VinUniversity

BANA4040 Predictive Analytics

Group Assignment

FORECASTING ATM CASH DEMAND: A PREDICTIVE MODELING APPROACH

Group 7

19 January, 2023

Group Members:

Le Linh Dan

Phung Trong Nghia

Do Quynh Trang

Lam Nguyen Thanh Thao

I. Executive summary

Forecasting demand has been one of the major applications of predictive analytics method in real life. Among many businesses, cash demand in ATMs requires the most accurate prediction. Each financial institution has thousands of ATMs across the nation. Therefore, a minor error in the forecast could cost incidents of cash-outs, overstocks, and other challenges. In reverse, an accurate forecast would help the financial institutions enhance efficiency by better managing their cash distribution and lowering the cost of replenishment.

Although there are a few AI solutions forecasting ATM cash demand available on the market, different local culture and different bank charateristics means that there is no one-size-fit-all cash demand model. Each bank should develop their own forecast model based on the historical data from their current network of ATMs.

In this report, we aim at constructing a forecast function for a specific bank to calculate the amount of cash withdrawal per day, based on certain characteristics of individual ATMs. After conducting different predictive modelling methodologies, we selected feed-forward neural network as the most suitable model, due to lowest MSE, mitigation of multicollinearity issue and reduced risk of overfitting.

Since banks would store 40% more cash in ATMs than its expected demand (Simutis, et. al, 2008), this prediction function is expected to optimize cash management and contribute to the mentioned bank’s higher earnings.

II. Data Overview

1. Business Context

Automated Teller Machines are the critical touchpoints between the banks and the individual customers. Therefore, it is crucial to develop an effective ATMs network for both banks and customers, which includes forecasting an accurate amount of cash withdrawal per ATM. Regarding the today competiveness of banking industry, regulating the minimum number of cash in the ATM, or avoiding “cash-outs”, will retain the customers' satisfaction level. According to survey conducted by Armenise, et. al. (2013), 21% of correspondent switch to competitor bank's ATMs due to cash absence situation. Meanwhile, having an excessive amount of money unused in an ATM, or “overstock”, also means loss to the bank because that resting amount could otherwise be utilized as an profitable investment.

Despite the importance of placing the appropriate amount of cash into ATMs, based on Akber, et. al, (2018), most banks still utilize the common, basic approach of doing an estimatios by the expected value, rather than considering the fact that the cash outflow of each ATM will depends on that machine's peculiar characteristics (location, time, nearby services, etc). Simutis et al. (2007) suggests that banks should manage their cash reserve more effectively by relying on advanced algorithms to accurately forecast cash supply and demand for each ATM.

Currently, there have been many software solutions for ATM networks as shown in Table 1.

Company	Software	Web page link
Carreker Corpora-tion	iCom	http://www.carreker.com/main/solutions/cash/icom .htm
Morphis, Inc	MorphisCM	http://www.morphisinc.com/product.php?pageIn = MorphisCM
Transoft Interna-tional	OptiCa$h	http://www.transoftnic.com/site/index.php?option = com_content&task = view & id = 18 & Itemid = 40
Wincor Nixdorf	Pro Cash Analyser	http://www.wincor-nixdorf.com/internet/com/Product/Software/Banking /CashManagement/Main.html

Table 1: Popular commercial solutions in use, Akber, et al., (2018).

However, these applications are not widely applicable for all banks. First of all, many small-sized banks do not possess the luxury of deploying an expensive software to predict cash flow. Second of all, Simutis et al. (2007) believes that local rule and culture encourage banks to build their models based on past transactional data from their own ATM network. This is considered as the most reliable and cost-efficient approach for banks. We also assume this as the rationale for our report, that is to help the designated bank to develop its own cash withdrawal forecasting model.

2. Data Background

The train dataset, namely ATM_training.csv, used this report was originated from the ATM network of a bank. The original dataset consists of 22,000 data points with 7 variables. The dependent variable is Withdraw, which is the total daily cash outflow from each ATM. There are six response variable and covarriate variables as follows:

- Shops: Number of shops/restaurants within a walkable distance (in 100)
- ATMs: Number of other ATMs within a walkable distance
- Downtown: =1 if the ATM is in downtown, 0 if not
- Weekday: = 1 if the day is weekday, 0 if not
- Center: =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.

The test dataset was also given in order for our team to make prediction and assess the accuracy of the selected predictive model. This dataset has the similiar structure to the *ATM_training.csv.*

3. Exploratory Data Analysis

Step 1: Data Understanding & Cleaning

We first examine the data by understanding basic components of this dataset. All datapoints are recorded either in float or interger type, thus, there is no need to further converting categorical variables into dummy variables. Another fortunate feature of this dataset is that there are no null values in any row or column.

```
RangeIndex: 22000 entries, 0 to 21999
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Shops     22000 non-null  float64
 1   ATMs      22000 non-null  int64  
 2   Downtown  22000 non-null  int64  
 3   Weekday   22000 non-null  int64  
 4   Center    22000 non-null  int64  
 5   High      22000 non-null  int64  
 6   Withdraw  22000 non-null  float64
dtypes: float64(2), int64(5)
```

Figure 1: Basic information of ATM_training.csv.

In terms of outliers, all seven variables have no data point beyond their whiskiers. Since there is no outlier observation, we decide not to omit any value from the original dataset, since we do not want to hold any assumptions about the significance and the relevancy of variables before constructing the models.

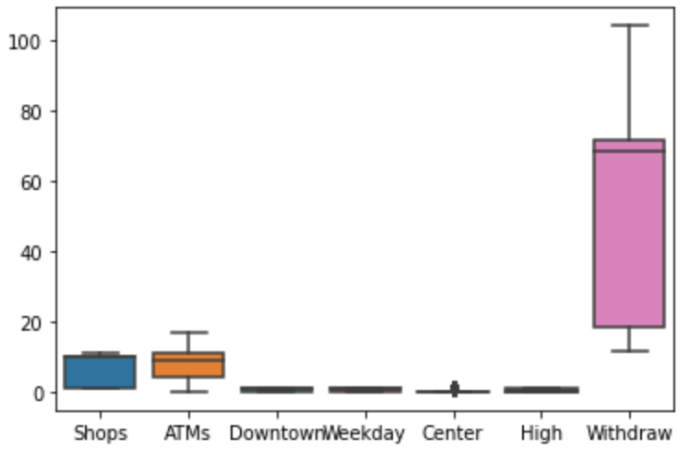

Figure 2: Box Plot of ATM_training.csv.

Step 2: Multicollinearity Checking

First of all, we examine the correlation relationship between six response variables and dependent variable “Withdraw" (shown in Output 2). The result indicates that Shops, ATMs, Downtown are top three variables highly correlated with the amount of cash outflow, whereas Weekday, Center and High are weakly correlated with “Withdraw" variable. However, the result cannot infer any causal relationshop. We need to use linear regression techniques to construct a more exact model.

Shops	0.985797
ATMs	0.824030
Downtown	0.983574
Weekday	-0.050470
Center	0.088103
High	0.021275
Withdraw	1.000000

Figure 3: Correlation result in relation to Withdraw.

Second of all, when we extend the correlation matrix to analyze the relationship between variables, mutlilinearity exists between three variables: Shops, ATMs, and Downtown. This may be due to the fact that downtown location, which is well-known for attracting huge number of consumer, is a popular option for both shop owners and banks. As multicollinearity has been exhibited, we take into consideration this fact when selecting the methodology that will substantially decrease the risk of multicollinearity.

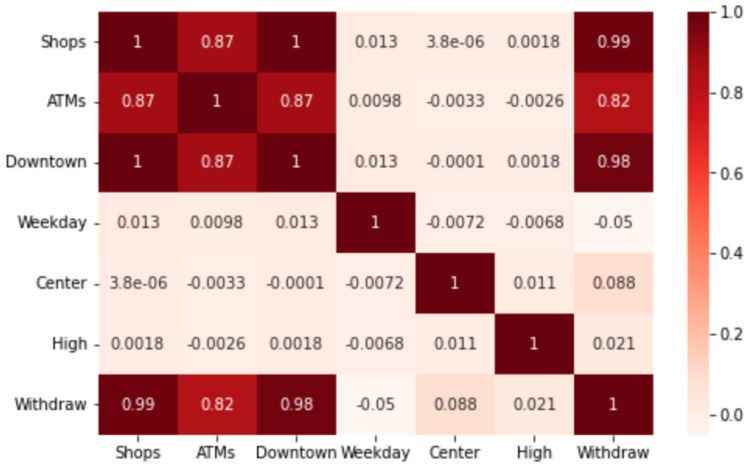

Figure 4: Heat map of correlation matrix (ATM_training.csv).

Step 3: Data splitting

In order to check and compare the MSEs of OLS, lasso regression, ridge regression, and elastic net regression, and neural network, we split the data into 2 subsets: 80% of the original dataset is the training data, while the 20% left is the testing data used for model validation.

We first set the global random seed for numpy and tensorflow to save the state of our random function for data splitting. In other words, it helps us ensure that the same subsets of data appear each time we run the codes. This is an essential step to guarantee consistent results across multiple code executions. Then, we split the data and had 4 variables, namely x_train, y_train, x_test, and y_test. X_train and y_train were used for model fitting, while x_test and y_test were used for validating and comparing the accuracy across different prediction models.

Step 4: Standardization

Standardization is another important pre-processing step before executing machine learning models like Ridge, Lasso, and Neural network. Ridge and Lasso regressions place a penalty on the magnitude of the coefficients associated with each independent variable, and the scale of variables will affect how much of a penalty will be applied to their coefficients. For example, coefficients of variables with large variances are small, so they will be less penalized than those with smaller variances.

Z-score is one of the most well-known methods for standardization, which is calculated by substracting the mean and dividing by the standard deviation for each value of each data element. Once the standardization is done, all

data have a mean of zero and a standard deviation of one, so they have the same scale. This method is adopted for our standardization stage.

III. Methodology

1. Ordinary Linear regression

In this project, we first used the Ordinary Linear Regression (OLS) technique to analyze the relationship between the Withdraw variable and the other variables in the training dataset. OLS is a statistical technique that is used to model the relationship between variables. The basic idea behind this method is to find the line that best fits the data, where the line is determined by the coefficients of the independent variables. Here, we developed and fit the OLS model using the training set through the following three steps.

Firstly, we used the OLS function from the statsmodels library to develop a linear regression model using the Withdraw variable as the response variable and the other variables as the predictor variables. The standard OLS model is specified as:

Withdraw = α + β_1Shops + β_2ATMs + β_3Downtown + β_4Weekday + β_5Center + β_6High + ε

Next, we fit the linear regression model to the training data using the fit method. Then, we calculated the Mean Squared Error (MSE) of the model using the test data by dividing the sum of squared residuals (SSR) by the number of observations, which is 6.202.

To further improve the model, we added interaction terms between the predictor variables to the model, specifically Shops*Weekday and Shops*High, based on the context of this business problem.

- We discovered several research suggests that these variables may have an interaction effect on cash withdrawals. Studies by Goh et al. (2018) and Ng et al. (2020) found that consumer spending is positively correlated with the number of shops in an area and are higher on weekdays.
- Furthermore, Liu et al. (2021) and Gao et al. (2019) found a positive correlation between ATM cash withdrawals and commercial activity.

We refit the model based on the new formula that includes interactions terms, which is

$$\text{Withdraw} = \alpha + \beta_1\text{Shops} + \beta_2\text{ATMs} + \beta_3\text{Downtown} + \beta_4\text{Weekday} + \beta_5\text{Weekday*Shops} + \beta_6\text{Center} + \beta_7\text{High} + \beta_8\text{High*Shops} + \varepsilon$$

We then calculated the MSE of the model using the test data, which is 6.047. This result indicates that the inclusion of the interaction terms in the OLS model improves predictive accuracy.

2. Ridge

Ridge regression, also known as L2 regularization, is a machine learning technique used to analyze multiple regression data suffering from multicollinearity and reduce the overfitting of linear models by penalizing the coefficients of the variables. When multicollinearity occurs, least squares estimates are unbiased, but their variances are too large to be true. By adding a penalty term to the error function that shrinks the size of the coefficients, ridge regression reduces the standard errors and gives more reliable and accurate predictions. Python provides multiple Ridge regression implementations, including Ridge from the skicit-learn package and RidgeCV

from the statsmodels package. Both of these packages were utilized for our calculations.

As mentioned above, ridge regression puts a constraint on the coefficients by introducing an error term called the lambda parameter (denoted as λ). The shrinkage penalty is lambda times the sum of squares of the coefficients, so coefficients that get too large are penalized. As lambda gets larger, the bias is unchanged, but the variance drops. Lambda controls the tradeoff between the penalty and the model fit; therefore, selecting a proper lambda has a crucial impact on the output. If $\lambda = 0$ or excessively small, the results approach OLS estimates, and the model fails to remove any variable. In other words, the model is unpenalized for complexity, potentially leading to unresolved overfitting issues. In contrast, if the λ is unreasonably large, the penalty dominates the objective function and removes variables that may be significant in the analysis, resulting in an underfitting model with greater variance.

To find the optimal lambda, we use the cross-validation technique. First, we created an array of all potential lambda values. Then, we did cross-validation for all the specified values with the RidgeCV method and found the optimal lambda, using MSE as the criteria. Finally, we fitted the model with the train data. Applying the generated model to our labeled test data, we arrived at an MSE of roughly 6.296.

3. Lasso

Similar to Ridge regression, Lasso is a regularization technique that adds a penalty term to achieve a minor variance with the tested data and restricts the influence of predictor variables over the output variable by compressing

their coefficients. Unlike Ridge which penalized the sum of the squared coefficients, Lasso adds a penalty equal to the sum of the coefficients' absolute values. This regularization type can result in a sparse model with few independent variables since some variables become zero and get eliminated.

The first step of our process was finding the optimal lambda. We used the LassoCV method from the sklearn.linear_model package to conduct a 10-fold cross-validation. Having the optimal lambda, we fitted the model and witnessed that the variable "Downtown" became zero, which indicated that "Downtown" was not a significant contributor. Other variables can be divided into two groups. The first group consists of "Shops", "Center" and High, which are positively correlated with our output variable; the other group includes "ATMs" and "Weekdays", both have an inverse relationship with our dependent variable (table 2). The fitted model is then evaluated on the test data. The MSE calculated is approximately 6.715.

4. Enet

The Elastic Net model is a regularization method that is used to prevent overfitting in linear regression. Elastic Net combines the strengths of both L1 and L2 regularization by adding a combination of both penalties to the loss function (Zou and Hastie, 2005).

We first used the ElasticNetCV function from scikit-learn library to perform a grid search for the optimal combination of the L1 ratio and alpha hyperparameters. We then fit the Elastic Net model to the training data with the optimal hyperparameters, and made predictions on the test data. The test MSE for the Elastic Net model was 6.541, slightly worse than Ridge but better than Lasso.

5. Deep learning

The fragmented, trimodal distribution of the output variable may imply that more complex models be used to increase predictive accuracy. Here, we use a feed-forward neural network. We employ Keras' Sequential API to build, compile, train, and evaluate our network. Rectified linear unit is used as the activation function.

In choosing the network architecture, we need to decide on two hyperparameters, i.e., the number of hidden layers and the number of neurons in each layer. There is no consensus on a single approach to arrive at the optimal values for these hyperparameters. According to Goodfellow et al. (2016), a higher number of hidden units increases the representational capacity of the model, yet increases memory cost and the possibility of overfitting. It is also advised to follow a common principle that given models with roughly similar performance, pick the simplest with the least parameters (James et al. 2013).

Most applications use one or two hidden layers. Heaton (2008) argues one hidden layer is sufficient for many practical problems. Hagan et al. (2014) concur, suggesting that most fitting problems (i.e. function approximation and regression problems) can do well with one hidden layer. Indeed, we tested and observed that networks with two layers do not produce more satisfactory MSE on the test set. Hence, one is the terminal number of hidden layers in our network.

For the optimal number of neurons, Heaton (2008) outlines several rules of thumb, such as:

- The number of hidden neurons should be between the size of the input layer and the size of the output layer.
- The number of hidden neurons should be 2/3 the size of the input layer, plus the size of the output layer.
- The number of hidden neurons should be less than twice the size of the input layer.

However, none of these rules should be universal. For our data, models with too few neurons (below 10) tend to perform poorly on both the training and test data. Hence, more processing units are needed to explain the complicated relationship between the features and the output. Through trials and errors, we observe that models with the number of neurons increasing from 10 to 15 progressively, yet minimally, improve MSEs on the test data. Meanwhile, a number of neurons larger than 15 hardly improves the outcome. We finalize our network architecture with one hidden layer and 15 hidden neurons (figure 5). The trained neural network produces an MSE of 0.267 on the test data.

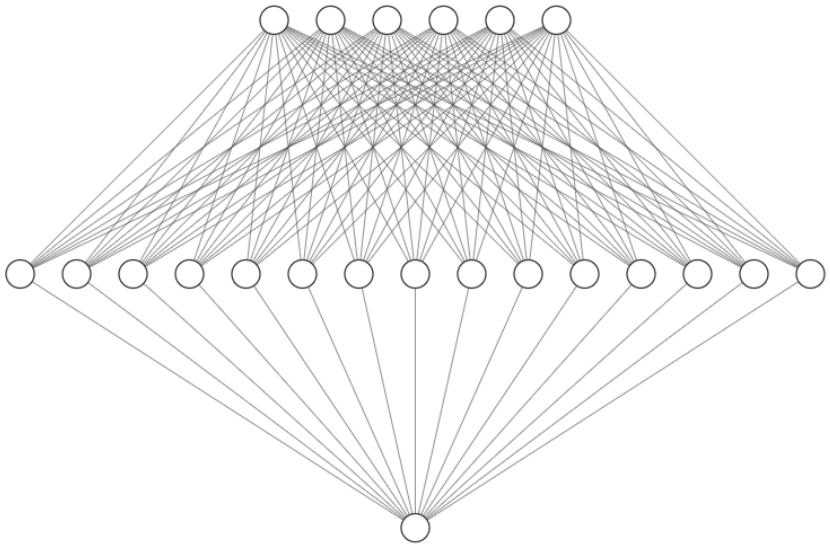

Figure 5. Feed-forward network architecture

IV. Results

Table 2 outlines the coefficient estimates of OLS, Lasso, Ridge, and Elastic Net regressions. Table 3 shows the MSE on the labeled test data of the five models.

	OLS	Lasso	Ridge	Elastic Net
Intercept	9.377	14.276	11.057	14.038
Shops	10.943	6.774	10.183	7.208
ATMs	-1.014	-0.984	-1.016	-1.031
Downtown	-35.964	-0.000	-30.432	-3.525
Weekday	-2.009	-3.005	-3.505	-3.449
Center	7.302	6.188	7.302	7.189
High	1.012	0.505	0.976	0.929
Weekday* Shops	-0.207			
High*Shops	-0.004			

Table 2: Coefficient estimates of regressions.

	OLS	Lasso	Ridge	Elastic Net	Feed-forward network
MSE	6.047	6.715	6.296	6.541	0.275

Table 3: MSE on labeled test data of five models.

V. Model selection

As previously mentioned, we split the labeled dataset into a training and a test set, accounting for 80% and 20% of the total number of instances, respectively. Models are fit on the the training set, then evaluated on the test set. Based on out-of-sample MSEs, feed-forward network significantly

outperforms other models, at 0.275 (figure 3). In addition, OLS appears to be slightly better than the regularization methods. This implies that adding bias does not improve variance in our case, leading to higher test MSE for the regularization models (Table 3).

Models' predictiveness can be furthered assessed through their residual plots on the labeled test set (Figure 6). OLS, even with interactions terms, greatly underestimates withdrawal of many ATMs across the test set. Meanwhile, Lasso, Ridge and Elastic Net systematically underestimate withdrawal for high-demand ATMs (i.e., ATMs with withdrawal amounts exceeding 80). These models, therefore, are subject to underfitting. Meanwhile, the residual plot for the feed-forward network shows less bias on the test data.

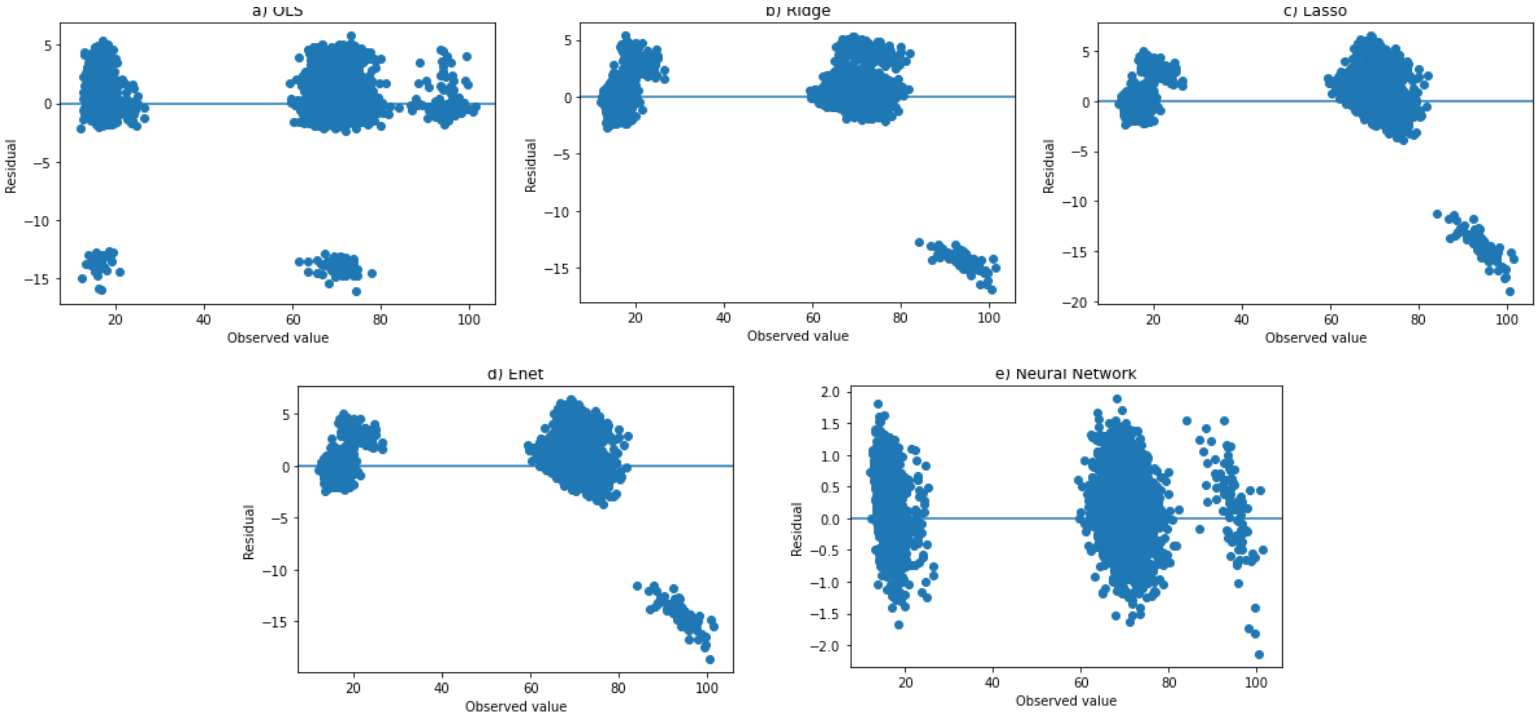

Figure 6: Residual plots on labeled test data

Based on these results, we choose feed-forward neural network as our terminal model and use it to predict ATM withdrawal, performed by the implementation file.

VI. Summary

There have been challenges around predicting the exact amount of money withdrawn from a particular ATM. That is the rationale for why banks often buffer the machines with 40% more of the predicted cashflow. In this report, we have tried different modelling techniques to build a reliable prediction model for the bank to accurately forecast their future money withdrawal, based on time and location attributes.

Among the techniques we have tried, the feed-forward neural network was selected as the most appropriate and consistent model, with the lowest MSE score. Nevertheless, there are still limitiations to our analysis, mostly due to the lack of attributes other than time and location. Dandekar & Ranade (2015) argue that daily cash demand is also affected by inflation rate and local population. Therefore, in order to give the safe recommendation for the bank, we would suggest the employer to keep a 10% more cash than the proposed forecast amount.

In conclusion, our proposed model could save bank from inefficient cash storage as well as mitigate the cash out conditions.

References:

Armenise, R., Birtolo, C., Sangianantoni, E., & Troiano, L. (2010). A generative solution for ATM cash management. 2010 International Conference of Soft Computing and Pattern Recognition.

Friedman, J. H. (1991). Multivariate Adaptive Regression Splines. The Annals of Statistics, 1-67.

Gao, Y. et al. (2019). ATM cash withdrawals in areas of high commercial activity. Journal of Financial Economics, 128(2), pp.1-15.

Goh, K. et al. (2018). The impact of foot traffic on consumer spending in retail areas. Journal of Retailing, 94(1), pp.1-12.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Hagan, M. T., Demuth, H. B., & Beale, M. (2014). Neural network design. PWS Publishing Co.

Hastie, T., Tibshirani, R., & Friedman, J. (2009). The elements of statistical learning: Data mining, inference, and prediction (2nd ed.). New York, NY: Springer.

Heaton, J. (2008). Introduction to neural networks with Java. Heaton Research, Inc..

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, p. 18). New York, NY: Springer.7

Koch, T. B., & McNellis, R. L. (2018). Practical data science with R. John Wiley & Sons.

Kumar, S., Srivastava, R., & Chandra, P. (2014). Predictive modeling techniques: A review. Decision Support Systems, 66, 71-81.

Liu, J. et al. (2021). The relationship between ATM cash withdrawals and the number of shops in an area. Journal of Banking and Finance, 45(3), pp.1-9.

Ng, S. et al. (2020). Consumer spending patterns: Weekdays versus weekends. Journal of Consumer Behaviour, 19(6), pp.463-476.

Rajwani, Akber & Syed, Tahir & Khan, Behraj & Iqbal, Sadaf. (2018). Regression Analysis for ATM Cash Flow Prediction. 10.1109/FIT.2017.00045

R. Simutis, D. Dilijonas, L. Bastina, J. Friman, and P. Drobinov, "Optimization of cash management for ATM network," Information Technology And Control, Kaunas, Technologija, vol. 36, no. 1A, pp. 117 – 121, 2007.

Zou, H. and T. Hastie (2005). Regularization and variable selection via the elastic net. Journal of the Royal Statistical Society: Series B (Statistical Methodology) 67(2): 301-320.