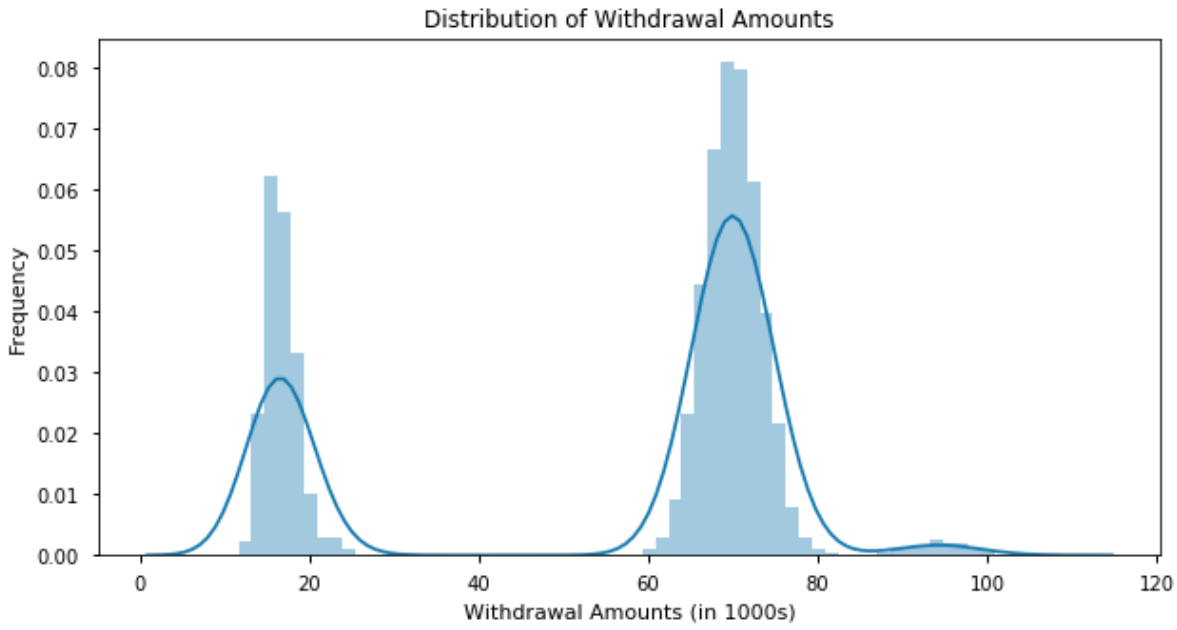
**Table 4** shows the correlations between the variables. Note for the binary variables, the point-biserial correlation is being used.

| *Variable* | **Shops** | **ATMs** | **Downtown** | **Weekday** | **Center** | **High** | **Withdraw** |
|---|---|---|---|---|---|---|---|
| **Shops** | 1.000 | 0.873 | 0.999 | 0.013 | 0.000 | 0.002 | 0.986 |
| **ATMs** | 0.873 | 1.000 | 0.874 | 0.010 | -0.003 | -0.003 | 0.824 |
| **Downtown** | 0.999 | 0.874 | 1.000 | 0.013 | -0.000 | 0.002 | 0.984 |
| **Weekday** | 0.013 | 0.010 | 0.013 | 1.000 | -0.007 | -0.007 | -0.050 |
| **Center** | 0.000 | -0.003 | -0.000 | -0.007 | 1.000 | 0.011 | 0.088 |
| **High** | 0.002 | -0.003 | 0.002 | -0.007 | 0.011 | 1.000 | 0.021 |
| **Withdraw** | 0.986 | 0.824 | 0.984 | -0.050 | 0.088 | 0.021 | 1.000 |

**Table 4. Correlations among variables**

One thing to notice is the near perfect multicollinearity between *Downtown* and *Shops.* Further inspection of the this reveals there are 0 records in the data where the number of nearby shops/restaurants is less than 200, and the ATM is also downtown. It can be concluded that all the information in *Downtown* is already being captured in *Shops.* It also has a very high correlation with *ATMs* and *Withdraw.* While having a high correlation with the response is typically good, the high multicollinearity is cause for concern depending on the model. For example, a neural network model is far less sensitive to multicollinearity than OLS. It could simply be removed entirely, however, given that the emphasis is on predictive accuracy, if using *Downtown* legitimately improves a model then there is consideration to keep it in.

EDA is concluded with looking at the distribution of the response variable, *Withdraw,* in **Plot 7**.




**Plot 7. Distribution of Withdrawal Amounts**

This provides a clearer picture of the distribution, though it can be inferred from previous plots that show a large gap between about 20 and 60. This corresponds with the large gap in number of walking distance shops/restaurants shown in **Plot 1**, but the reason for this gap is unclear and would require more knowledge of the bank and locations. As a final observation, the two main clusters seem to be pretty normally distributed, though this doesn't have any clear implications for modelling.

# 4 Modelling

A few models, both parametric and nonparametric, are estimated and evaluated in an attempt to find an optimal predictive model, but first standardisation is discussed.

**Standardisation**

Standardising the variables for the training and test data is used to ensure that all numerical variables are on the same scale with mean 0 and a standard deviation of 1. The standardisation will not be performed on dummy variables.

$X= (x- \mu)/\sigma$

Where x is the initial value of the variable, $\mu$ is the mean value for the variable and $\sigma$ is the standard deviation of the variable. For some models, standardisation is a requirement. For others it isn't necessary, though it may be recommended.

**Train/Test Split**

As per standard modeling procedures, the data is split into a training and testing set at a 70/30 split. Training and cross validation will be done on the training data, and evaluation will be done on the test data.

**Linear Regression**

Linear regression is first used as a simple base model to see how it performs with the data, as well as compare other models to. Since standard OLS can be quite sensitive to multicollinearity, *Downtown* is removed and the rest of the original variables are used. Two interaction terms ATMs_Center and Center_Weekday are added as per the EDA. Ten fold cross validation is performed on the training data and both the cross validation and test MSE are output. The results are then recalculated with *Downtown*, as well as just the original six variables for comparison. The results are below.

| Score | Interactions, No Downtown | Include Downtown | Original 6 Variables |
|---|---|---|---|
| Cross Validation MSE | 2.428 | 1.965 | 6.431 |
| Test MSE | 2.340 | 1.832 | 5.880 |

Though high multicollinearity can cause coefficient estimates to be unstable, *Downtown* is left in due to the consistent improvement of the results. The interaction terms are also shown to be quite effective in reducing the MSE.

**Polynomial Features with Forward Selection on Interaction Terms:**

In this section we consider some methods which can improve the simple linear model for manipulating covariates and selecting subsets of predictors.

The relationships between the response and variables displayed in **Plots 1 and 2**, show that the trend of the response does not appear to be quite linear, and since the sample size the model will be built on is large, which makes the fitted model more reliable, generating a new feature matrix consisting of all polynomial combinations of the features is decided. So as to determine a good polynomial order, the average MSE is calculated for each different polynomial orders by using cross validation, with the results given below:

| **Polynomial Order** | **Average CV MSE** |
|---|---|
| 1 | 6.445 |
| 2 | 1.593 |
| 3 | 0.262 |
| 4 | 0.270 |

Results show that the lowest average MSE can be reached when the degree of the Polynomial Features function is three. However, when the degree is three, and the input sample is six dimensional, the number of the features is quite large that can cause another problem: overfitting.

Forward stepwise selection is used not only to prevent this problem; but to find the variables giving the greatest additional improvement to the model, and the number of the features is decreased to twenty from eighty four. In an attempt to reduce the number of variables further without affecting performance, some of the latter variables selected by forward selection are removed and the model refitted. The final set of variables used are in **Table 5**

| | |
|---|---|
| Shops | Shops * Center |
| ATMs | Downtown * Weekday |
| Weekday | Shops^2 * Center |
| Downtown | Shops^3 |
| Center | Weekday * Center |
| High | Shops * ATMs * Downtown |
| Shops^2 | Shops * Center * Weekday |

**Table 5. Predictor set**

An interesting thing to note is the importance of the three-way interaction between *Shops*, *Center* and *Weekday*. This term alone decreases the test MSE score from 1.532 to 0.249. This is suggests an effect on *Withdraw* from *Shops* moderated by *Center* and *Weekday.* Another thing to note is an inclusion of some triple interactions without all the individual and two-way effects. This is a problem for interpretation, but given prediction is the main concern this general rule is relaxed. The cross validation and test MSE for this model are below:

| | |
|---|---|
| **CV MSE** | 0.248 |
| **Test MSE** | 0.249 |

**Ridge Regression**

The ridge regression regularization is used on the predictors selected from the previous method's forward selection. The purpose of using this method is to attempt to further adjust for overfitting by shrinking the coefficients by a shrinkage factor $\lambda$ using the L2 norm as a complexity penalty.

$$\hat{\beta}_{\text{ridge}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_i x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 \right\},$$

The shrinkage factor is 0.000420 selected using cross validation, and applied to the regression. The results are below. Because the shrinkage is extremely small, ridge does not offer any improvement to the model and the MSE remains at 0.249

**Lasso Regression**

The lasso regression regularisation is used on predictors generated from the polynomial feature engineering method. Lasso seeks to shrink large coefficients by applying the L1 norm as a penalty on the magnitude of these coefficients. In addition to shrinkage, Lasso also performs variable selection on potential predictors. With an increase in lambda, there will be a higher degree of variable selection and coefficient shrinkage involved.

$$\hat{\beta}_{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{p} \beta_i x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\},$$

Lasso regression returns a test MSE of 0.283, which is a bit worse because it is now slightly underfitting the model.

**K-Nearest Neighbors**

The K-Nearest Neighbors algorithm can be potentially effective for this data, as it does not make any underlying assumptions and has the ability to capture nonlinear relationships. To determine the k value, a range of possible k values from 1 to 10 were looped and evaluated by calculating the prediction accuracy. Hence, k was chosen to be 9. The predictors used are the original 6 given in the dataset. No additional terms are added as adding too many predictors could subject the KNN algorithm to the curse of dimensionality; as the number of predictors increase, it becomes exponentially more difficult to find training observations that are reasonably local to the prediction point x. This causes the model to be unstable and produce inaccurate results. The KNN results are below.

| | |
|---|---|
| **CV MSE** | 0.323 |
| **Test MSE** | 0.312 |

The KNN model does an alright job of capturing a lot of the nonlinear effects using only the original 6 predictors, but ultimately has a higher cross validation and test MSE. As mentioned above, adding more predictors is not likely to improve the model.

**Neural Networks**

Neural networks have the ability to learn the relationships between a series of inputs such as the independent variable and predictors and the corresponding output of dependent and outcome variables (May, Dandy and Maier 2011). This can be achieved by training the network with the training data so the network can be programmed to adjust internal weights based on the relationships of the inputs and outputs in the dataset. Once the data is trained, it can be used to predict the ATM cash demand in the test data set. Hence, both manual and grid searches are the basic process used to build our neural network model parameters. These methods were relied upon to find the most appropriate combination of activation functions, nodes, optimizers and layers used in the model.

Our process began with a manual search for a benchmark model. This model had the most simple architecture with only one hidden layer with ten nodes. Activation functions and optimizers were defined in the parameter settings for grid search algorithm to select the best

parameters for the dataset. Activations functions used in the grid search were selu, relu, tanh and linear, while the optimisers used in the grid search are shown in Table 6.

| **Types of Optimisers** | **Definition** |
|---|---|
| Adam | Adaptive moment estimation |
| AdaGrad | Adaptive subgradient descent |
| Nadam | Nesterov adaptive moment estimation |
| Adadelta | A novel per-dimension learning rate method for gradient descent |
| RmsProp | Divides gradient by a running average of its recent magnitude |

**Table 6: Optimisers used in grid search**

The use of a grid search determined the 'selu' activation and 'adam' optimizer to be the most appropriate parameter choices. Therefore, Adam will be used as the main optimiser for the neural network. It combines of classical momentum using decaying mean and RMSProp to improve the performance on the number of benchmark. It uses adaptive estimates of lower-order moments for first-order gradient based optimisation of stochastic objective functions (Kingma and Ba 2015). It is easy to implement, computationally effective and suitable for datasets with large parameters. It is also useful for non-stationary objectives and problems with sparse gradients. Based on the results, it has the best performance among the optimisers and works well for the dataset.

Following this, additional nodes and layers were added to benchmark model in the aim to improve the model. Due to the exceedingly high computational power required to perform a grid search over numerous layers, the activation function and optimizer that was identified earlier as the most appropriate were kept for the other additional layers. However, it was the simplest model, as identified below, which ended up having the best performance.

Additionally, since deep learning models are stochastic, each time the model was fitted, the resulting test MSE was different. Therefore, to solve this, the experiment was repeated 30 times and the Grand Mean Test MSE was taken by averaging the 30 individual test MSE scores of the same model. We therefore used figure to evaluate the performance of each Neural Network that was constructed shown in Table 7.

| Model | Grand Mean Test MSE |
|---|---|
| model = Sequential()<br>model.add(Dense(10, activation='selu', input_shape = input_shape))<br>model.add(Dense(10, activation='linear'))<br>model.add(Dense(1, activation='linear))<br>model.compile(optimizer = 'adam', loss = 'mse') | 0.269 |
| model = Sequential()<br>model.add(Dense(50, activation='selu'', input_shape = input_shape))<br>model.add(Dense(50, activation='linear''))<br>model.add(Dense(1, activation='linear'))<br>model.compile(optimizer = 'adam', loss = 'mse') | 0.28 |
| model = Sequential()<br>model.add(Dense(10, activation='relu', input_shape = input_shape))<br>model.add(Dense(10, activation='relu'))<br>model.add(Dense(10, activation='relu'))<br>model.add(Dense(1, activation='relu'))<br>model.compile(optimizer = 'adam', loss = 'mse') | 0.29 |
| model = Sequential()<br>model.add(Dense(100, activation='relu', input_shape = input_shape))<br>model.add(Dense(100, activation='relu'))<br>model.add(Dense(1, activation='relu'))<br>model.compile(optimizer = 'adam', loss = 'mse') | 0.29 |
| model = Sequential()<br>model.add(Dense(100, activation='relu', input_shape = input_shape))<br>model.add(Dense(100, activation='relu'))<br>model.add(Dense(100, activation='relu'))<br>model.add(Dense(1, activation='relu'))<br>model.compile(optimizer = 'adam', loss = 'mse') | 0.32 |

**Table 7: Model Fit**

Our modelling focused on comparing three sets of predictors to evaluate the neural network Test MSE performance. The first is a set of original variables. The second is a set of original variables and two interaction terms based on the exploratory data analysis (EDA). The third is generating a new feature of variables consisting of all polynomial combinations of features less than or equal to the specified degree of 3. In the interest of developing a more interpretable model to manage the bias-variance trade off, forward selection was applied on the polynomial feature engineering to obtain a smaller set of 14 variables. The predictors sets are shown in Table 8.

| Predictors set | Predictors type | Variable | Number of Predictors |
|---|---|---|---|
| 1 | Original Variables | Shops, ATMs, Downtown, Weekday, Center, High | 6 |
| 2 | Original Variables + 2 interaction terms | Shops, ATMs, Downtown, Weekday, Center, High, Weekday*Downtown, ATM*Shops | 8 |
| 3 | Polynomial Feature Forward Selection | Shops,ATMs,Downtown,Weekday,Center, High,Shops^2,Shops*Center,Downtown*Weekday, Shops^2*Center, Shops_3,Weekday*Center,Shops*ATMs*Downtown, Shops*Weekday*Center | 14 |

**Table 8: Predictors Set used in Neural Networks**

Epoch size of 200 and batch size of 100 are used to run the neural network. Although computing the model with one hidden layer is sufficient for most datasets, triple-layer perceptron will possess the ability of modelling non-linear functions and reducing the chances of overfitting. Other than the using best model shown in grid search, other optimisers such as Nadam and Adamax were used to compare the performance. After performing the three predictor sets on the neural network model, predictor set 3 has the best test MSE of 0.256. As the neural network has limited ability to identify causal relationships, it is not efficient in determining the important predictors to a particular output and may contain unimportant predictors (Tu 1996) Hence, variable transformation is essential for the neural network to improve its performance as shown in Table 9 results.

| Predictors set | Neural Network Model | Optimiser | Test MSE |
|---|---|---|---|
| 3 | nn = Sequential()<br>nn.add(Dense(10, input_dim = 14, activation = 'selu'))<br>nn.add(Dense(10, activation = 'linear'))<br>nn.add(Dense(1, activation = 'linear')) | Adam | 0.256 |
| 3 | nn = Sequential()<br>nn.add(Dense(10, input_dim = 14, activation = 'selu'))<br>nn.add(Dense(10, activation = 'linear'))<br>nn.add(Dense(1, activation = 'linear')) | Nadam | 0.261 |

| | | | |
|---|---|---|---|
| 3 | nn = Sequential()<br>nn.add(Dense(10, input_dim = 14, activation = 'selu'))<br>nn.add(Dense(10, activation = 'linear'))<br>nn.add(Dense(1, activation = 'linear')) | AdaMax | 0.262 |
| 2 | nn = Sequential()<br>nn.add(Dense(10, input_dim = 8, activation = 'selu'))<br>nn.add(Dense(10, activation = 'linear'))<br>nn.add(Dense(1, activation = 'linear')) | Adam | 0.284 |
| 2 | nn = Sequential()<br>nn.add(Dense(10, input_dim = 8, activation = 'selu'))<br>nn.add(Dense(10, activation = 'linear'))<br>nn.add(Dense(1, activation = 'linear')) | AdaMax | 0.291 |

**Table 9: Neural Network Results**

### Final Model Selected

The final model selected was the linear regression with interactions terms created by polynomial features of degree 3 and forward selected. This model was opted for over the next best model, the neural network, due to its better score and simplicity. The underlying data as seen in EDA was not at all linear, which favors the use of neural networks to capture complex relationships, but the many interaction terms used in the linear regression model - especially between *Shops*, *Weekday* and *Center* - do enough to capture the nonlinearities.

| **Model Type** | **Model** | **Test MSE Score** |
|---|---|---|
| **Parametric** | Linear Regression | 1.832 |
| | Polynomial Features with Forward Selection on Interaction Terms | 0.249 |
| | Lasso | 0.283 |
| | Ridge | 0.249 |
| **Non-Parametric** | KNN | 0.312 |
| | Neural Network (Predictor Set 3) | 0.256 |

**Table 10: Test MSE Summary**

**Reference**

Kingma, D. and Ba, D. 2015, Adam: A Method for Stochastic Optimization, Cornell University, United States, viewed 5 November 2017, <https://arxiv.org/abs/1412.6980v8>

May, R., Dandy, G. and Maier, H., 2011, 'Review of input variable selection methods for artificial neural networks.', In *Artificial neural networks-methodological advances and biomedical applications*. InTech.

Tu, J.V., 1996, 'Advantages and disadvantages of using artificial neural networks versus logistic regression for predicting medical outcomes.', *Journal of clinical epidemiology*, vol. 49, no.11, pp.1225-1231