ECON3203: Economic Theory & Methods

GROUP REPORT: ATM NETWORK

GROUP B:
Arya Ahluwalia, z5480544
Amal Khan, z5477807
Fiza Shafeeque, z5419886
Kiran Singh, z5488154
Zach Wan, z5417138

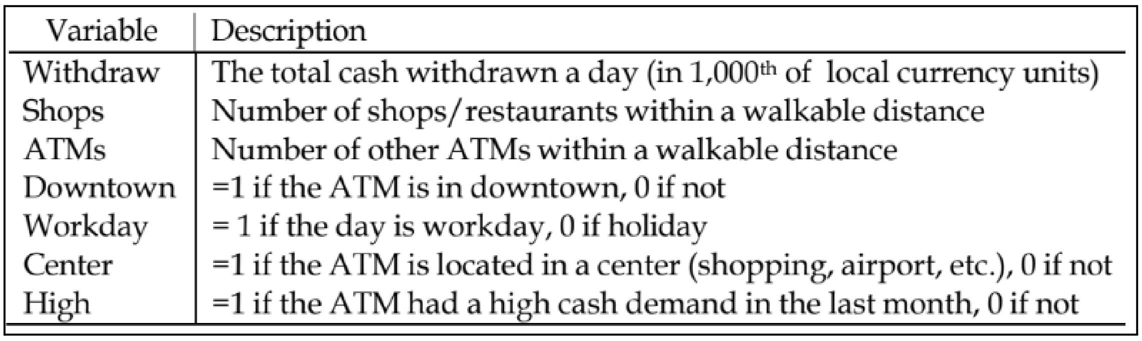
TABLE OF CONTENTS

1. Introduction

In today's fast-paced world of technology, cashless transactions are becoming more common due to a rise in digital payment platforms and mobile banking. The increasing trend towards cashless transactions presents a challenge for ATMs to predict demand for withdrawals and hence optimise cash management.

The aim of this project is to develop a predictive model that examines the relationship between the response variable Withdraw and the predictor variables Shops, ATMs, Downtown, Workday, Center, and High. This model will allow financial institutions to predict cash demand and make informed decisions about its ATM network, such as where to place ATMs and how much cash each location would need.

Figure 1.1: Summary of Response & Covariate Variables.



Variable	Description
Withdraw	The total cash withdrawn a day (in 1,000th of local currency units)
Shops	Number of shops/restaurants within a walkable distance
ATMs	Number of other ATMs within a walkable distance
Downtown	=1 if the ATM is in downtown, 0 if not
Workday	= 1 if the day is workday, 0 if holiday
Center	=1 if the ATM is located in a center (shopping, airport, etc.), 0 if not
High	=1 if the ATM had a high cash demand in the last month, 0 if not

The stages of this included exploratory data analysis (EDA) and modelling. Our findings from the EDA phase informed our choices in model development where various models were developed. These included linear regression models, such as OLS, Lasso, Ridge and Elastic net, tree based models, such as Random Forest, and other models such as Neural Networks and k-Nearest Neighbours (kNN) Regressor. Each model was assessed based on its predictive performance (using Test MSE as the primary metric), computational efficiency and interpretability. Overall, our OLS model excelled across these criteria when compared with other models, having a test mean squared error (MSE) of **0.2489**.

2. Exploratory Data Analysis (EDA)

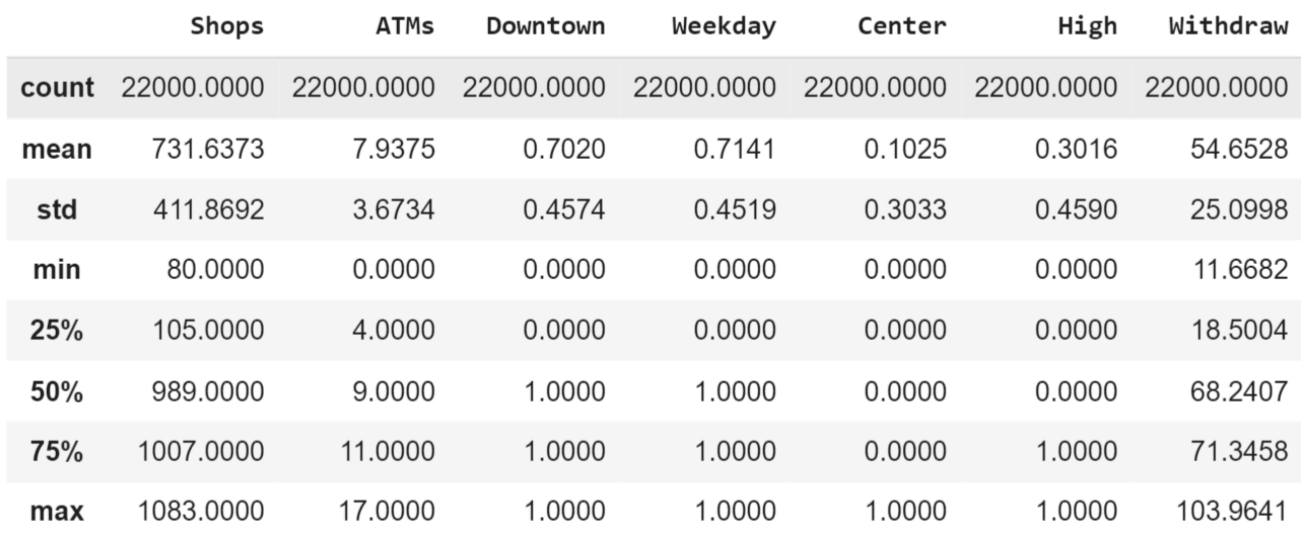
Understanding the Dataset + Data Cleaning

The ATM_sample.csv dataset used for this analysis consists of 22,000 observations and 7 variables (including the response variable). Each variable represents an important feature for predicting the response variable Withdraw. Predictor variables include only numerical features so no encoding or dummy creation was necessary. Initial data inspection revealed that the dataset contains no missing values in any field, so it was concluded that data cleaning was not required.

Summary Statistics

The summary statistics for numerical variables provide insights into the distribution of the data. This step helps us in identifying any variables that may require normalisation or transformation, as well as any outliers.

Figure 2.1: Summary Statistics Table of the ATM_sample Data



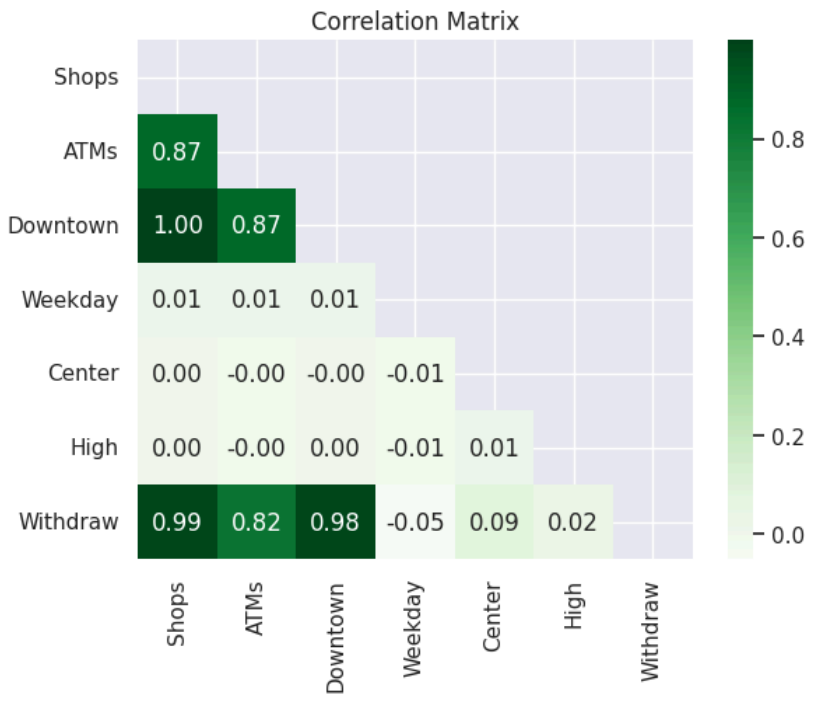
	Shops	ATMs	Downtown	Weekday	Center	High	Withdraw
count	22000.0000	22000.0000	22000.0000	22000.0000	22000.0000	22000.0000	22000.0000
mean	731.6373	7.9375	0.7020	0.7141	0.1025	0.3016	54.6528
std	411.8692	3.6734	0.4574	0.4519	0.3033	0.4590	25.0998
min	80.0000	0.0000	0.0000	0.0000	0.0000	0.0000	11.6682
25%	105.0000	4.0000	0.0000	0.0000	0.0000	0.0000	18.5004
50%	989.0000	9.0000	1.0000	1.0000	0.0000	0.0000	68.2407
75%	1007.0000	11.0000	1.0000	1.0000	0.0000	1.0000	71.3458
max	1083.0000	17.0000	1.0000	1.0000	1.0000	1.0000	103.9641

The standard deviations of Shops and Withdraw are high compared to all other features, suggesting high variability or potential outliers. To explore this issue further we looked into the correlation of features with each other and the distribution of each feature. The distribution was also used to detect any outliers within the features with high standard deviations

Correlation Matrix

A correlation matrix was generated to explore the relationships between the variables. Figure 2.2 visualises the correlations and, in extension, the multicollinearity amongst the features.

Figure 2.2: Correlation Heatmap of All Variables



The variables Downtown and Shops, Withdraw and Shops, Withdraw and Downtown were highly correlated, suggesting the need to either drop one of these variables or create interaction terms to avoid multicollinearity. We opted to create interaction terms to improve model flexibility and avoid misspecification while preserving important information.

Distribution of Features

To understand how each numerical variable is distributed and the high variability of Shops and Withdraw, the distributions of each feature were plotted using the seaborn library's histplot() function.

Figure 2.3: Distribution of Shops

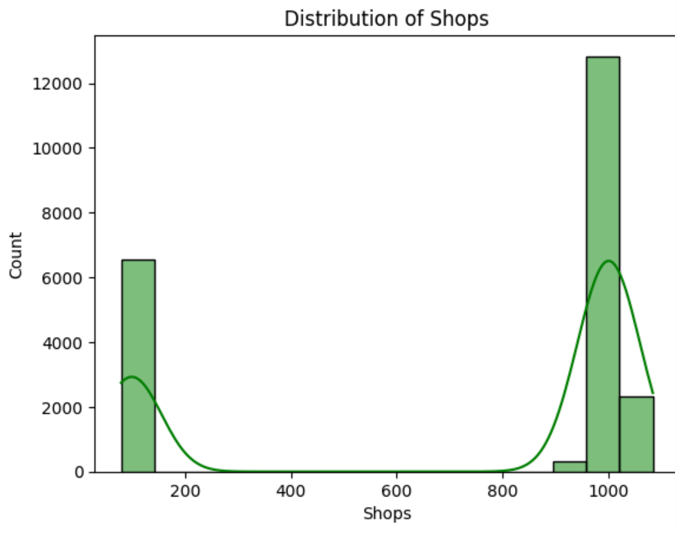
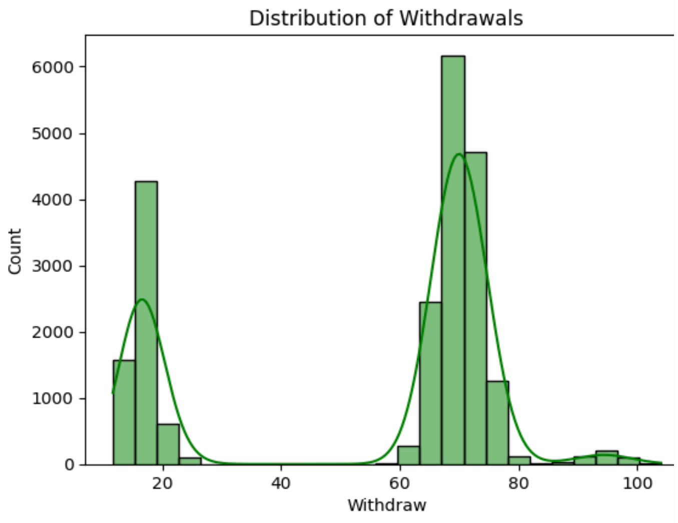


Figure 2.4: Distribution of Withdrawals



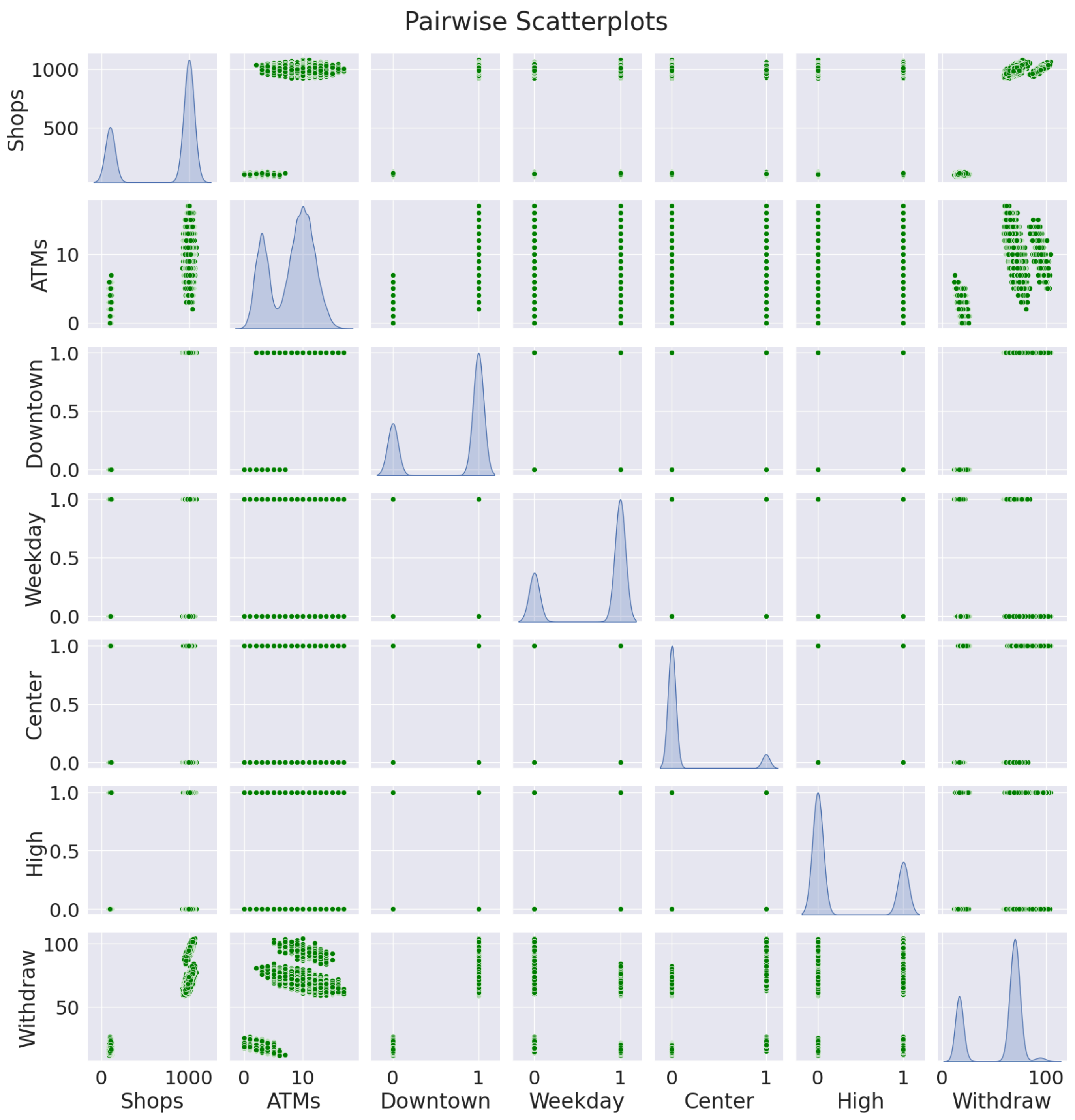
The distribution of Shops (Figure 2.3) has a skewness of **-0.8780** and Withdraw (Figure 2.4) has a skewness of **-0.7724**. The bimodal nature of these two variables is explained by their high correlation with Downtown, as shown in Figure 2.2. By inspection, we can also see that there are no outliers within the features with high initial standard deviations. Hence no rows were needed to be removed.

We examined the correlations with Downtown further by splitting the dataset into Withdraw < 40 (first mode) and Withdraw >= 40 (second mode). This showed that the data was clearly separated into two clusters, as the first mode was entirely from ATMs not situated downtown (Downtown = 0) and the second mode was entirely from ATMs downtown (Downtown = 1).

Relationship Between Variables

To further understand the relationship between variables we looked at scatterplots of pairwise variable relationships. We produced the following graph with the seaborn library's pairplot() function. As shown below, there are no linear or non-linear relationships that are explicitly shown in the plots. This further validates that there are no explicit outliers in our dataset. However, we do still see bimodality across the diagonal - we kept this in mind when choosing our model, as some models address bimodality and multicollinearity better than others.

Figure 2.5: Pairwise Plots of All Variables



By conducting thorough EDA, we have gained a deeper understanding of the data's structure, the relationships between variables, and potential issues such as bimodality, multicollinearity and skewness. These insights guided the selection of appropriate predictive models and preprocessing steps.

3. Models & Methods

We explored four different categories of regression models to predict ATM withdrawals: Linear Regression Models (including OLS, Lasso, Ridge, and Elastic Net), a Polynomial Regression Model, Tree-Based Regression Models (such as Random Forest), and others (including Neural Networks and kNN). Each model was selected for its ability to capture different patterns in the data.

To ensure reproducibility of our results and consistency across models, a random seed of 42 was set. Moreover, all models were allocated 75% of the dataset as a training set, and the remaining 25% as a test set.

Tree Based Regression

As the EDA revealed that the target variable Withdraw is bimodal, we decided to start with tree based models, as we hypothesised that they would be able to split the two modes well based on the Downtown variable.

Decision tree:

It was inferred from the EDA that Downtown could effectively be the first split, we explored an iterative method of optimising the max depth value, one that minimised MSE while retaining simplicity. The optimal depth was found to be 12, and the tree generated from this (visualised using sklearn.tree's export_text() and plot_tree() functions[7]) gave a test MSE of **0.3410**. However, the resulting tree was still overly complex (refer to Appendix 6.1), and likely overfit. We verified this by obtaining a training MSE of 0.1833, which is much lower than the test MSE and indicates potential overfitting. We attributed this to the high multicollinearity present in the data[1], so we needed a model which wouldn't be affected by multicollinearity. This motivated our experimentation with the Random Forest Model[2].

Random Forest:

We performed GridSearchCV()[8] from the sklearn.model_selection library and obtained the optimal parameters to be max_depth = 20, min_samples_split = 5, and n_estimators = 300. We chose not to provide more parameters for GridSearchCV to optimise due to the higher processing times this would entail.

We fit this optimal model, and received a test MSE of **0.3067**. While this was an improvement from the decision tree, we hoped to do even better, particularly with a linear model, as linear regression models tend to be simpler and more interpretable.

Interaction Terms

In response to the highly correlated variables and skewness discovered during EDA, we designed interaction terms[3-4] to capture the complex nature of these interconnected features. We used the itertools library to create interaction terms consisting of every possible combination, this amounted to a total of 63 features. On examination of the new correlation matrix between these interaction terms (refer to Appendix 6.2), high multicollinearity had carried into the newly created features.

When including these interaction terms in an experiment with a naive Random Forest Regression, the feature with greatest importance was Shops-ATMs (0.1943). This indicates that the relationship between Withdraw and Shops or ATMs isn't purely additive, the combined effect of features may capture unique information which isn't evident when they are considered separately. Thus, we continued modelling with interaction terms to capture these relationships.

However, we recognised that this was an excessive amount of features and in future modelling we should look to select a subset of these.

Linear Regression Models

We first performed standardisation on the data using the sklearn.preprocessing library's StandardScaler() function. Then, using the sklearn.linear_model library's LassoCV(), RidgeCV(), ElasticCV() and LinearRegression() functions, we tested different linear regression models. For all these models, we used a train-test split of 75-25 on the dataframe with interaction terms.

LASSO:

Since we had more than 60 interaction terms with high multicollinearity, we tried LASSO regression first. This was due to its ability to set feature coefficients to 0 and essentially perform feature selection, whilst simultaneously dealing with multicollinearity. To obtain the optimal value of the regularisation parameter, we performed cross validation with 5 folds, and obtained the value 0.0248.

The LASSO regression model then gave us a training MSE of 0.7889 and test MSE of **0.8155**. This was worse performance than Random Forest, so we eliminated this model.

Ridge:

Ridge regression is also known to deal with multicollinearity very well, so we tried this next. Once again cross validation with 5 folds was performed to get the optimal regularisation parameter value, which was obtained to be 0.0043.

The Ridge regression model then gave a train MSE of 0.2474 and test MSE of **0.2499**, which was our best MSE so far.

Elastic Net:

Due to Ridge Regression's positive performance, we attempted Elastic Net since it balances the L1 and L2 regularisation, i.e. balances the work of LASSO and Ridge. Once again, we performed cross validation with 5 folds to obtain the L1 ratio and the regularisation parameter.

This gave us lambda as 0.0248 and an L1 ratio of 1.0. These values meant that Elastic Net was behaving identically to the LASSO regression model. Unsurprisingly, the training and test MSEs were also identical to the LASSO model, being 0.7889 and **0.8155** respectively. Hence, this model was also eliminated.

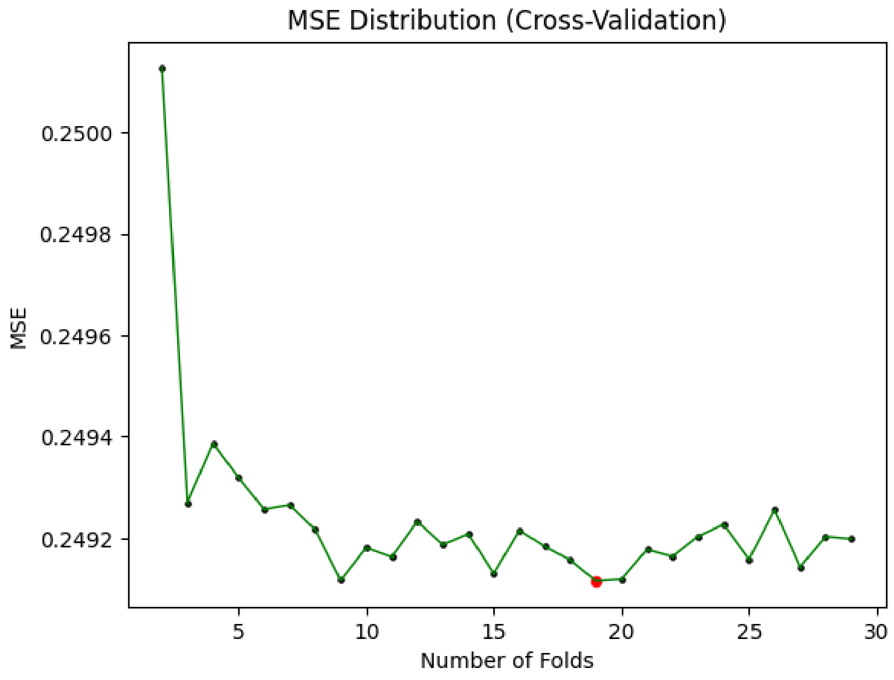
OLS:

We fit a simple OLS regression model on all interaction features, including the original predictors. This gave us a training MSE of 0.2473 and test MSE of **0.2500**. Interestingly, this was only 0.0001 higher than the MSE from Ridge regression. We recognised that this model could be further improved, so we opted to continue working with and fine tuning it.

We utilised cross-validation to reduce the risk of overfitting and leverage all available data, as this would allow each fold to be used in both training and testing. Optimising the number of folds was crucial as the optimum would ensure a balance between the bias and variance of our predictions. A lower number of folds would mean a smaller training set and potentially introduce bias, whereas a higher number of folds would reduce the size of the test set, increasing variance by making the model more sensitive to fluctuations in the data.

Figure 3.1 plots the number of folds against the obtained test MSE. A minimum is achieved at 19 folds, as highlighted. Thus, 19-fold cross-validation was applied to the OLS model. This reduced the test MSE to **0.2491**.

Figure 3.1: MSE Plot with Optimal Number of Cross-Validation Folds highlighted

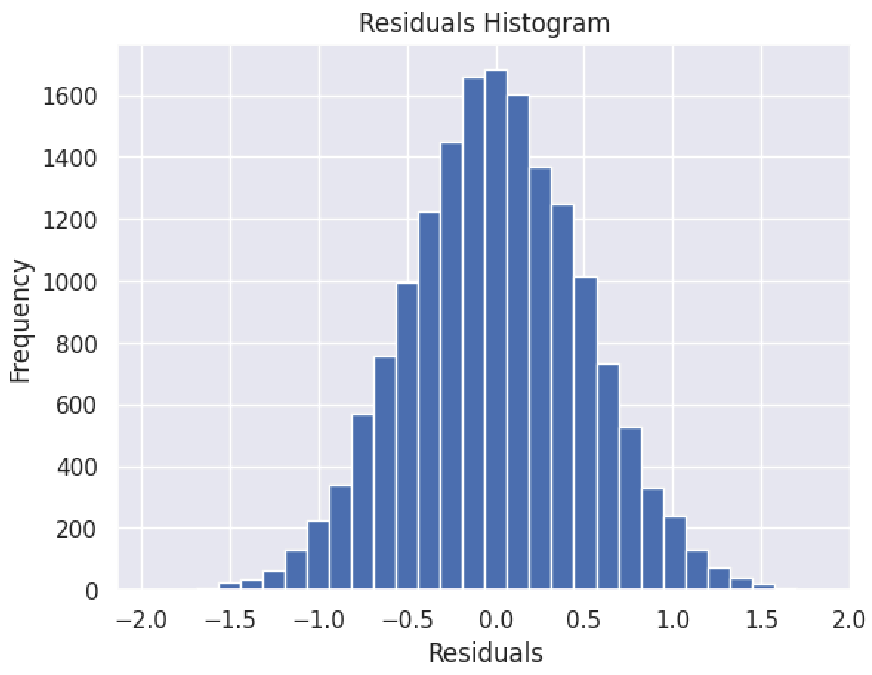


To further improve the model, we extracted the p-value of all 63 features using the ols_model.p_values() function from the statsmodels.api library. Due to the abundance of features, we set the significance level to 0.001% and any terms with a p-value greater than this were dropped from the model. This cut 24 terms, leaving 39 features and an improved test MSE of **0.2489**.

As this was our best overall test MSE, we selected this as our final model.
Withdraw ~ 1 + Shops + ATMs + … + Shops-ATMs-Downtown-Weekday-Center + Shops-ATMs-Downtown-Weekday-High (see Appendix 6.3 for full OLS model equation including intercept and coefficients).

To ensure that this model captured the patterns in the unseen data well and checked off the normally distributed error assumptions, we plotted a histogram of the residuals for the fitted model.

Figure 3.2: Histogram plot of residuals of the chosen fitted OLS model

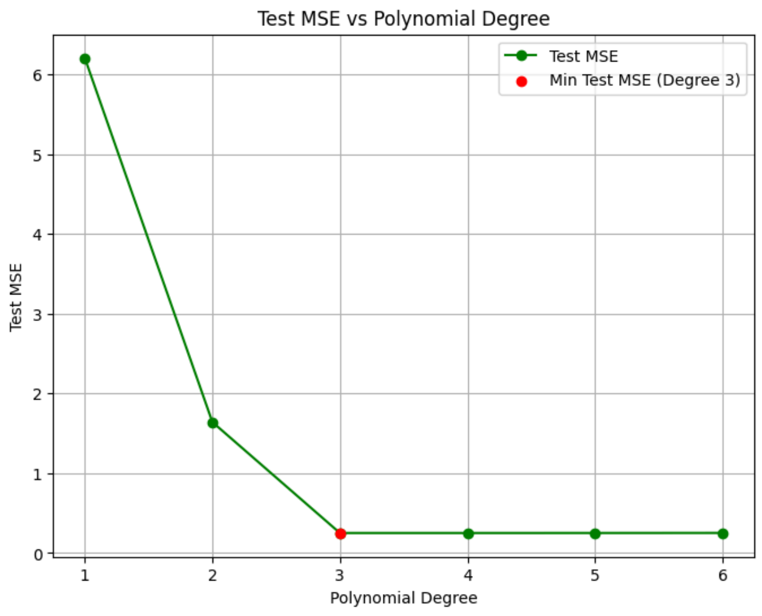


This showed us that the residuals were normally distributed, indicating that the model was a good fit for the data and likely unbiased. Thus, the concerns of skewness had been addressed by this model and its risks mitigated.

Polynomial Regression

We also explored polynomial regression[5] to examine the nonlinear relationships between predictor variables and the target variable. For this, the PolynomialFeatures() function from the sklearn.preprocessing library was utilised. Since polynomial regression is flexible and can model various degrees of curvature, we tested for the optimal degree and plotted the results in an MSE plot as depicted in Figure 3.3.

Figure 3.3: MSE Plot against Degree of Polynomial



The clear elbow at degree 3 indicated that this is the optimal degree of the polynomial regression as it would capture the underlying patterns most effectively without overfitting. This model provided a test MSE of **0.2492** with the equation: Withdraw ~ 19.0948 + 0.0015 * 1 + 0.0084 * Shops + -0.8772 * ATMs + -90.5111 * Downtown ... (refer to Appendix 6.4 for full equation).

We attempted to further improve the model in a similar fashion to the OLS model, using cross-validation at the previously found optimum of 19 folds. The results were as follows:

Train MSE	0.2473
Test MSE	**0.2485**

As such, polynomial regression yielded the lowest test MSE, ~0.0004 lower than the OLS Model with 19 folds of Cross-Validation. However, it is important to take into account the tradeoff between interpretability and accuracy. Although the polynomial regression may provide a slightly better MSE, it had an increased number of parameters (84 compared to the OLS model's 39), leading to longer training times and thus lower computational efficiency. Moreover, higher-degree polynomial terms (e.g. squared or cubed terms) don't indicate a straightforward positive or negative relationship between the predictor and the response variable, so the polynomial regression compromises on interpretability. Thus, we decided to forfeit the ~0.0004 improvement of MSE as the interpretability, computational efficiency and chances of overfitting are all more favourable in the OLS model.

Other Models

Neural Networks:

Due to Neural Networks' capabilities to pick up complex patterns, we decided to investigate the possibility of using it to predict withdrawals. Using a combination of interaction and polynomial terms, we used the TensorFlow library to build our models. To optimise our Neural Network model we had to find the right permutation of hyperparameters that would lead to a low test MSE without overfitting the data.

In the end, we found that using 1 layer with 150 Neurons with the ReLU (Rectified Linear Unit) activation function had the best results. Furthermore, over many iterations, we found a learning rate of 0.01 and using the reduced learning function, by a factor of 0.9, improved our results. The difference between optimisers wasn't large, but generally the 'Adam' optimiser performed better over more iterations. In our final Neural Network model we had 850 epochs (iterations) with a batch size of 20. This model returned a test MSE score of **0.2531**.

Even though this was a really strong model, with great predictive capabilities, it was not considered in our final model selection due to the inherent lack of interpretability of Neural Network models. It is worth mentioning that this Neural Network is more likely than not, a suboptimal model which could be improved upon. However, the improved test MSE would likely not be that much lower than our best regression models. As such, we opted to halt the development of our Neural Network models.

Quick caution: The code for our Neural Network implementation takes around 30 minutes to run.

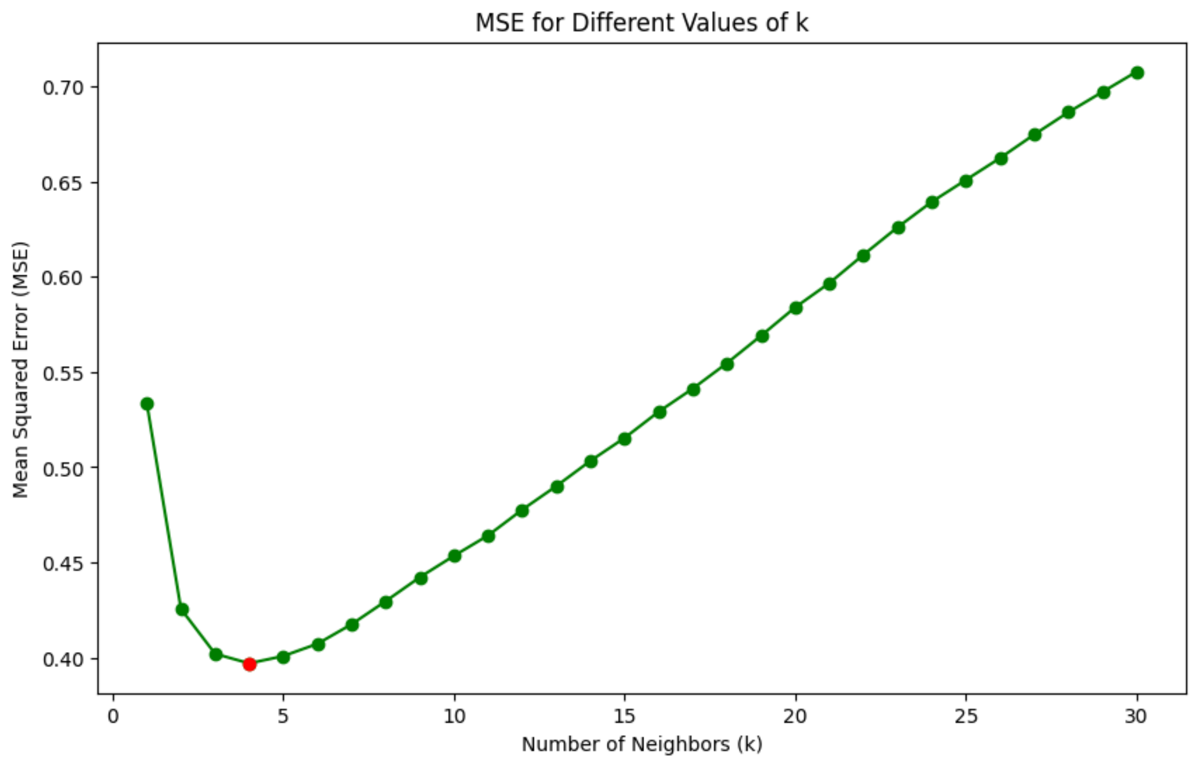
K Nearest Neighbours (KNN):

Due to the clustered nature of the data we had observed in our EDA, a more obscure choice of model was a K-Nearest Neighbours Regression model[9].

Once again we used both the interaction terms and polynomial terms to model with. At first, we standardised all the data, before fitting for an optimal k value, which we found to be 5 (as shown in the figure below). However, after running further models, we found that using a Min-Max Scaler()[10] performed better. And so the final test MSE of our kNN model was **0.3753** using 5 nearest neighbours and the Min-Max Scaler.

This model performed poorly, most likely due to the curse of dimensionality impairing concepts of distance in this model. Similar to Neural Networks, kNN models have poor levels of interpretability in comparison to regression models. In the end, the model could likely be improved by finding an optimal subset of predictors, however, due to the models already poor performance, we felt no reason to investigate it further.

Figure 3.4 MSE Plot For Different Values of K



4. Discussion & Conclusion

The final model that we chose was our OLS model, trained through 19-fold cross-validation and containing 39 of the 63 available interaction terms. Our final test MSE with this model is **0.2489** .

Since we performed OLS on only variables with low p-values, and the MSE is already lower than ridge regression which reduces the effects of multicollinearity, we did not do any further exploration on how multicollinearity affected the result of our final model. We recognised that our final model did not in fact remove all multicollinearity, although the inherent transformation to the data by creating interaction terms and then eliminating high p-value terms would have addressed this issue.

In the real world, an asymmetric loss function would be recommended to be used since an under-prediction causes more harm (e.g. ATM running out of cash, transaction errors, unhappy customers, etc.) than an over-prediction. A possible approach for this would be to utilise models with gradient boosting. Penalties could be applied differently to our model in different scenarios using class weighting - either ATMs having too much money in them or ATMs running out of money (the latter representing the more problematic scenario hence higher weights).

However, we did not utilise this approach because this is outside the scope of our project brief, and because they are better suited to data with more non-linear relationships[6]. OLS takes care of linear relationships much better, and is also much more interpretable than gradient boosting algorithms.

Additionally, the purpose of the project was purely to find the best predictive model. Therefore, making alterations to the model to make considerations for over-counting and under-counting would harm our results.

Our chosen OLS model is highly interpretable and efficient by design, each coefficient representing the predictors impact on the response variable Withdraw. For this reason, we decided to forego the ~0.0004 improvement in test MSE offered by polynomial regression to ensure our model would be understandable for our client, who are specialists in banking. Furthermore, OLS requires less computational processing, thus it would be better suited to large quantities of data which a bank likely has.

In conclusion, our OLS model balances predictive accuracy, interpretability and computational efficiency, enabling suitable predictions regarding the demand for cash at ATMs.

Table 4.1: Train and Test MSE of Models Ranked From Smallest to Largest

Model	Test MSE	Train MSE
Polynomial regression	0.2485	0.2473
OLS	**0.2489**	**0.2476**
Ridge	0.2499	0.2474
Neural Network	0.2534	0.2508
Random Forest with original features	0.3067	0.2011
Decision Trees	0.3410	0.1833
kNN	0.3753	0.3969
LASSO	0.8155	0.7889
Elastic Net	0.8155	0.7889

Quick caution: The code for our Neural Network implementation takes around 30 minutes to run

5. References

A bulk of the content in this report and in the submitted .ipynb notebook are attributed to the ECON3203 Tutorial Solutions in Jupyter Notebook files and the ECON3203 Lecture Slides. Additional resources are listed below.

[1] Stack Exchange, 2018. *Multicollinearity in Decision Tree.* [online] Available at: https://datascience.stackexchange.com/questions/31402/multicollinearity-in-decision-tree [Accessed 24 October 2024].

[2] Raj, S., 2019. *Effects of Multi-Collinearity in Logistic Regression, SVM & RF*. Medium. [online] Available at: https://medium.com/@raj5287/effects-of-multi-collinearity-in-logistic-regression-svm-rf-af6766d91f1b [Accessed 10 October 2024].

[3] NVIDIA, 2022. *A Comprehensive Guide to Interaction Terms in Linear Regression*. Developer. [online] Available at: https://developer.nvidia.com/blog/a-comprehensive-guide-to-interaction-terms-in-linear-regression/ [Accessed 24 October 2024].

[4] Stat Trek, n.d. *Interaction in Multiple Regression*. [online] Available at: https://stattrek.com/multiple-regression/interaction [Accessed 24 October 2024].

[5] Sápi, P., 2021. *Polynomial Regression in Python Using Scikit-Learn*. Data36. [online] Available at: https://data36.com/polynomial-regression-python-scikit-learn/ [Accessed 31 October 2024].

[6] Abdo Alabdo (Umea University), 2024. [online] Available at: https://umu.diva-portal.org/smash/get/diva2:1896701/FULLTEXT01.pdf [Accessed 9 November 2024]

Python Libraries (external to what has already been covered in the course)

[7] scikit-learn, n.d. *sklearn.tree*. [online] Available at: https://scikit-learn.org/1.5/api/sklearn.tree.html [Accessed 3 November 2024].

[8] scikit-learn, n.d. *sklearn.model_selection.GridSearchCV*. [online] Available at: https://scikit-learn.org/dev/modules/generated/sklearn.model_selection.GridSearchCV.html [Accessed 3 November 2024].

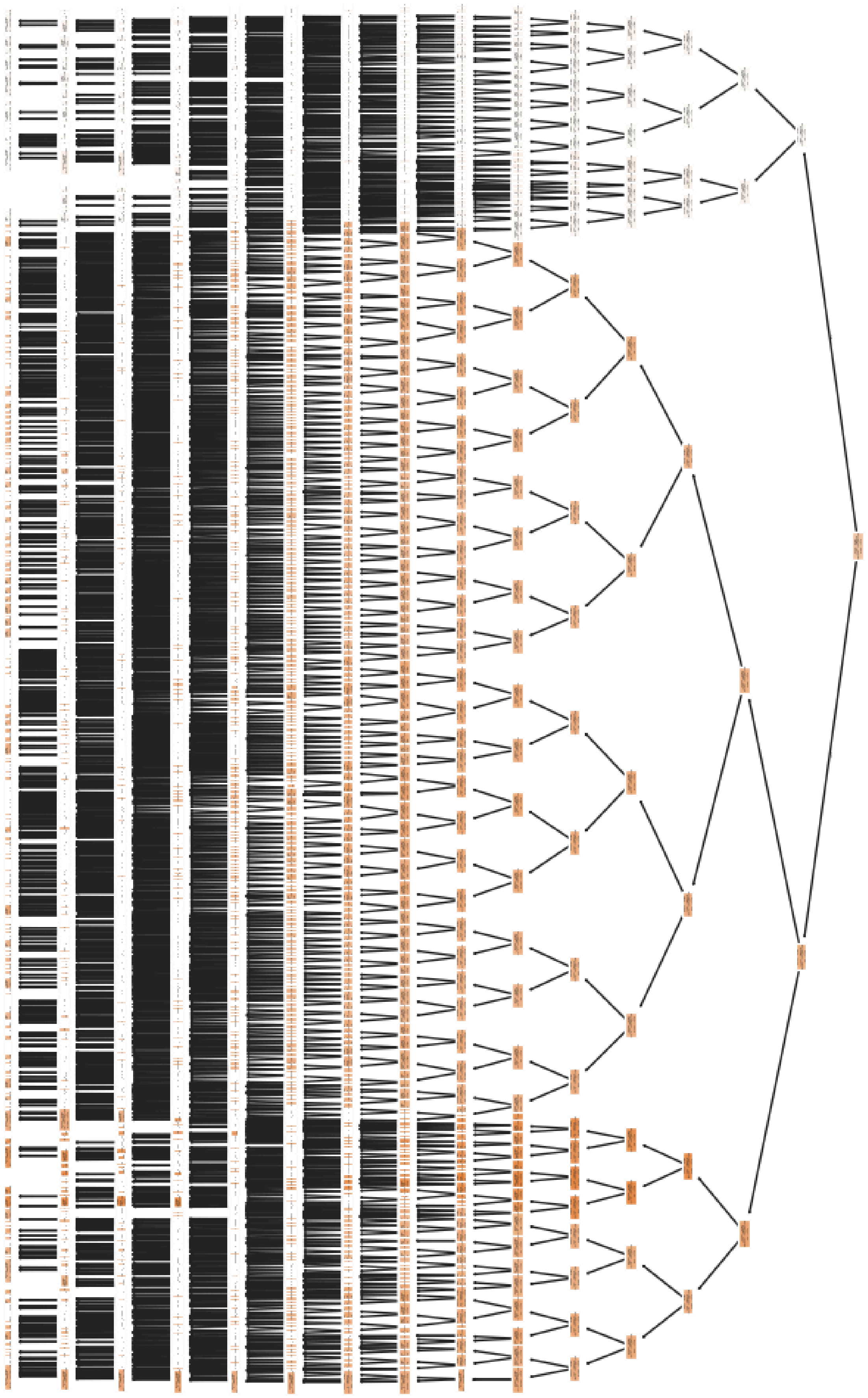
[9] scikit-learn, n.d. *sklearn.neighbors.KNeighborsRegressor*. [online] Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.neighbors.KNeighborsRegressor.html [Accessed 7 November 2024].

[10] scikit-learn, n.d. *sklearn.preprocessing.MinMaxScaler*. [online] Available at: https://scikit-learn.org/1.5/modules/generated/sklearn.preprocessing.MinMaxScaler.html [Accessed 3 November 2024].

6. Appendix

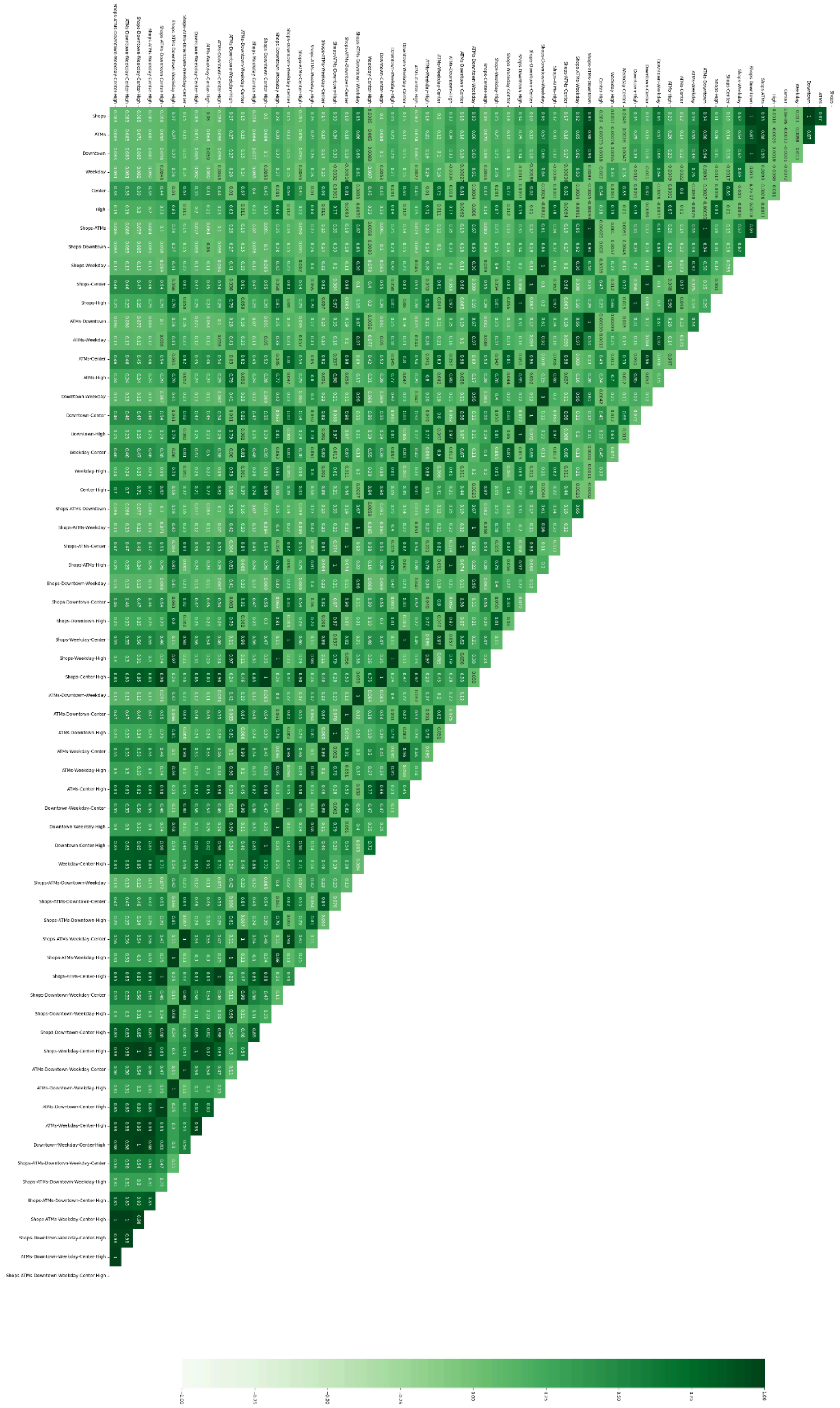
Appendix 6.1: Decision Tree Diagram

Please excuse the lack of readability, the diagram is as large as this page will fit.



Appendix 6.2: Interaction Terms Correlation Matrix

Please excuse the lack of readability, the diagram is as large as this page will fit.



Appendix 6.3: OLS Regression Equation (Full Form)

OLS formula: Withdraw = 19.4937 + Shops * 0.0108 + ATMs * -1.3233 + Downtown * -48.8852 + Weekday * -3.5535 + Center * 3.787 + High * -0.6493 + Shops-ATMs * 0.0031 + Shops-Downtown * 0.0996 + Shops-Weekday * 0.0147 + Shops-Center * 0.0107 + Shops-High * 0.0158 + ATMs-Downtown * 0.3792 + ATMs-Weekday * 0.5858 + ATMs-Center * -0.2238 + ATMs-High * 0.3198 + Weekday-Center * 0.1268 + Weekday-High * 2.4807 + Shops-ATMs-Downtown * -0.0031 + Shops-ATMs-Weekday * -0.0056 + Shops-ATMs-Center * 0.0025 + Shops-ATMs-High * -0.003 + Shops-Downtown-Weekday * -0.0131 + Shops-Downtown-Center * 0.0085 + Shops-Downtown-High * -0.0138 + Shops-Weekday-Center * -0.02 + Shops-Weekday-High * -0.0242 + ATMs-Downtown-Weekday * -0.3849 + ATMs-Downtown-High * -0.2591 + ATMs-Weekday-Center * 0.0898 + ATMs-Weekday-High * -0.5814 + Shops-ATMs-Downtown-Weekday * 0.0054 + Shops-ATMs-Downtown-Center * -0.0022 + Shops-ATMs-Downtown-High * 0.0029 + Shops-ATMs-Weekday-Center * -0.0011 + Shops-ATMs-Weekday-High * 0.0056 + Shops-Downtown-Weekday-High * 0.0214 + ATMs-Downtown-Weekday-High * 0.3807 + Shops-ATMs-Downtown-Weekday-Center * 0.001 + Shops-ATMs-Downtown-Weekday-High * -0.0054

Appendix 6.4: Degree 3 Polynomial Regression Equation (Full Form)

Withdraw ~ 19.0948 + 0.0015 * 1 + 0.0084 * Shops + -0.8772 * ATMs + -90.5111 * Downtown + -0.5301 * Weekday + 0.9842 * Center + 0.3508 * High + 0.0000 * Shops^2 + -0.0018 * Shops ATMs + 0.3697 * Shops Downtown + -0.0024 * Shops Weekday + 0.0107 * Shops Center + -0.0017 * Shops High + -0.0026 * ATMs^2 + 2.9600 * ATMs Downtown + -0.0193 * ATMs Weekday + 0.0105 * ATMs Center + 0.0060 * ATMs High + -90.5103 * Downtown^2 + 2.2885 * Downtown Weekday + -8.3239 * Downtown Center + 8.4416 * Downtown High + -0.5303 * Weekday^2 + 0.0699 * Weekday Center + 0.0003 * Weekday High + 0.9838 * Center^2 + 0.0133 * Center High + 0.3513 * High^2 + 0.0000 * Shops^3 + 0.0000 * Shops^2 ATMs + -0.0007 * Shops^2 Downtown + 0.0000 * Shops^2 Weekday + -0.0000 * Shops^2 Center + 0.0000 * Shops^2 High + 0.0000 * Shops ATMs^2 + -0.0091 * Shops ATMs Downtown + 0.0005 * Shops ATMs Weekday + -0.0001 * Shops ATMs Center + 0.0000 * Shops ATMs High + 0.3697 * Shops Downtown^2 + -0.0038 * Shops Downtown Weekday + 0.0467 * Shops Downtown Center + -0.0483 * Shops Downtown High + -0.0024 * Shops Weekday^2 + -0.0208 * Shops Weekday Center + -0.0006 * Shops Weekday High + 0.0107 * Shops Center^2 + 0.0002 * Shops Center High + -0.0017 * Shops High^2 + -0.0000 * ATMs^3 + -0.0290 * ATMs^2 Downtown + 0.0004 * ATMs^2 Weekday + 0.0037 * ATMs^2 Center + -0.0025 * ATMs^2 High + 2.9600 * ATMs Downtown^2 + -0.4637 * ATMs Downtown Weekday + 0.0434 * ATMs Downtown Center + -0.0076 * ATMs Downtown High + -0.0193 * ATMs Weekday^2 + -0.0174 * ATMs Weekday Center + 0.0224 * ATMs Weekday High + 0.0105 * ATMs Center^2 + -0.0262 * ATMs Center High + 0.0060 * ATMs High^2 + -90.5103 * Downtown^3 + 2.2885 * Downtown^2 Weekday + -8.3239 * Downtown^2 Center + 8.4416 * Downtown^2 High + 2.2885 * Downtown Weekday^2 + 0.7620 * Downtown Weekday Center + 0.3181 * Downtown Weekday High + -8.3239 * Downtown Center^2 + 0.0158 * Downtown Center High + 8.4416 * Downtown High^2 + -0.5303 * Weekday^3 + 0.0699 * Weekday^2 Center + 0.0003 * Weekday^2 High + 0.0699 * Weekday Center^2 + -0.0243 * Weekday Center High + 0.0003 * Weekday High^2 + 0.9838 * Center^3 + 0.0133 * Center^2 High + 0.0133 * Center High^2 + 0.3513 * High^3