PREDICTING ATM CASH DEMAND

Rhys Kilian

October, 2017

Executive Summary

This report outlines the approaches used to develop an analytical model to predict the daily cash demand at ATMs belonging to a particular bank. The model is expected to be used as a decision making tool by the bank to determine how various parameters affect cash demand at ATMs and incorporate such business intelligence into their strategy. Model building commenced with an exploratory data analysis, following which insights from this process were used in feature engineering. New features were built to capture interactions between variables and transform the data.

Following this, a neural network approach was adopted for the purpose of model building whereby various permutations of network structure were tried and optimized to the ADAM (Adaptive Moment) algorithm to a pre-specified model selection criterion based on achieving the lowest mean square error (MSE) through 3-fold cross validation. It was found that a wide and deep model with 3 layers had the greatest predictive accuracy and is recommended for future use.

Contents

1 Introduction

An important aspect of business intelligence for any financial institution (or otherwise) involves asset management which is often informed by heavy data mining of the business' own operations and resources. Such investments in data mining and analysis enable these institutions to leverage off an evidence based approach to asset planing and resource management. From a bank's perspective, one use of data mining techniques, is examining cash demand at ATMs through data collection and predictive model building. The purpose of this analysis is to determine which parameters affect demand for cash at an ATM, and how significant they are. This is used as an approach to evaluate potential ATM sites based on demand (Shiu, 2009). This report outlines the methodology undertaken in such business context whereby a training data set, containing the response variable and six other covariates, is used to build a predictive model. All variables, including the response are described in Table 1.

Table 1: Model variables and response

Variable	**Description**
Withdraw	The total cash withdrawn a day (in 1000 local currency)
Shops	Number of shops/restaurants within a walkable distance (in 100)
ATMs	Number of other ATMs within a walkable distance (in 10)
Downtown	=1 if the ATM is in downtown, 0 if not
Weekday	= 1 if the day is weekday, 0 if not
Centre	=1 if 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

2 Data Processing

The model building processes commences with pre-processing of data to ensure that the dataset is free from errors, presented in the correct format and to check for missing data.

For the dataset at hand, errors may include negative values for the response variable, Withdrawal, Shops and ATMs or non-binary values for Downtown, Centre, Weekday and High. By this criteria, it can be seen from the Table 2 that none of the variables contain any error values. Note that whilst outliers may be considered erroneous, they will not be excluded for the purposes of this analysis.

Table 2: Missing Data

	Total Missing	**Percent**
Withdraw	0	0.00
High	0	0.00
Center	0	0.00
Weekday	0	0.00
Downtown	0	0.00
ATMs	0	0.00
Shops	0	0.00

Subsequently, the data is converted to matrix form to ensure it is in the required format for modeling. There are no missing values in the dataset, hence the data is progressed to

the next stage whereby it is split into 2 parts:

- Training set: used for model building and contains approximately 90% of the data
- Test set: accounted for 10% of the dataset for model evaluation purposes once model selection was complete.

No validation set was produced as the k-fold cross-validation method is used throughout the analysis. This method is preferable over the hold-out validation method as the final result isn't as dependent on the split of the training set, as every data point takes a turn at being in the test set once, and in the training set k-1 instances. As a result, the variance of the estimate is reduced when compared to the hold-out method, giving a better indication of the generalization performance of the model in question (Schneider, 2017).

3 Exploratory Data Analysis

An exploratory data analysis (EDA) is undertaken (on the training dataset only) prior to model building in order to gain further insight into the data. This involves considering the univariate distribution of important variables and the nature of bivariate relationships between different variables and the response, as well as between covariates. Interesting patterns identified from this exercise can be used to inform decisions in the feature engineering and modeling phase prior to model building. Specifically, it can aid in recognizing which variables need to be transformed and which transformations are appropriate, recognizing interactions between covariates and dealing with issues of multicollinearity.

3.1 Univariate Analysis

Figure 1 below, shows the distribution of the response variable 'Withdraw'. The bimodality of this distribution suggests the presence of set denominations of currency resulting in customers withdrawing combinations of those denominations.

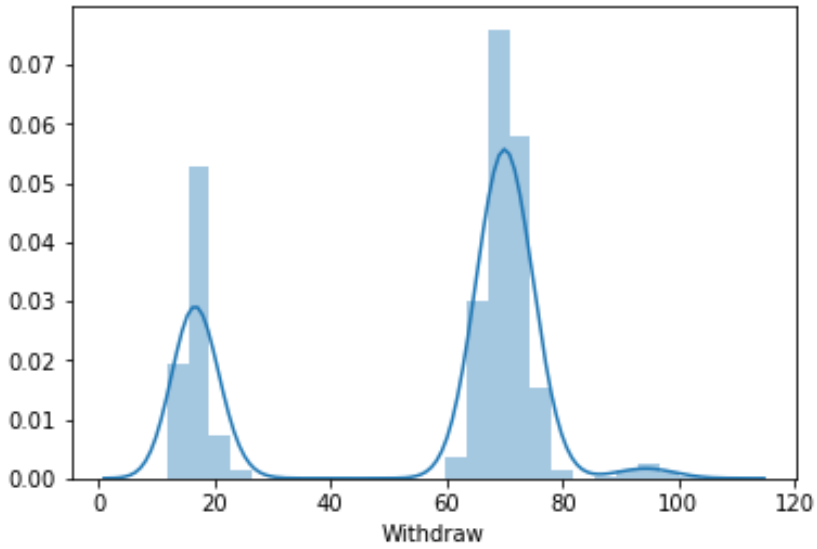

Figure 1: Univariate distribution of the response variable

Table 3 shows summary statistics of the response.

Table 3: Withdraw Summary Statistics (Training)

Count	22,000
Mean	54.65
Std	25.10
Min	11.67
25%	18.50
50%	68.24
75%	71.35
Max	103.96

From the plot in figure 1, it is seen that 15 and 70 are approximately the modes of the two peaks, however, the distribution itself suggests that it is possible to withdraw in increments of less than 2.5 units (or even lower). Moreover, the accuracy of the data points reaches 6 decimal places, seen from the min and max values in the table, suggesting that it is possible to withdraw fractions of the local currency. Given that the Withdraw variable is in 1000s of the local currency, such a level of precision suggests that one-thousandth of the local currency can be withdrawn. This seems unlikely and is probably attributed to ATM fees which are dependent on the amount of value withdrawn. However, for the purposes of this exercise, this matter will not dwelled upon as the primary goal is to build a predictive model based on the given dataset.

Before proceeding to bivariate analysis, it is also interesting to note that the mean in the table, given as 54.65 is highly mis-representative as the average amount withdrawn, given that in the distribution plot, the number of withdrawal amounts around the 50 mark is virtually zero. But the double-peakedness results in an intermediate value between the two peaks being represented as the mean.

3.2 Bivariate Analysis

Prior to examining the relationships between the covariates and the predictors, it is often useful to produce a correlation matrix as shown in Figure 2 to gain an understanding of which predictors have the greatest effect on the response, as well as which covariates are correlated with one another.

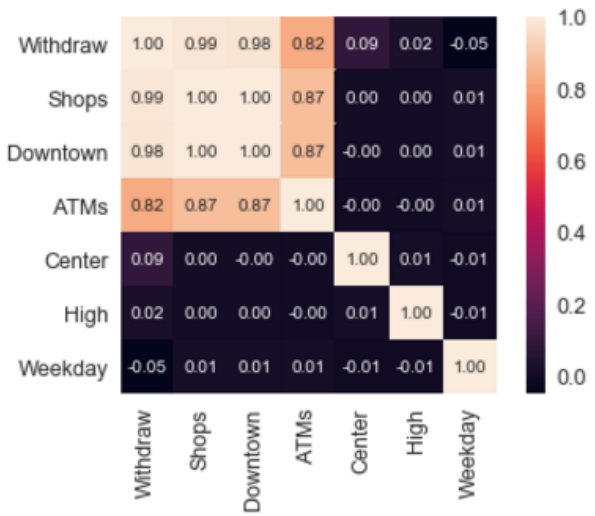

Figure 2: Correlation matrix of all variables in dataset

From Figure 2, ATM, Shops and Downtown are the most correlated with the response Withdraw suggesting that these are likely to be the most important variables in model building. However, the high collinearity between the same three covariates potentially violates the multiple linear regression (MLR) assumption of no perfect multicollinearity; hence conventional linear regression models may not be suitable for the purposes of this exercise unless a complex form of penalty is applied to the collinear predictors. A ridge regression approach could possibly be adopted given that the L2 regularization technique heavily penalizes the presence of multiple covariates contributing to the model in a similar manner, effectively dealing with multicollinearity and producing more reliable estimators.

Figures 4 to 5 in Appendix A shows the bivariate relationship between the response and the numerical variables ATM and Shops. Similarly, Figures 6 to 9 show the distribution of the binary variables, Downtown, Weekday, Centre and High against Withdraw.

Linear relationships are observed between three previously identified covariates and the response whereas Weekday, Centre and High are virtually uncorrelated with the response indicated by the flat slopes of the regression lines. However, violation of homoscedasticity is observed for Withdraw versus Shops, ATMs and Downtown, indicated by the increasing variance in all cases. Transformations can be applied to see if there are any effects however this is unlikely given that the response itself is asymmetric and bimodal in its univariate distribution resulting in skewness.

3.3 Outliers

As the distribution is bimodal, the data is initially standardized before outliers are identified. Table 4 shows the lower and higher ends of the distribution. Given that the data points are quite close together in both cases and the values themselves are not too far from 0 (within 2 standard deviations), it can asserted that there are no outliers in this scenario. Hence, all data points will be retained for the purposes of model building.

Table 4: Out Range of Withdraw Distribution

Low Range of Distribution	**High Range of Distribution**
-1.713	1.964
-1.704	1.960
-1.700	1.940

4 Methodology

A supervised learning approach is adopted for the purposes of model building in this context, whereby the training data is used to develop a model. Hyperparameters are chosen using a cross-validated grid search approach. The model with the best predictive performance in terms of the cross validated MSE is selected and used to determine the test score for the purposes model evaluation and subsequently final submission.

4.1 Model Selection Criterion

The different models considered were compared using a cross-validation mean squared error (MSE) score. The cross validation approach takes multiple splits of the training data. Given the computational cost increases for larger number of splits and the large number

of predictions involved, a 3 and 5-fold cross validation was used for these models. For each fold (split of the training data), the model is estimated on all other folds combined, with the fold in question used as the validation set. The cross validation error is taken as the average mean squared error across the 3/5 validation sets.

4.2 Modelling

Multiple linear regression (MLR) and linear regression models in general are usually appropriate when the functional form of the relationship between the covariates and response are well defined and known. From the EDA, it was clear that the data violated several assumptions of linear models including heteroscedasticity, multicollinearity and linearity (for some predictors). These attributed render the linear regression approach unsuitable for the purposes of this exercise and instead, neural networks are preferred. Moreover, the large amount of data can be used to develop a deep learning structure.

Neural network modeling works by inductive data representation learning methods whereby the training dataset is used to transform covariates such that the loss function is optimised (in this case for the validation fold). In forward-feed or forward-propagation neural network models, the model takes in a set of inputs, feeds it through layers of composite functions and in the last layer, produces an output (which in this task is a scalar output as the response variable is of numeric type). The loss is computed as the difference between actual and predicted output, as in any other regression or classification problem.

Each layer that is intermediate between the input and output nodes is a composite function of hidden processing units which are essentially the covariates in the dataset. Each node is dictated by a propagation rule and an activation function whereby the former specifies the total input from the units that send information to the node and the latter is a function that determines the output based on the input. The weights of connections between layers and also the weights of the nodes are determined within the model by running the model for several epochs of a mini-batch size (a subset of the data). The presence of many layers is referred to as deep learning or deep neural network modelling.

The core objective of the neural network modelling process is to build a model by the number of layers (and nodes in each layer), propagation rule, activation function and connection weights such that for the loss in minimized through an optimization algorithm. Optimisation algorithms identify the optimal set of hyperparameters which result in a minima in the loss function. They achieve this by initializing a set of values and then, through inductive deep learning, moving in steps (known as learning rate) towards localized lower losses, eventually converging to a minimum. Different algorithms exist for such purposes, with static and adaptive learning rates (discussed in later sections)

The chosen hyperparameters for this assessment are detailed in the subsequent section titled Model Training.

4.3 Model Training

In the process of model building and selection, the following processes were implemented sequentially. The details of each process are discussed in the individual subsections to follow:

1. Model structuring: determining the number of neurons and number of layers

2. Data transformation
3. Epochs and batch size (determining the optimum number of epochs and batch size)
4. Algorithm for learning rate optimisation
5. Weight initialization
6. Dropout regularization

For the purposes of this investigation, the random seed was set to a value of 7. Furthermore, it should also be noted that an iterative, yet sequential process was completed to determine the following model. While every effort was made to complete the steps in the above order, processes which showed promise in lowering the training MSE were focused on. Therefore, the train MSE a section of this report may be higher than its previous section as it showed more promise and was hence focused on for further analysis.

4.3.1 Data Transformations

As mentioned previously, the fundamental aspect of neural network modeling is changing the format of data representation to make classification or in this case, regression, more meaningful. Transforming the data enables all the data to be on a similar scale, which results in a smoother learning process. The three different data transformation methods that were tried as part of this process were inclusive of the following:

- Normalization, whereby the data was fit to normal distribution
- Standardization, whereby the data was fit to a normal distribution and scaled such that the mean was 0 and standard deviation equaled 1.
- Scaling between 0 and 1.

The 5-fold cross validated MSE scores are given in the table below. These were tested on an initial model (12 neurons $\rightarrow$ 6 neurons $\rightarrow$ one neuron). It can be seen in table 5 that standardization obtains the lowest MSE and will therefore be used as the chosen transformation for further analysis.

Table 5: Data Transformation & MSE

Transformation	**Validation MSE**
Normalisation	8.44
Standardisation	0.29
Between 0 and 1	8.16

Shanker, Hu and Hung (1996) investigated the effect of data standardization on neural networks, ultimately concluding that standardized data leads to better results. However it was found that the benefit to standardizing decreases as network and sample size increases, along with standardization slowing down the training process. However given the relatively small scale of this investigation, the standardization process is seen to be optimal.

4.3.2 Epochs and Batch Size

In training the model, the neural network is usually run through several iterations as deep learning is an inductive learning process. Each iteration is referred to as an epoch and increasing the number of epochs minimises the training error to an optimal level until a threshold is reached, beyond which running more iterations makes no difference to the model.

For example, Figure 3 shows how the MSE changes with increasing epoch number. This is completed on the training data and a validation set, which represent 70% and 20% of the total data, created solely for the purpose of this image. As can be observed, the loss for both the training and validation sets decrease rapidly for low values of epochs. The training loss value is relatively stable for all values of epochs, however, the validation (test as it is commonly (mis-)referred to as) is unstable. Nonetheless, this figure suggests that some threshold exists such that no further prediction performance can be gained.

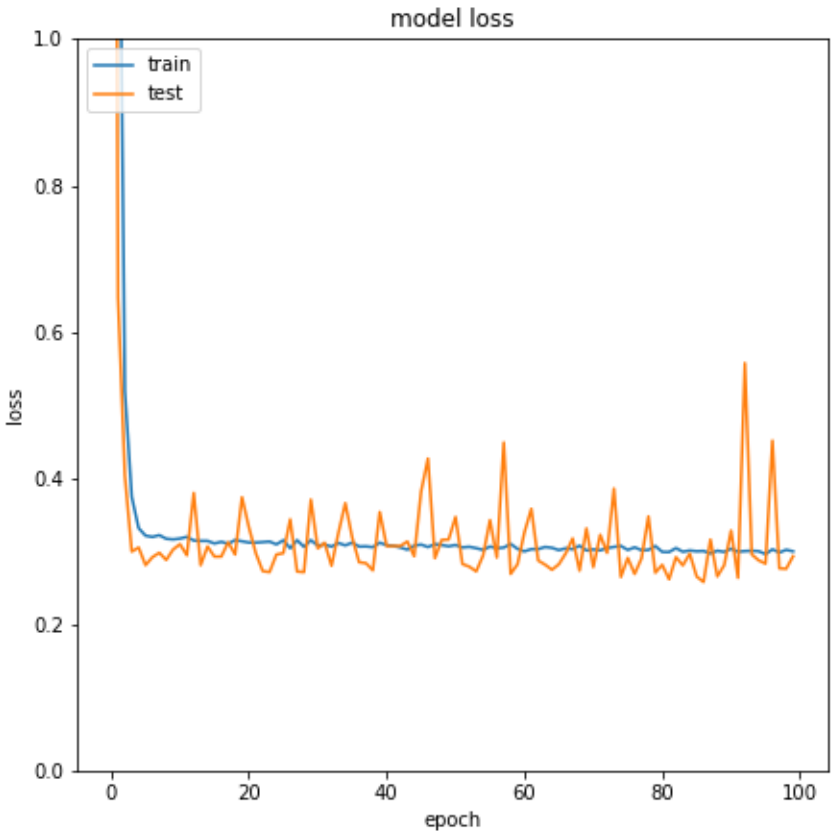

Figure 3: Loss Compared to Epoch Number

Each epoch doesnt necessarily use the entire data as this is often computationally intensive, hence a subset is chosen. The size of the subset is referred to as batch-size.

Choosing an optimum number of epochs and the appropriate batch size is of high importance as it determines the efficiency and reliability of the model building process. Hence, to assist in choosing appropriate values for these hyperparameters, grid search cross-validation is used whereby the grid search process in *scikitlearn* constructs and validates a model for each combination of parameters within a specified range. 3-fold

cross validation is then used to evaluate each individual model to identify the combination of hyperparameters which yield the lowest MSE. For the purposes of this exercise, grid search was performed within a batch size range of 10 to 80 in increments of 5. Epochs tested were from 10 to 80 in sizes of 10.

From the specified ranges, grid search identified a batch size of 50 run for 80 epochs to be the optimal hyperparameter configuration as it yielded a minimized validation MSE of 0.275. Table 6 shows a series of selected results.

Table 6: Batch Size and Epoch CV MSE Results

Batch Size	**Epochs**	**3-Fold CV MSE**
35	40	0.289
45	60	0.280
50	80	0.275
60	80	0.281
70	80	769.121
80	60	0.313

4.3.3 Training Algorithm Optimisation

A 3-fold grid search on different optimization algorithms were performed to find the one that yielded the lowest MSE.

As discussed previously, optimization algorithms can be adaptive or static; the former achieves loss minimization by using a coordinate specific learning date (i.e. as the variance of the loss function decreases, the learning rate is reduced in order to converge on minima) whereas the latter identifies a loss function minima through a constant learning rate. The mean square errors from different algorithms vary, as shown in the table below.

Table 7: Optimisation Algorithms

Optimisation Algorithm	**3-fold CV MSE**
Stochastic Gradient Descent (SGD)	455.021
RMSprop	636.213
AdaGrad	3.049
AdaDelta	0.277
ADAM	0.286
AdaMax	381.974
NADAM	0.276

It was found that the Nadam and Adadelta optimisation algorithms performed the best. However, it was found that these two optimisation algorithms performed poorly when further hyperparameter selection was conducted. This suggests that their performance is not independent of the other hyperparameters, which wsa to be expected.

On the other hand, consistent results were obtained with the ADAM algorithm. These were replicated and remained fairly constant for other hyperparameter selection processes. Therefore, the ADAM algorithm was selected, despite its higher MSE.

4.3.4 Weight Initialisation

Furthermore, it is also possible to initialise the weights of the neural network using the Keras package prior to training. This is expected to increase the speed and accuracy at which an appropriate model converges. Within the Keras package, a series of initialisation are specified. The ones which were analysed using the 3-fold CV grid are provided in the table below along with their associated 3-fold CV MSE scores.

Table 8: Weight Initialisation Approaches

Initialisation	**3-fold CV MSE**
Uniform	645.175
Lecun_uniform	0.294
Normal	0.280
Zero	1926.944
Glorot_normal	0.301
Glorot_uniform	0.290
He_normal	0.282
He_uniform	0.294

Using this approach, the normal weight initialization was found to be the best performing. Therefore, it was selected for further analysis.

4.3.5 Dropout Regularization

Dropout is a regularization technique to improve the generalization performance and reduce the chance of over-fitting of a neural network. It is a technique proposed by Srivastava, et al. (2014) in which selected neurons are randomly removed during the training process. This means that other neurons in that given layer are required to handle the representation and prediction that the dropped neuron otherwise would have. As a result, the neural network developed becomes less sensitive to the weights of particular neurons. Hence, this is over-fitting is reduced.

In order to get good results from dropout regularization, it is suggested that dropout is combined with the max norm weight constraint(Brownlee, 2017). Dropout was implemented on the first hidden layer of the neural network being trained. Using a 3-fold cross validation approach several combinations of dropout percentages were considered in tandem with several maxnorm weight constraint values. The grid tested ranged from:

- *Dropout percentage*: 0.0 - 0.9, increasing by 0.1
- *Maxnorm weight constraint*: 0 - 5, increasing by 1

The results of a select few combinations of dropout and maxnorm weights are included in table 9:

As can be seen in table 9 the combination which yielded the lowest CV-MSE was the model with a dropout rate of 0.0 and a weight constraint of 5 with a score of 0.282. Clearly, adding dropout regularisation on the first hidden layer of the neural network did not improve model performance, in fact, it had the opposite effect. This suggests that after dropping a neuron from the first hidden layer of the model the remaining neurons were unable to represent that data properly. As a result, a poor CV MSE was obtained. For these reasons, dropout regularisation was not implemented on the final model on the first hidden layer.

Table 9: Dropout Regularisation Results

Dropout Rate	Weight Constraint	3-Fold CV MSE
0.0	5	0.282
0.1	2	4.260
0.2	5	6.079
0.3	5	9.561
0.4	2	23.628
0.5	5	36.910

4.3.6 Model Structure

Similarly, the number of layers in the model and the number of neurons was determined by using a 3-fold cross validation grid search using the training data.

Wide and deep networks often lead to optimal performance, whilst reducing the need for feature engineering (Dhanjal, 2016). Furthermore it can be adapted to a wide variety of problems. However wide a deep networks are computationally expensive and require a large amount of data. Furthermore there are no strong guidelines for deciding on topology and hyperparameters (Dhanjal, 2016). Overall these methods do have significant advantage to reducing MSE.

The network topology which yielded the lowest mean square error (MSE) was chosen for further analysis. The first and second layers were grid-searched with the following number of layers:

- Layer 1: [1, 5, 10, 12, 15, 20, 25, 30] neurons
- Layer 2: [1, 5, 10, 12, 15, 20, 25, 30] neurons

Table 10 below shows three different topologies that were trialled and their associated MSE score.

Table 10: Model Structure Topologies & MSE

Topology	3-fold CV Grid Search MSE
1 neuron $\rightarrow$ 1 output	8.012
20 neurons $\rightarrow$ 1 output	0.305
30 neurons $\rightarrow$ 1 output	0.276
30 neurons $\rightarrow$ 1 neuron $\rightarrow$ 1 output	0.277
30 neurons $\rightarrow$ 15 neurons $\rightarrow$ 1 output	0.279
30 neurons $\rightarrow$ 25 neurons $\rightarrow$ 1 output	0.276
45 neurons $\rightarrow$ 15 neurons $\rightarrow$ 1 output	0.275

Hence, the model with a topology of 45 neurons $\rightarrow$ 15 neurons $\rightarrow$ 1 output was selected for further analysis as it had the lowest MSE.

4.4 Final Model

The final model is characterised by the following hyperparameters:

- Network structure: Named as '45-15-1', the model consists of 3 layers with the number of neurons in each layer being 45, 15, and 1 in the same order with the last layer constituting as the output.

- Activation function: Rectifield Linear Unit (ReLu) is used as it is much faster at training neural networks compared to other functions due to its sparsity and constancy of gradient.
- Weight initialisation: Normal
- Optimisation algorithm: Adaptive Moment (ADAM)
- Batch size: 45; epochs: 90
- Standardised data transformation
- Dropout regularisation: None

4.5 Model Evaluation

The model structure detailed in the preceeding section was chosen based on its low cross-validated MSE score on the training data. When completing model evaluation on the test data a MSE score of 0.271 was obtained. Compared to the baseline ridge regression model which had a MSE score of 2.43 based on 5-fold cross validation, the chosen model yeilded more than 93% reduction in the cross-validated MSE score. Moreover, it is worthy to note that the final model had a very similar test and train score indicative of the model's reliability.

As the model selection has been completed, the next step involved evaluation whereby the trained, submitted model would be used to predict the test set (which formed 10% of the original training dataset). From this the test error would be computed as a metric of the model's predictive performance.

5 Conclusion

As per the initial intent, a predictive model was built to predict the daily ATM cash demand based on base variables which considered the ATM's proximity to shops, other ATMs and location in a business district, among other parameters. A deep neural network modelling approach was undertaken due to nonlinearities in the predictors and covariates. Model selection through optimisation of hyperparameters (chosen by minimising MSE) resulted in the development of a model with a validation error of 0.27, yielding a 93% reduction in CV MSE score compared to the baseline logistic regression which was set as an initial benchmark. The neural network that achieved this had 3 layers (named 45-15-1, in alignment with the number of nodes in each layer).

It is expected that this model can greatly assist in the bank's asset management, resourcing and future planning strategies; understanding the parameters that affect cash demand will enable the bank to realise which ATMs need to be equipped with higher cash capacities (and consequently better security systems) and where more ATMs need to be installed based on the correlations observed between ATM location and demand.

Whilst the model has high predictive accuracy (an assertion made based on its low CV score), to further improve the model, the following recommendations are made:

1. Randomised search: Hyperparamater optimisation can be made an easier and more computationally efficient process by adopting a randomised grid search. THis method is usually preferred over grid search as it doesn't get too involved on finding the local optima. Moreover, with increasing number of parameters, grid search becomes tedious and exhaustive, in which case a randomised approach would assist (Bergstra et al. 2011)

2. Increase computational power: This enables various combinations of hyperparameters to be searched at any given time without having to do them independently.This makes the process of hyperparameter optimisation more efficient and also allows interrelationships between parameters to be captured.

3. Further feature engineering: Create 'flags' based on pre-established criteria to alert the model fo any patterns in the data. Such flags can be coded in with additional dummy variables to allow the neural network to find further patterns within the data

6 References

Bergstra, J. and Bengio, Y., 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb), pp.281-305.

Brownlee, J. (2017). How to Grid Search Hyperparameters for Deep Learning Models in Python With Keras. [online] Machine Learning Mastery. Available at: https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/ [Accessed 5 Nov. 2017].

Dhanjal, C. (2016). On Artificial Neural Networks and Deep Learning. [online] SimplyML: Simply Machine Learning. Available at: https://simplyml.com/on-deep-learning/ [Accessed 6 Nov. 2017].

Schneider, J. (2017). Cross Validation. [online] Cs.cmu.edu. Available at: https://www.cs.cmu.edu/ schneide/tut5/node42.html [Accessed 5 Nov. 2017].

M. Shanker, M.Y. Hu, M.S. Hung, Effect of data standardization on neural network training, In Omega, Volume 24, Issue 4, 1996, Pages 385-397, ISSN 0305-0483, https://doi.org/10.1016/0305-0483(96)00010-2.

Shiu, Eric C. C 2009, Marketing research, McGraw Hill Higher Education, London

Srivastava, N., Hinton, G., Krizhevsky, A. and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning, [online] 15 (Jun), pp.19291958. Available at: http://jmlr.org/papers/v15/srivastava14a.html.

A Further Exploratory Data Analysis

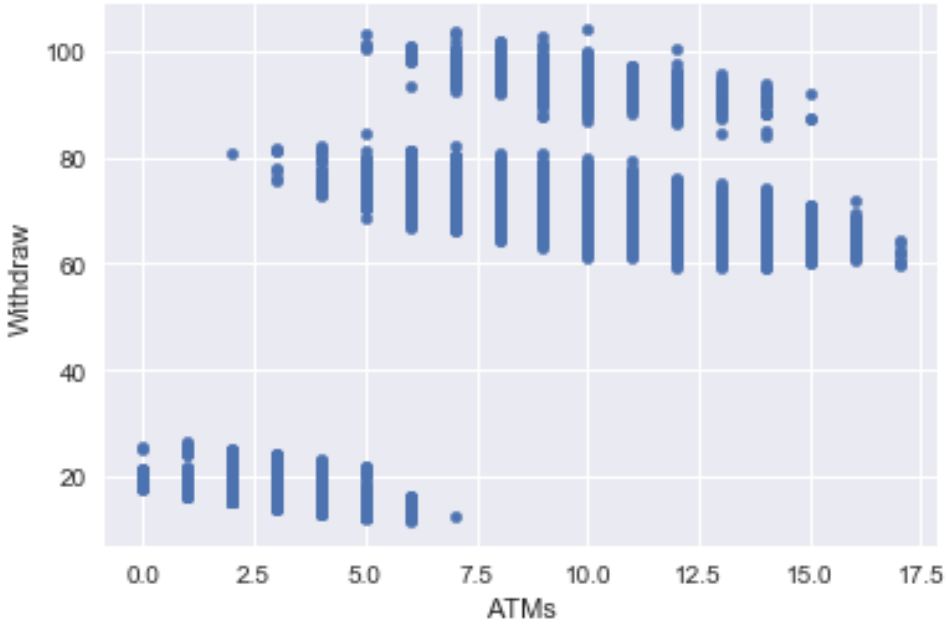

Figure 4: Bivariate Distribution of Withdraw and ATMs

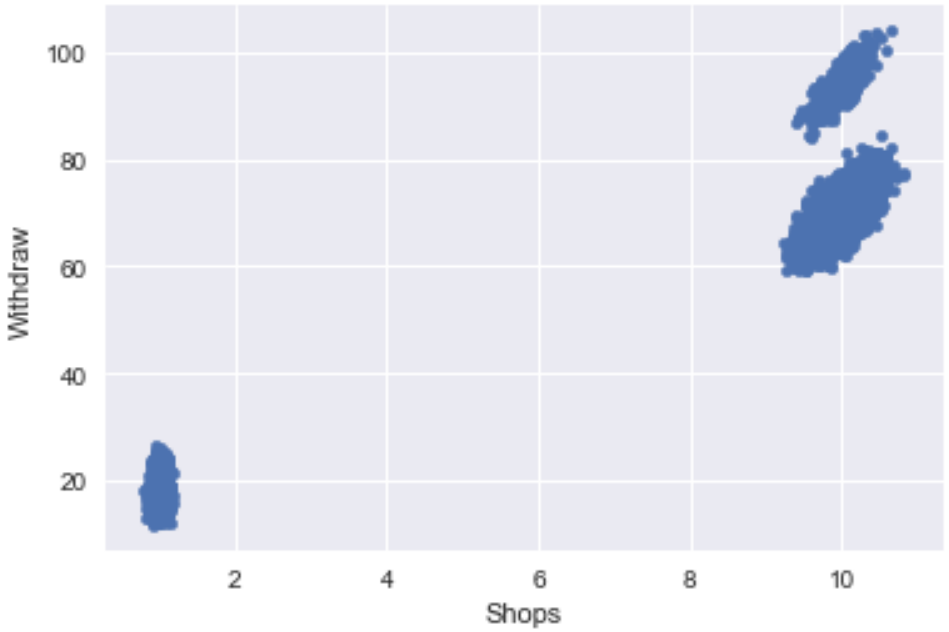

Figure 5: Bivariate Distribution of Withdraw and Shops

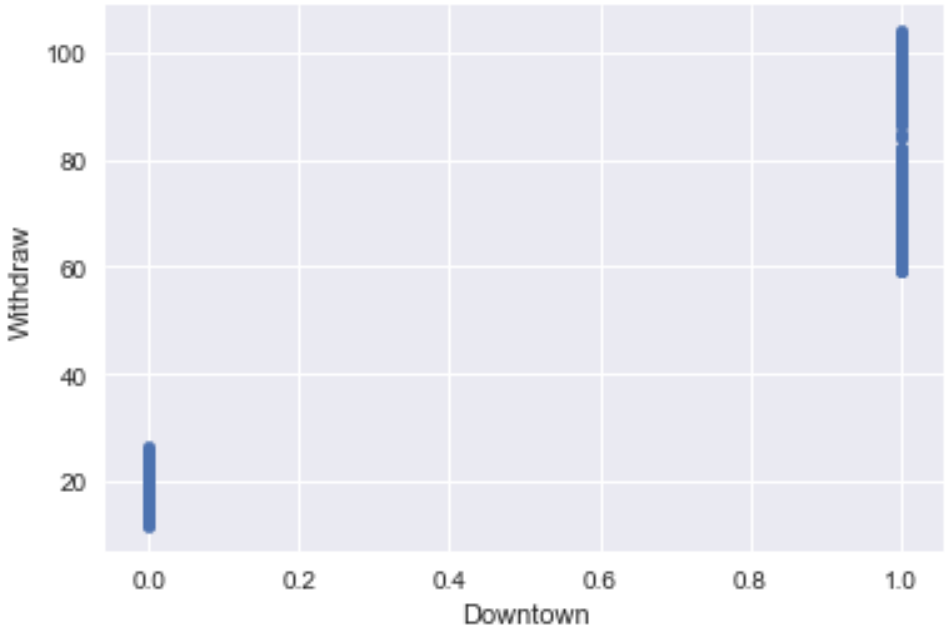

Figure 6: Bivariate Distribution of Withdraw and Downtown

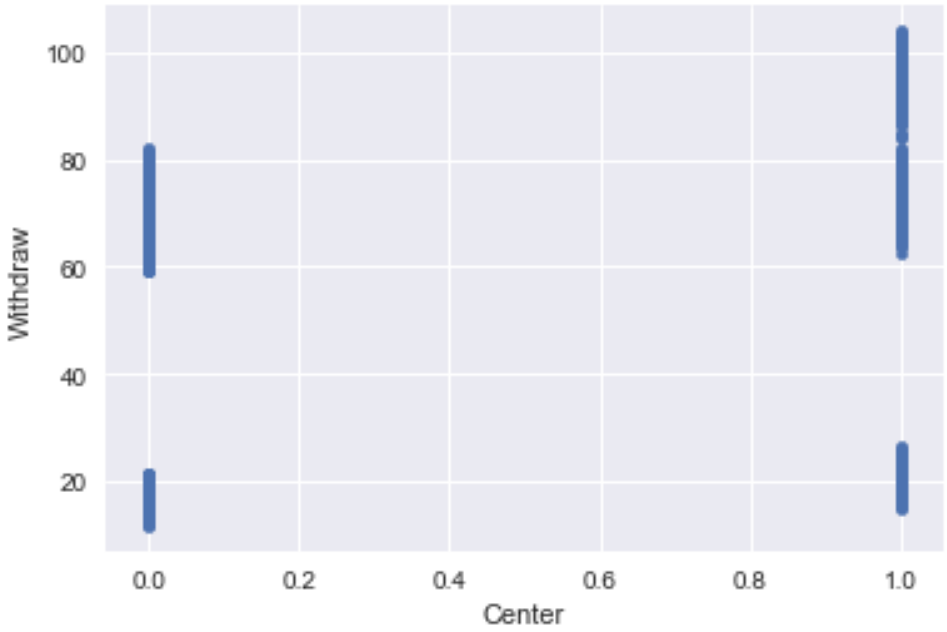

Figure 7: Bivariate Distribution of Withdraw and Center

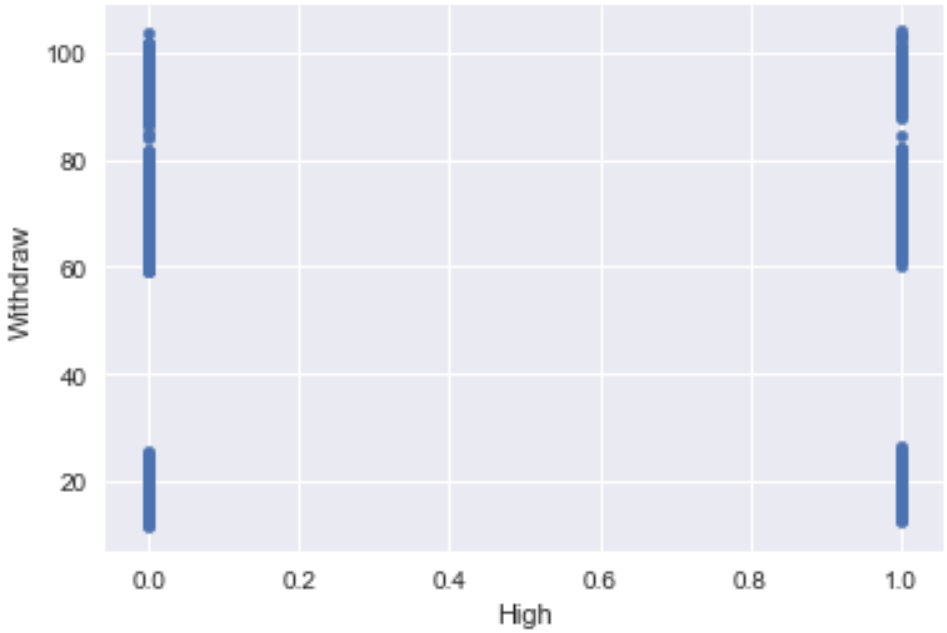

Figure 8: Bivariate Distribution of Withdraw and High

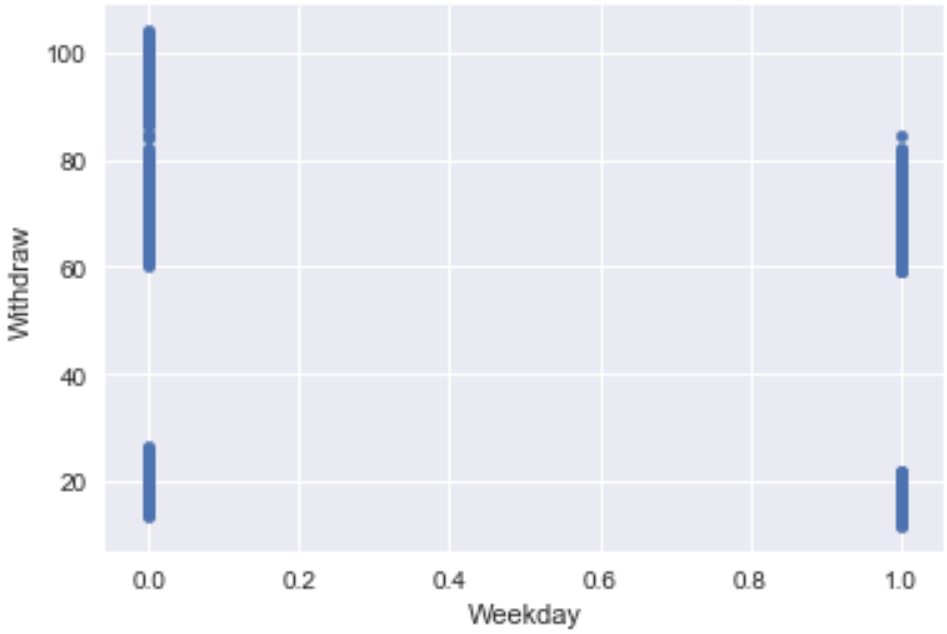

Figure 9: Bivariate Distribution of Withdraw and Weekday