

**To: GSBA 524: Managerial Statistics**

**From: Saif Rehman**

**Date: November 9, 2018**

**Subject: Options Pricing**

## Options Pricing

Pricing options is a very complicated procedure because the price of the option depends on the underlying stock, the variability of the stock, and the time to maturity of the option. An option is only valuable if the strike price of the option is below the future asset value at the time of exercising the option, referred to as an option being “in the money”. However, since the option is sold before it can be exercised, it is a probabilistic approach to pricing an option. The Black-Scholes formula was created to price such options and works remarkably well in pricing options in practice.

However, the purpose of this report is to determine whether machine learning models can be constructed to predict option prices with a greater accuracy than the Black-Scholes formula. In this report, I will be using a training dataset that consists of pricing information for 2,800 separate options to create a machine learning algorithm that is able to predict the value of the option based on the characteristics of the option: the strike price (K), the current value of the underlying asset (S), the annual interest rate (r), and the time to maturity (tau). Furthermore, I will create another machine learning model to predict whether the Black-Scholes formula will overestimate the current option price or underestimate the current option price.

Since I was given a training dataset of 2,800 options, our first step was to split the dataset into a training and a validation dataset using a randomized 70-30 split so that I can accurately test each model’s performance on data that it hasn’t seen before. I also performed some preprocessing steps on the data by testing if there are any outliers, missing data, or any abnormalities in the dataset. I decided to create a new feature called “diff” which is the difference between the current value of the underlying asset and the strike price of the option, effectively capturing the information of how far away the option will be from being in the money. I tested all of our models both with the added diff variable and with the original S and K variables and found that the diff variable leads to better generalizations out-of-sample.

For predicting the current value of the option, since that is a regression problem, I utilized a variety of different models. I tested multiple linear regression, support vector machines (SVM), random forest, and XGBoost to see which models performed the best on the validation dataset. To accurately test our performance, I used performance measurements of root-mean squared error (RMSE) and mean absolute percentage error (MAPE) since the variable that I am predicting is a continuous variable. I decided to use XGBoost as our final model to predict the current value of the option since that is the one that performed the best on the validation dataset.

Similarly, I also tested a variety of models to predict whether Black-Scholes would underestimate or overestimate the current option value which is a binary classification problem. For this problem, I tested binomial logistic regression, decision trees, support vector machines, kNN, random forest, and XGBoost. For this classification problem, I used a general accuracy measure which is calculated from dividing the total number of correct predictions over the total number of rows since we are creating a model to maximize the overall accuracy of the model and not concerned with the general specificity and sensitivity of the model performance. After testing all of the various models, I decided to use support vector machines to submit our final predictions of the model since SVM’s performed relatively well and had less risk of overfitting the data as compared with some of the other models used.

Options Pricing Report

# Overview

The objective of this project is to generate models that will be able to predict the price of European call options and predict whether the Black-Scholes formula will overstate or understate the option price with the greatest accuracies. A European call option gives the option holder the right, but not the obligation, to purchase the underlying asset at a prespecified strike price. Therefore, valuing the option is a very difficult problem because its value depends on the value of the underlying asset.

I am going to construct two models, one to predict the price of European call options which is a continuous variable and one to predict whether Black-Scholes will overstate or understate the option price which is a categorical variable. Therefore, I will be constructing a regression model and a classification model for the two problems respectively.

We were given two datasets, one training dataset with historic pricing on 2,800 options and one testing dataset with features for 200 options, but no pricing information was given. Therefore, the first action that I took on the training dataset was to split the training dataset into a training and validation dataset using a 70-30 split so that we can accurately test the models’ performances. In this report, when building the models, I only trained the models on the training dataset and only used the validation dataset to test the performance of the models on unseen data. Therefore, all of the performance measurements stated in this report are measures that are reported on the validation dataset because the model has not seen the data in the validation dataset.

# Preprocessing

Whenever I first start constructing a model, the first action that I take is to examine the dataset and perform any data cleaning and preprocessing, if necessary. After examining the dataset, there were no missing values that needed to be dealt with and there were no values that I could discern that were incorrectly inputted. I then wanted to see whether there were any outliers in the dataset. This is quite a complex problem in it of itself since outlier detection is a field of its own. However, since the goal of this project is to construct predictive models that have the highest accuracy, we wanted to create models that model the normal pricing scenarios and exclude the values of outliers since that will ultimately bring down the accuracy of the model.

Visually, I wanted to see whether there were any material or notable differences between the training and testing data regarding the values of the features that were given. This is illustrated in Figure 1 and Figure 2 where I constructed visualizations of the respective feature values in both of the datasets. To my understanding, I did not notice any obscure differences between the datasets other than the fact that the testing dataset generally has more concentrated (smaller range) variable distributions as compared with the training dataset. This suggests that when training the model, we should focus on making sure that the models should particularly perform well on those specific ranges.

## Outlier Detection

Another important facet of the preprocessing stage is to detect and treat outliers to make sure that the model’s inputs are representative of what we are trying to predict. To test for outliers, I z-scored the training dataset to check how many data points were at a greater distance than three standard deviations. When testing the individual features, I found that there were no such rows. However, for the Value variable, I found that there were 7 occurrences where the data point for the value variable was just outside three standard deviations from the mean. However, since there are 2,800 data points, we can reasonably expect that 0.3% of the data points will be greater than three standard deviations away from the mean. Therefore, I did not take any action on the data points.

I also used various boxplots to see whether there were any outliers in the datasets. As shown in the appendix from Figure 3 through Figure 11, it shows that there are no outliers in the majority of the variables except for value and r features in the training dataset. However, as I tested earlier from the z-score metric, there are only a small handful of outliers and that is due to the probabilistic nature of the dataset so I have elected to not take any action on these outliers.

## Feature Engineering and Feature Selection

Another important facet of the model building process is to make sure to incorporate as many important variables as possible to make sure that the inputs to the model are as sound as possible. After examining the variables, I noticed that it makes a lot more sense to include another variable called “Diff” which takes the difference between the current asset value, S, and the strike price of the option, K. I did this because what is really important for option pricing is to know whether the option is currently “in the money”, “out of the money”, or how far it is away from being “in the money”. Stated another way, I want to know whether it makes economical sense for the option holder to exercise the option and how far away the option holder is currently from exercising the option. If the current asset value is way below the strike price, then it is very improbable that the option holder will be able to exercise the option and therefore, the option will be priced very low. On the contrary, if the strike price of the option is below the current asset value of the stock, then the option holder will profit on the option and the option price will likely be very high. Although the complex models will learn this pattern indefinitely using only the S and K variables, I tested each of the models using both the S and K variables and the resulting Diff variable and found that the models are more robust and more accurate using only the Diff variable. Therefore, I have added the Diff variable to the dataset and have chosen to not use the S and K variables because those variables will be perfectly collinear with the Diff variable and will therefore cause problems in the model building process and introduce additional error. I also formally perform a feature selection procedure by utilizing stepwise regression later in the report.

# Value Prediction

The first model that I am going to create is a model that will be able to predict the current value of the option. Since the value of the option is a continuous variable, this is a regression type problem so I have chosen the different types of algorithms to suit this type of problem. Since this is a regression problem, I wanted to test how our models perform by measuring how close they are to the actual values. Specifically, I am going to use root-mean squared error (RMSE) and mean absolute percentage error (MAPE) to check the performance of the models. I chose these two performance measurements specifically because we are most interested in how close the model’s prediction is to the actual values.

## Multiple Linear Regression

The first model that I used was multiple linear regression as that is the simplest and most interpretable model for regression type problems. It also helps us determine the most important variables in the model building process. The first multiple linear regression model that I built was using all of the variables together without any interaction terms. This model is shown in Figure 12 which shows that all of the variables are significant in predicting the value of the option. I tested the performance of this model against the validation dataset and found that the RMSE of this model is 4.3 and the MAPE is 429.1 as shown in Figure 13.

After this, I constructed a model that includes all of the two-way interaction terms and used a stepwise regression using both forwards and backwards elimination to determine the most relevant variables to the model. The resulting model is shown in Figure 14 where it shows that all of the interactions were found to be significant. I tested the accuracy of the model on the validation dataset and found that the RMSE is 4.18 and the MAPE is 429.8 as shown in Figure 15.

## Support Vector Machines

The next model that I tested is using Support Vector Machines (SVM). SVM’s are a good model to use for regression problems because they can effectively map out more complex non-linear relationships. I constructed a SVM with the current variables as shown in Figure 16. I tested the performance of the SVM model on the validation dataset and achieved an RMSE of 0.76 and a MAPE of 46.53 which are drastic improvements from the multiple linear regression approach as shown by Figure 17.

## Random Forest

The next model that I tested is using Random Forests for this regression problem. I decided not to use decision trees because those perform better at classification problems and found that the performance of the decision trees were below the performances of our other models. However, I wanted to test random forests since they generally perform pretty well on both regression and classification type problems. I tested the number of trees to use for the model to prevent overfitting and found that 200 trees seemed to produce the best results in terms of reproducibility. I tested the performance on the validation dataset and found that the RMSE of this model is 1.53 while the MAPE is 88.4.

## XGBoost

The last model that I created for predicting the value of the option is the most complex model, using XGBoost. I wanted to use this advanced model so that I can see whether this model outperforms some of the other, more simpler models even though it is computationally expensive. To train the XGBoost algorithm, I used a 10-fold cross validation sample so that the model is able to get trained on all of the datapoints while also not overfitting the training data. I tested the performance of this model against the validation dataset and achieved an RMSE of 0.427 and a MAPE of 9.24. This means that on average, our predictions are off by 9.24% and that we are off by 0.427 units, on average.

## Conclusion

After testing all of the various models to predict the value of the option, there was a clear winner in terms of accuracy measures that performed the best and that was XGBoost. XGBoost had the highest accuracy measures followed by our SVM model which both outperformed the other models by quite a significant margin. So for the purposes of submitting the final predictions on the testing dataset, I have used XGBoost simply because of the minimal RMSE and MAPE performance measurements on the out-of-sample validation dataset. Furthermore, since XGBoost is cross-validated, I can infer that it has done a better job of seeing through all various combinations of the dataset and can generalize its performance quite well. If I had more time, I would have tested cross validation on all of the models to see how that impacts their performance and I would have tested more bagging and boosting advanced models to see how their performance compares.

# BS Prediction

The next problem that I need to tackle is to create a model that is able to determine whether the option price will be underestimated by the Black-Scholes formula or whether it will be overestimated by the formula. This is a binomial classification problem since we are creating a model that has two outcomes.

## Binomial Logistic Regression

The first model that I tried for this classification problem is the binomial logistic regression because it is one of the simplest and most interpretable models for a binary classification problem. I first tested a standard regression using all of the available variables excluding any interaction variables or transformations as shown in Figure 20. The model shows that all of the variables were found to be significant and that the model works reasonably well. When using the model for prediction on the validation dataset, the model has an accuracy of 90.8% on the validation dataset as shown in Figure 21. To calculate the prediction values, since a binomial logistic regression initially outputs probabilities, I used a cutoff value of 0.5 so that all values above 0.5 are classified as 1, or “overestimated”, and below 0.5 will be classified as 0, or “underestimated”.

Then, I included all of the different two-way interaction terms between all of the available variables and ran a stepwise binomial logistic regression using both forwards and backwards elimination. The resulting model is shown in Figure 22 which shows that the interaction variable diff:tau was added to the model. However, the accuracy of the model did not improve as it remained exactly the same when testing the performance of the model on the validation dataset as shown in Figure 23.

## Decision Trees

The next model that I tried for this classification problem is decision trees as decision trees are very good at classification problems in general. The improvement that decision trees have over a binomial logistic regression is that it can model the data in a different, nonlinear manner using simple decisions that make it very easy to interpret. A visual representation of the resulting decision tree can be seen in Figure 24 which shows that the decision tree is very shallow and easy to interpret as it solely bases the classification based on the diff variable. When using the decision tree for classification, I achieved an improved accuracy of 91.9% on the validation dataset. This was very amazing for us to find out as this decision tree was much simpler than the binomial logistic regression, but achieved greater accuracy using only a single decision rule.

## Support Vector Machines

The next model that I tried for this classification problem is support vector machines (SVM). SVM’s are a good model to use for this type of problem because they help minimize the effect of irrelevant data points and generate better separation lines for classification because of the use of support vectors. The SVM model that I constructed is depicted in Figure 26. For the SVM models, I tested both with the additional diff variable and with the S and K variables separately and found that for this model, using the S and K variables resulted in better out-of-sample accuracy as compared with the model using solely the diff variable. When testing the performance of this model on the validation dataset, the SVM model achieved an accuracy of 92.3% as shown in Figure 27. Including all of the interaction terms improved the model accuracy to 92.67%.

## Random Forest

The next model that I tested was random forests which is an extension of decision trees. This is also generally a good model that has good predictive power, however, one of the main apprehensions I have about using this model is that since there are limited data points, there is a tendency for this model to overfit the data and not generalize to the actual relationship of the data. I tested out various tree sizes ranging from 5 to 500 trees and found that the optimal number of trees to prevent overfitting on the training dataset was 8 trees. After constructing the random forest, I tested the performance on the validation dataset and achieved an accuracy of 93.92% as shown in Figure 28.

## Neural Networks

I also wanted to test artificial neural networks to see whether they have good performance on this classification problem. Just like random forests, neural networks also have a tendency to overfit the data for small datasets such as the one that I have currently. Therefore, I tested the performance, experimented with the different number of hidden nodes, and concluded that this would not be a good choice of model to use for this problem specifically since the accuracy measures were performing relatively poorly as compared with other models.

## XGBoost

Another model that I wanted to test is more boosting algorithms such as XGBoost. I had similar apprehensions that this model will overfit such a small dataset and result in poor predictive power. However, to combat this overfitting, I implemented cross validation when training the model on the training dataset using a 10-fold cross validation. This is helpful because this way, the model can get to see every single data point in the dataset, while not overfitting to the training dataset. I found that this model had an accuracy of 93.2% when tested on the validation dataset as shown in Figure 29.

## Conclusion

After testing all of these models, I noticed that their final performance measurements were not that different for most of the models. Most of the model’s performances changed slightly when retraining the models so the decision to pick the best model is quite difficult. If simplicity was concerned, I would have chosen decision trees as they were very simple, easy to interpret, and produced good generalizable results. However, since we are concerned with obtaining the highest accuracy, I opted to use random forest for the final predictions on the testing dataset. I was concerned about overfitting the data with some of the other models such as random forest so I opted to choose Support Vector Machines for our final submission. If I had more time, I would have tested cross validation on all of the various models to see whether that would improve their performance.

Appendix

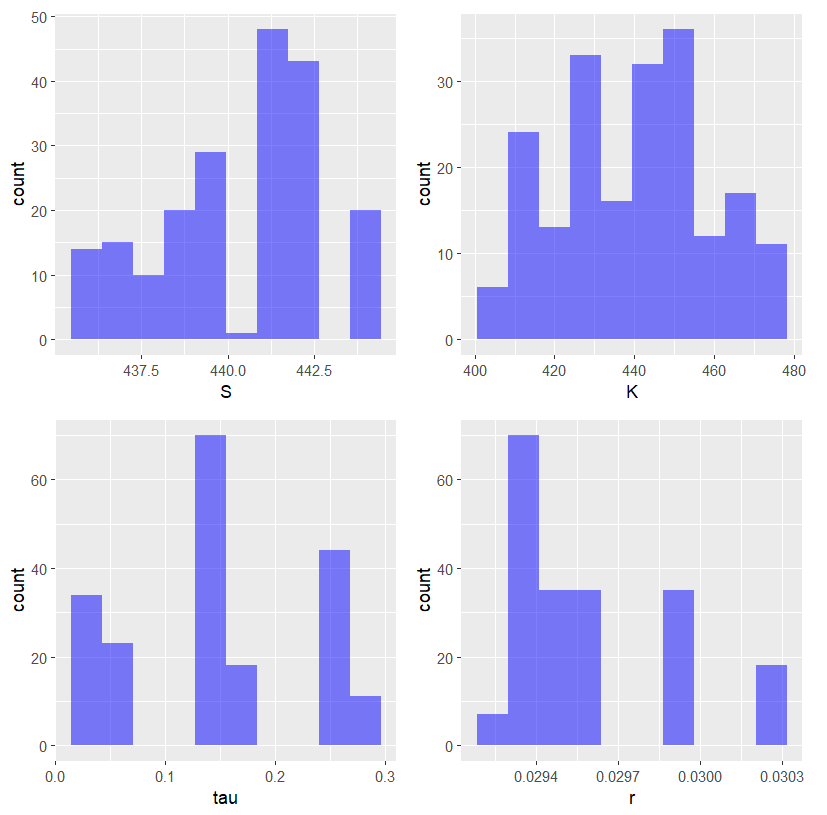


Figure 1. Testing Data Visualizations

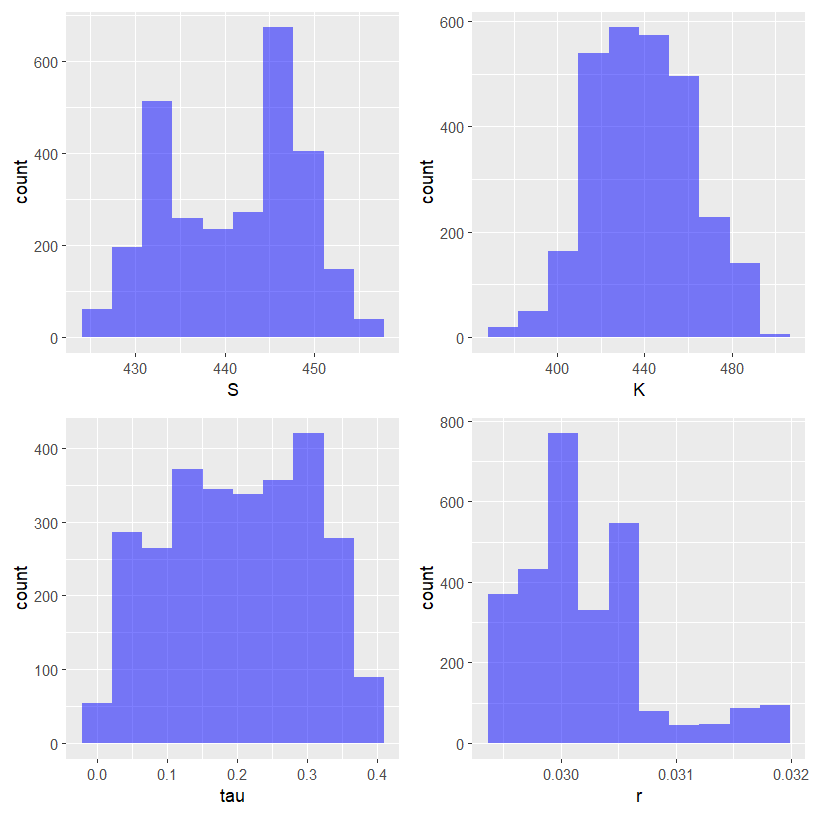


Figure 2. Training Data Visualizations

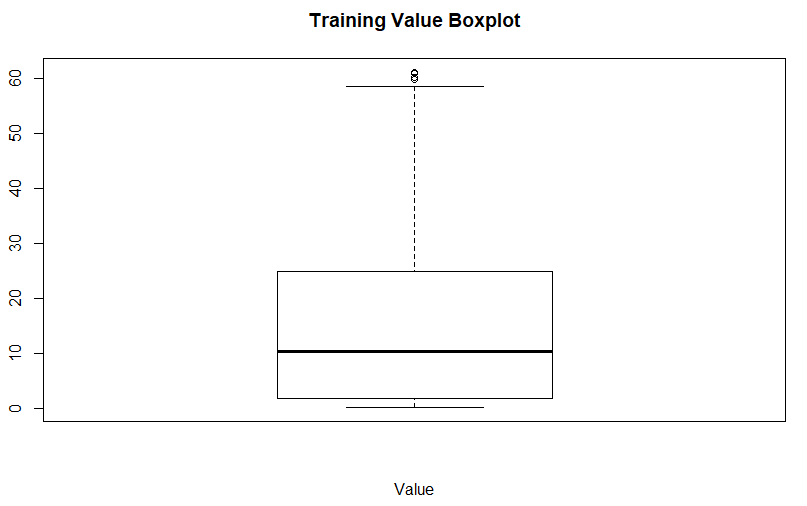


Figure 3. Boxplot of Value from the Training Dataset

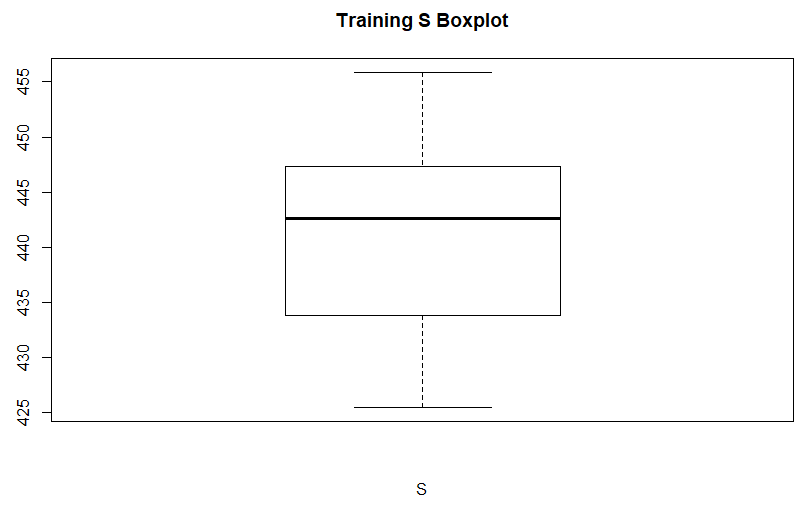


Figure 4. Boxplot of S from the Training Dataset

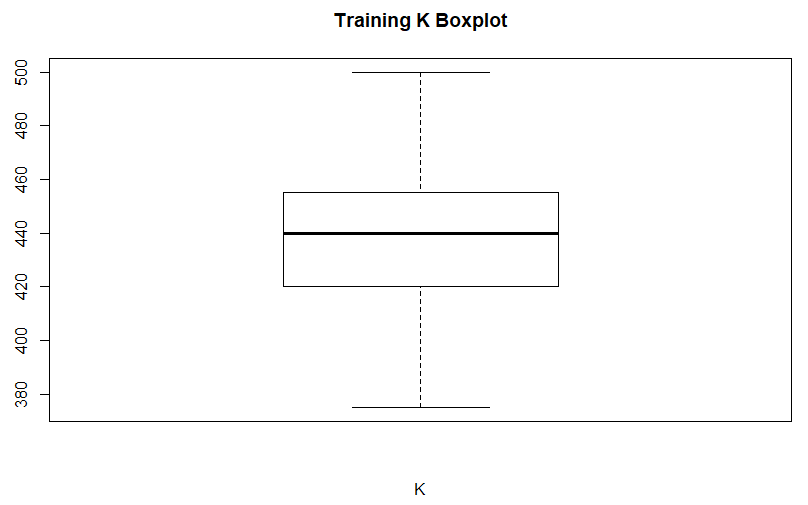


Figure 5. Boxplot of K from the Training Dataset

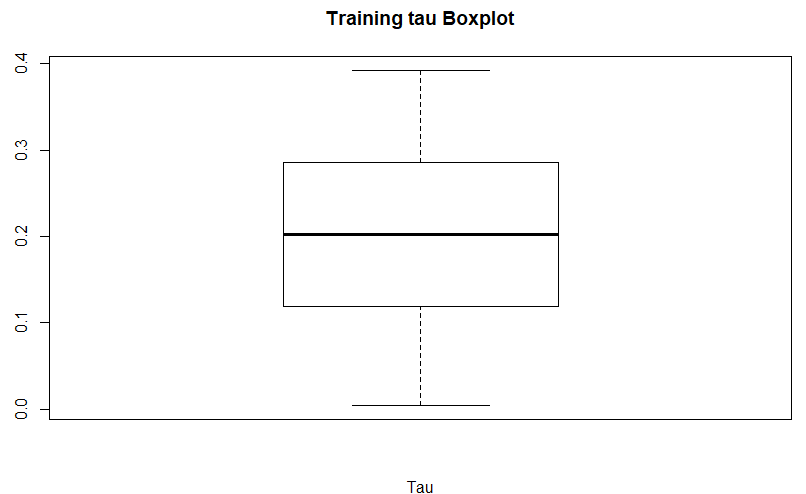


Figure 6. Boxplot of Tau from the Training Dataset

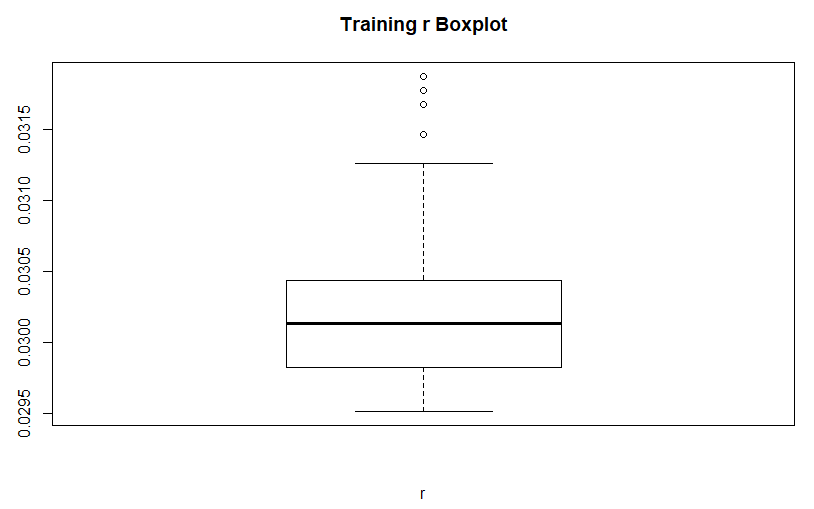


Figure 7. Boxplot of R from the Training Dataset

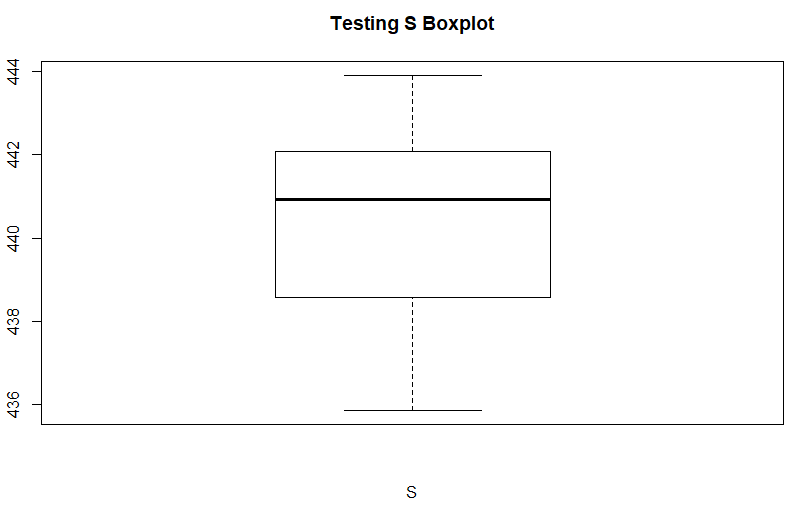


Figure 8. Boxplot of S from the Testing Dataset

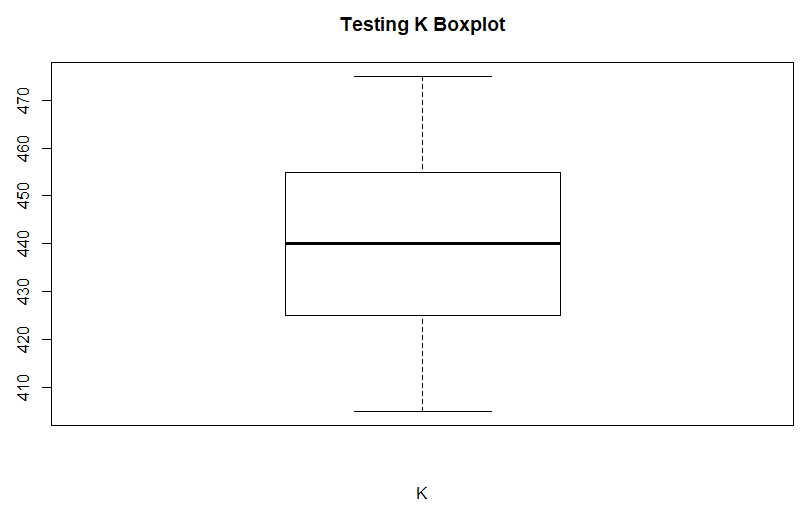


Figure 9. Boxplot of K from the Testing Dataset

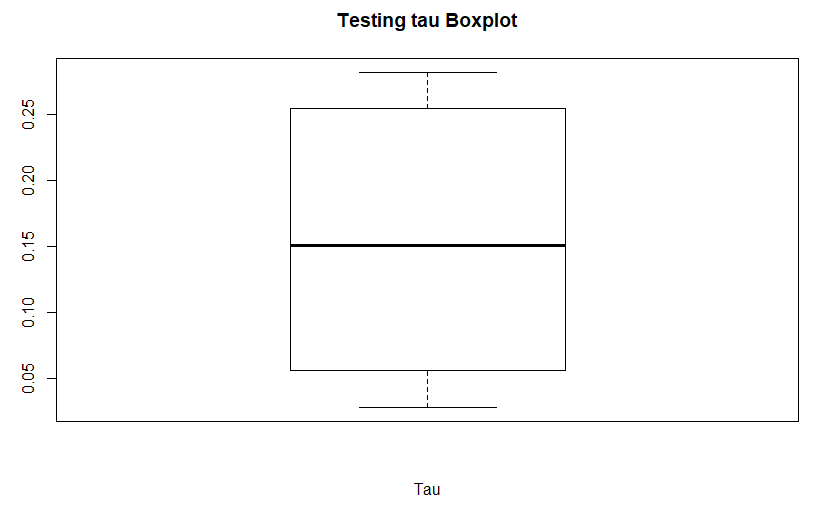


Figure 10. Boxplot of tau from the Testing Dataset

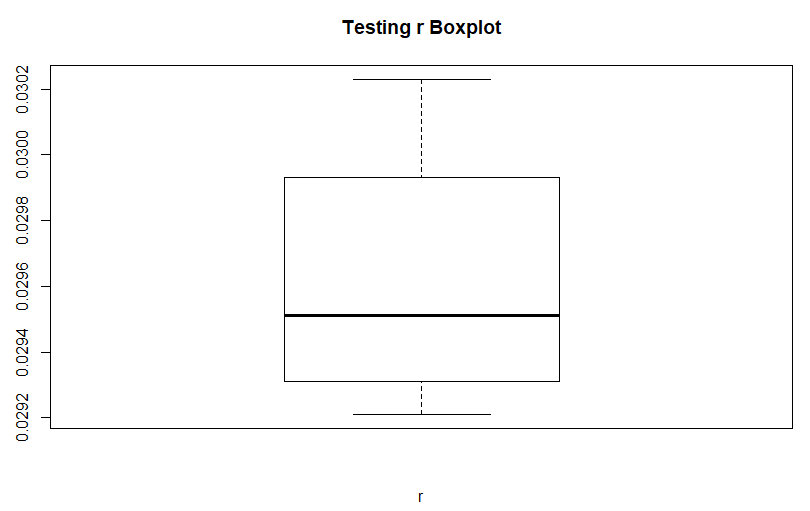


Figure 11. Boxplot of R from the Testing Dataset

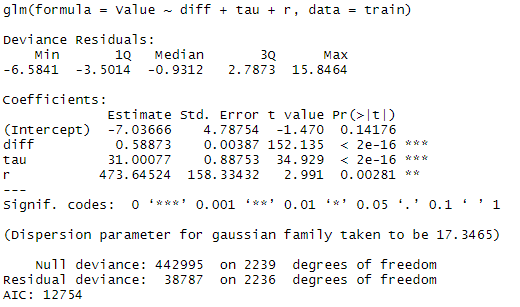


Figure 12. Multiple Linear Regression 1 for Value



Figure 13. Multiple Linear Regression 1 Accuracy

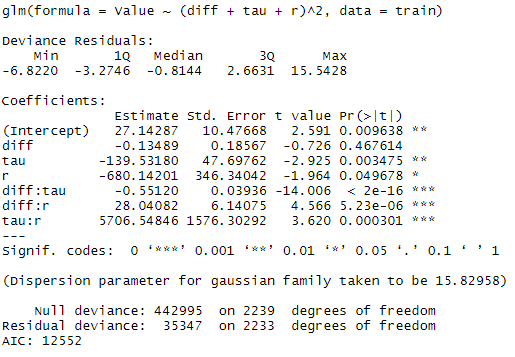


Figure 14. Multiple Linear Regression 2 for Value after Stepwise Regression



Figure 15. Multiple Linear Regression 2 Accuracy

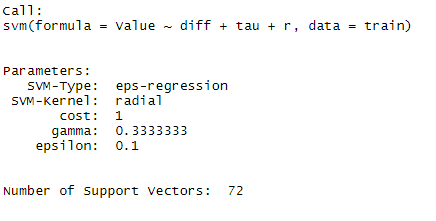


Figure 16. Support Vector Machine (SVM) for Value



Figure 17. SVM Accuracy



Figure 18. Random Forest Accuracy



Figure 19. XGBoost Accuracy

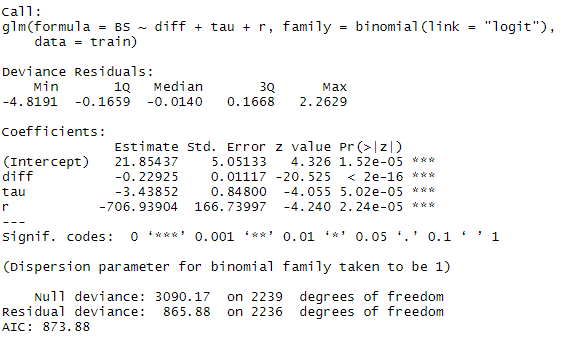


Figure 20. Binomial Logistic Regression 1 for BS



Figure 21. Accuracy of Binomial Logistic Regression 1 for BS

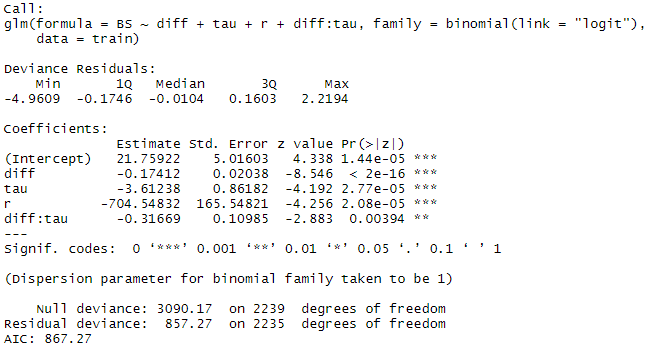


Figure 22. Binomial Logistic Regression 2 for BS after Stepwise Regression



Figure 23. Accuracy of Binomial Logistic Regression 2

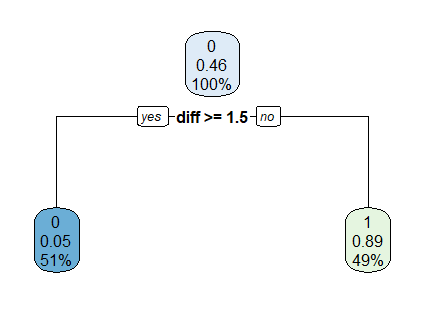


Figure 24. Decision Tree for BS



Figure 25. Accuracy for Decision Tree

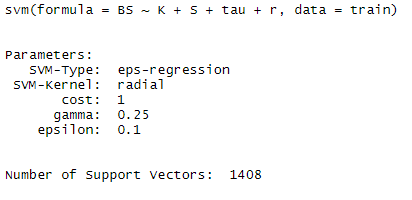


Figure 26. Support Vector Machine (SVM) model for BS



Figure 27. Accuracy for SVM



Figure 28. Random Forest for BS Accuracy



Figure 29. XGBoost for BS Accuracy