Data Mining and Predictive Analytics for Business (BUDT758T)

Spring 2023

Classification Model for Twitter Spam Detection based on User and Tweet Characteristics

Team Members: Aviva Mehta, Jacob Mackoff, Jarrar Haider, Shiv Jethi, Pengfeng (Jonathan) Yei

ORIGINAL WORK STATEMENT

We the undersigned certify that the actual composition of this proposal was done by us and is original work.

	Typed Name	**Signature**
	Aviva Mehta	Aviva Mehta
	Jarrar Haider	Jarrar Haider
	Shiv Jethi	Shiv Jethi
	Jonathan	Ye Pengfei
	Jacob Mackoff	Jacob Mackoff

II. Executive Summary

Our research aimed to develop a predictive model for identifying Twitter spammers by analyzing user account and tweet characteristics. This study is important due to the need for accurate and efficient methods to detect and prevent cyber attacks, online fraud, and abuse on social media platforms. By identifying patterns in the data that distinguish spammers from legitimate users, we can improve cybersecurity measures and protect users from harm.

We analyzed a dataset of 5998 tweets, consisting of both account-based features (e.g., account age, number of followers) and tweet-based features (e.g., number of retweets, number of hashtags). After data preprocessing, we used seven classification techniques to evaluate the performance of each model based on accuracy and AUC metrics.

Our key findings include the identification of important predictors for spam accounts, particularly the number of times the twitter user was listed, number of hashtags in the tweet, number of characters in the tweet, and the age of the Twitter account. We observed that account age below one year had the highest proportion of spammers, indicating its potential as a distinguishing factor. We identified Random Forests as the model with highest accuracy, followed by Bagging and Boosting models.

III. Data Description

The data used in this study was obtained from a publicly available dataset on Github, linked here. The dataset contains information on 5998 tweets with 13 variables. The variables in the dataset include both account-based features and tweet-based features. Table 1.1 shows which features are included in this analysis and categorizes them as either account-based features or tweet-based features.

Table 1.1: Independent variables with descriptions

Feature Category	**Feature Name**	**Feature Description**
Account-based features	account_age	Age of the account (days)
	no_follower	Number of followers
	no_following	Number of users that follow the account
	no_favorites	Number of times the account was favorited
	no_lists	Number of times the user has been listed
	no_tweets	Number of user-posted tweets
Tweet-based features	no_retweets	Number of times the tweet was retweeted
	no_hashtags	Number of times a hashtag appears in tweet
	no_mentions	Number of times this tweet is mentioned
	no_urls	Number of URLs contained in the tweet
	no_char	Number of characters in the tweet
	no_digits	Number of digits in the tweet

Our dependent variable is named ‘classification’ and each observation is classified as either ‘spammer’ or ‘non-spammer’. A snapshot of our data can be seen in Appendix 1.1.
The dataset is of particular interest in the cybersecurity field because by identifying the most predictive variables for spam accounts/tweets, we can develop a more efficient and accurate way of detecting and classifying Twitter spammers. This has practical applications for social media platforms and marketers who want to filter out spam accounts. Additionally, this project could have broader implications for detecting and preventing online fraud and abuse.

III. Research Questions

As social media platforms become increasingly vulnerable to cyber attacks and other forms of online fraud and abuse, it has become more important to develop accurate and efficient methods of detecting and preventing these types of activities. This data mining project aims to develop a predictive model for identifying Twitter spammers based on a set of user account and tweet characteristics. By identifying patterns in the data that are indicative of cyber attacks or other malicious activities, social media companies like Twitter can develop better cybersecurity measures tailored to the specific characteristics of malicious accounts. Ultimately, this will help to ensure the integrity and trustworthiness of social media platforms and protect users from harm. The questions this projects hopes to answer are:

1. Can patterns in the data be identified that are indicative of a Twitter account being used for cyber attacks or other malicious activities?
2. How does the behavior of Twitter spammers differ from that of legitimate users in terms of account and tweet characteristics, and how can this information be used to improve cybersecurity measures?
3. Can the likelihood of a Twitter account being a spam account be predicted based on its account and tweet characteristics, and what are the most important predictors of this outcome?

IV. Exploratory Data Analysis

In the dataset, 64.04% of the accounts are not spammer accounts, while the rest 35.96% are all spammer accounts, which can be seen in Figure 4.1. This was slightly surprising since if this is a true random sample of Twitter users, almost one-third of all Twitter users are spam accounts.

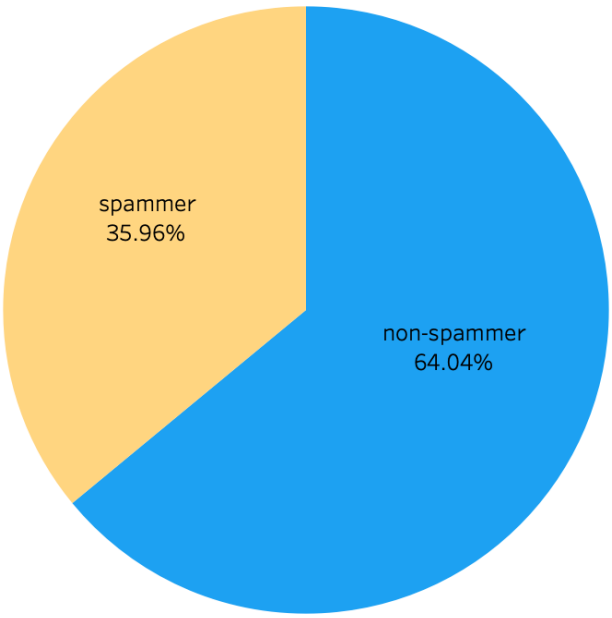

Figure 4.1: Proportion of all spam vs. non-spam accounts in the dataset

More than 80% of accounts with an account age of 1 or less are spammers (seen in Figure 4.2), which makes this an important variable for detecting spam accounts.

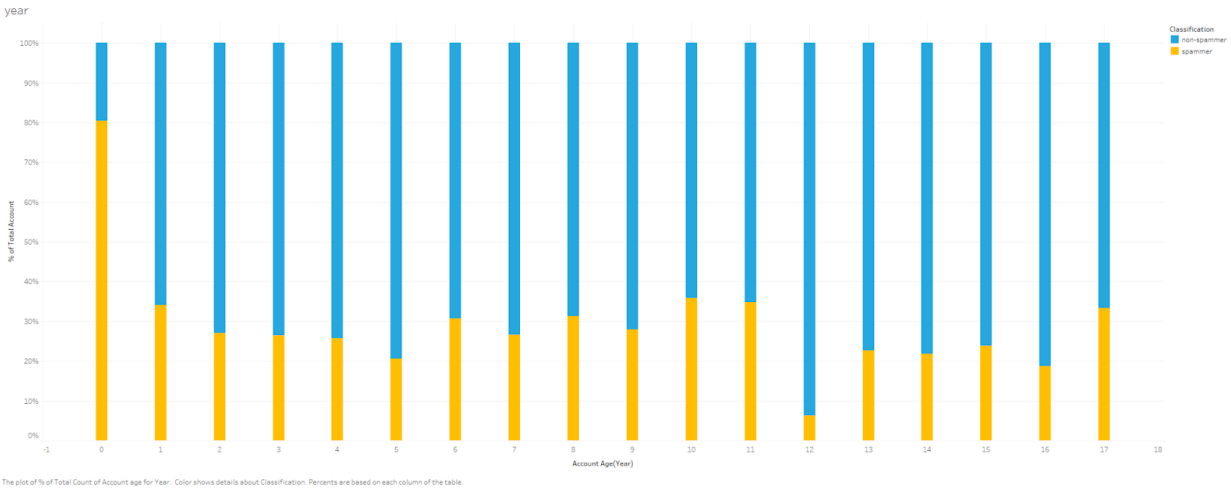

Figure 4.2: Proportion of spam vs. non-spam accounts by account age

Finally, the heatmap in Figure 4.3 displays correlation between our independent variables. Account age is strongly correlated with the number of followers the account has and number of favorites a tweet has, and the number of accounts the user is following is strongly correlated with number of favorites. This could cause complications in our analysis, and could affect our accuracy.

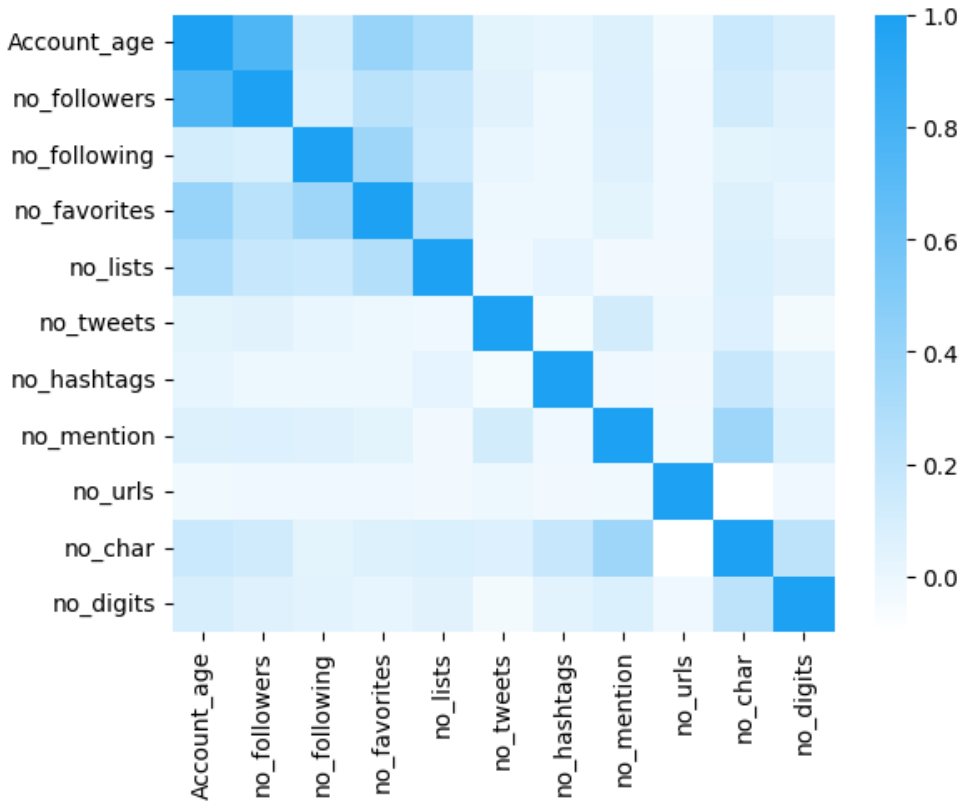

Figure 4.3: Heatmap showing correlation between independent variables

V. Methodology

a. Data Preprocessing:

The original dataset obtained from Kaggle contained three spreadsheets: a training set with 1998 observations and two testing sets, each containing 2000 unique observations. To initiate data preparation, the three spreadsheets were merged into a consolidated dataset consisting of 5998 rows. This consolidation was performed to allow a random split using R. Subsequently, exploratory analysis was conducted on the data through data visualizations to identify potential outliers that could adversely affect result accuracy.

During the analysis of the "Account_age" column, which represents the number of days since the creation of a Twitter account, notable outliers were detected with values in the hundreds of thousands, suggesting account ages in the range of hundreds of years. Considering Twitter's launch in 2006 and the dataset's timeframe of 2021, it was evident that these observations were implausible. Consequently, all rows with an Account_age exceeding 5110 days (14 years) were

removed from the dataset. A 14-year threshold was chosen to encompass all accounts created from 2006 to 2020, while excluding accounts created in 2021 due to the unlikelihood of the dataset containing accounts from that year. After filtering for Account_age values less than or equal to 5110, the dataset decreased from 5998 to 5658 observations. In addition, the column containing the predictor no_retweets was removed as it possessed solely 0 values. Following the elimination of illogical outliers and the removal of n, the data was divided into training (60%), validation (20%), and testing (20%) sets.

b. Model Development:

To select the model that best classifies spammers in the dataset, all techniques learned throughout the semester that have classification capabilities were utilized, including: Logistic Regression, K-Nearest Neighbors (KNN), Naive Bayes, Classification Trees, Random Forests, Bagging, and Boosting. All models were built exclusively on the training set, and their performance metrics (Accuracy and AUC) were evaluated on the test set. This approach ensured a fair comparison of performance across different models. Model tuning techniques, such as classification tree pruning and optimal K selection for KNN, were applied on the validation set. Upon constructing all models and calculating their respective performance measures, the model with the lowest error rate and highest AUC on the test set was selected. To conclude the analysis, the data was resplit into two sets: a 60% training set and a 40% validation set. Performance measures were recalculated on this new split to obtain more accurate results regarding the optimal model's ability to accurately classify spam accounts on Twitter.

VI. Results and Finding

Table 6.1: Performance of Machine Learning Models

Model	**Accuracy**	**AUC**
Logistic Regression	70.95%	0.755
Naive Bayes	47.97%	0.734

KNN	79.95%	0.850
Classification Tree	81.71%	0.817
Bagging	87.72%	0.921
Random Forest	88.25%	0.932
Boosting	85.77%	0.842

Table 6.1 presents the accuracy and AUC measures for each classification model, reflecting their performance on the test dataset in classifying spam tweets. The results highlight a significant variation in the levels of performance among the models ranging from poor to moderate to good performers

Among the models, Naive Bayes displayed the weakest performance, with an accuracy of 47.97% and an AUC of .734. While Logistic Regression achieved a significant increase in both accuracy and AUC, being 70.95% and .755 respectively, it still can be considered a poor performer. KNN, Classification Trees, and Bagging all fell into the moderate performer category, achieving accuracies around 80% and AUC values of over 0.8. Finally, the ensemble methods Bagging and Random Forests exhibited good performance. Both models had accuracies of above 87% with AU values over 0.9. This can be attributed to the additional predictive power that comes from the use of a majority rule from a large number of trees. Overall, the Random Forest model yielded the best performance measures for classifying spam tweets.

Table 6.2: Relative Importance of Top Four Important Variables Using Random Forest

Predictors	**Mean Decrease Accuracy**	**Mean Decrease Gini Index**
Account_age	97.23	391.43
no_lists	101.45	306.86
no_hashtags	122.93	179.06

no_chars	109.24	203.71

Table 6.2 presents the most important variables in classifying tweets as spammers. This list was aggregated by comparing the Mean Decrease Accuracy and Mean Decrease Gini Index of all variables using the importance function of random forest, located in Appendix 3.1, and choosing the variables that had the highest. These were then cross checked with the logistic regression summary to ensure that all were significant predictors in terms of their p-values. This ultimately shows that when trying to detect if a tweet is spam, one should most focus on the above variables

Table 6.3: Updated Metrics for Random Forest Using 60-40 Data Split

Model	**Accuracy**	**AUC**
Random Forest	88.02%	0.9331

After recomputing the accuracy and AUC of the Random Forest model on a 60-40 split to increase the number of observations in the test set and thus increase the accuracy of our results, the above metrics were calculated. While the accuracy of the model slightly suffered, going from 88.25% to 88.02%, the AUC increased from 0.932 to 0.9331.

VII. Conclusion

Over the past decade, Twitter's popularity has attracted an increasing number of malicious actors to the platform. To classify users as spammers, our research paper employs a model that utilizes a variety of user-based and tweet-based features. We utilized seven different data mining techniques to determine the most effective approach for identifying spam users and the most important variables. Our results indicate that the Random Forests model was the most successful, achieving an accuracy of 88.25%. Although this is not an exhaustive list of methods for classifying spam tweets, our findings provide Twitter with valuable information for identifying spam accounts and have broader implications for cybersecurity. Our study highlights the

importance of data mining techniques in combating malicious activity and encourages social media and other website platforms to consider their use.

VIII. Appendix (Any additional information to be submitted):

Appendix 1.1: Raw data used for classification (dependent variable is column 13)

Account_age	no_followers	no_following	no_favorites	no_lists	no_tweets	no_retweets	no_hashtags	no_mention	no_urls	no_char	no_digits	classification
0	0	0	0	3	0	0	0	1	1	17	0	spammer
0	0	0	0	3	0	0	0	1	1	14	0	spammer
0	0	0	0	5	0	0	0	1	1	12	2	spammer
0	0	0	0	1	0	0	0	0	1	21	0	spammer
0	0	0	0	1	0	0	0	1	1	13	2	spammer
0	0	0	0	4	0	0	0	1	1	16	0	spammer
0	0	0	0	4	0	0	0	1	1	15	0	spammer
0	0	0	0	2	0	0	0	1	1	11	0	spammer
0	0	0	0	3	0	0	0	1	1	13	1	spammer
0	0	0	0	3	0	0	0	1	1	12	3	spammer
0	0	0	0	3	0	0	0	1	1	13	0	spammer
1	17	0	0	31	287	0	3	1	1	102	0	spammer
0	0	0	0	5	0	0	0	1	1	15	2	spammer
1	15	0	0	29	0	0	2	0	1	64	0	spammer

Appendix 1.2: R Code for logistic regression

````
**LOGISTIC REGRESSION**
```{r}
library(readxl)
log_df <- read_excel("BUDT758T Project Data.xlsx")
library(dplyr)
log_data = select(log_df, -1)
log_data$classification <- ifelse(log_data$classification == "spammer", 1, 0)
set.seed(123)
log_data$classification <- as.factor(log_data$classification)
head(log_data)
library(caret)
in_train <- createDataPartition(log_data$classification, p = 0.6, list = FALSE)
df_train <- log_data[in_train, ]
df_temp <- log_data[-in_train, ]
head(df_temp)
inVal <- createDataPartition(df_temp$classification, p = 0.5, list = FALSE)
df_validation <- df_temp[inVal, ]
df_test <- df_temp[-inVal, ]
head(df_test)
model <- glm(classification~., data = df_train, family = binomial())
predicted_values <- predict(model, df_test, type = "response")
binary_predictions <- ifelse(predicted_values > 0.5, 1, 0)
accuracy <- mean(binary_predictions == df_test$classification)
accuracy
summary(model)

#calculate AUC
library(pROC)
cat("AUC of Logistic Regression model = ", auc(df_test$classification, predicted_values), "\n")
```
````

Appendix 2.2: R Code for Naive Bayes Model

NAIVE BAYES

```{r}
set.seed(123)
#install.packages("naivebayes")
library(naivebayes)
naive_model <- naive_bayes(classification~., data = df_train)
summary(naive_model)
predicted_naive <- predict(naive_model, newdata = df_test)
accuracy_naive <- mean(predicted_naive == df_test$classification)
accuracy_naive

#calculate AUC
library(pROC)
cat("AUC of NAIVE BAYES model = ", auc(df_test$classification, predicted_naive$posterior[,"1"]), "\n")
```

Appendix 2.3: R Code for KNN Model

KNN Model

```{r}
library(readxl)
library(class)
dat<-read_excel("BUDT758T Project Data.xlsx")
dat$classification<-as.factor(dat$classification)
dat$classification <- relevel(dat$classification, ref = "spammer")
str(dat)
set.seed(123)

dat <- subset(dat, select = -Obs)
dat <- subset(dat, select = -no_retweets)
dat <- dat[dat$Account_age < 5110,]

model = glm(classification ~., data = dat, family = 'binomial')
summary(model)

# Normalize each variable
fun <- function(x){
  a <- mean(x)
  b <- sd(x)
  (x - a)/(b)
}

dat[,1:11] <- apply(dat[,1:11], 2, fun)
```

```
#Splitting Dataset
index <- sample(nrow(dat), 0.6*nrow(dat))
train <- data.frame(dat[index,])
dataremain<- data.frame(dat[-index,])
index1 <- sample(nrow(dataremain), 0.5*nrow(dataremain))
validation <- data.frame(dataremain[index1, ])
test <- data.frame(dataremain[-index1, ])
rm(dataremain)

train_input <- as.matrix(train[,-12])
train_output <- as.vector(train[,12])
validate_input <- as.matrix(validation[,-12])
test_input <- as.matrix(test[,-12])

#Max k is 20
kmax <- 20
ER1 <- rep(0,kmax)
ER2 <- rep(0,kmax)
ER3 <- rep(0,kmax)
```

```r
#Predictions, confusion matrices, error rates
set.seed(123)
for (i in 1:kmax){
  prediction_train <- knn(train_input, train_input, train_output, k=i)
  prediction_val <- knn(train_input, validate_input, train_output, k=i)
  prediction_test <- knn(train_input, test_input, train_output, k=i)

  CM1 <- table(prediction_train, train$classification)
  ER1[i] <- (CM1[2,2] + CM1[1,1])/sum(CM1)
  CM2 <- table(prediction_val, validation$classification)
  ER2[i] <- (CM2[2,2] + CM2[1,1])/sum(CM2)
  CM3 <- table(prediction_test, test$classification)
  ER3[i] <- (CM3[2,2] + CM3[1,1])/sum(CM3)
}

#Plot training and validation error rates
plot(c(1,kmax),c(0,1), type= "n", xlab="k",ylab="Error Rate")
lines(ER1,col="red")
lines(ER2,col="blue")
legend(15, 0.5, c("Training","Validation"),lty=c(1,1), col=c("red","blue"))

#Find Minimum Validation Error
z <- which.min(ER2)
cat("Minimum Validation Error k:", z)
points(z,ER2[z],col="red",cex=2,pch=20)

#Create predictions for train, test, validation
prediction <- knn(train_input, train_input,train_output, k=z)
prediction2 <- knn(train_input, validate_input,train_output, k=z)
prediction3 <- knn(train_input, test_input,train_output, k=z)
```

Appendix 2.4: Random Forests and Bagging

BAGGING AND RANDOM FOREST

```{r}
#Import Dataset and Prep
library(readxl)
library(randomForest)
library(ROCR)
library(caret)
dat<-read_excel("BUDT758T Project Data.xlsx")
dat$classification<-as.factor(dat$classification)
dat$classification <- relevel(dat$classification, ref = "spammer")
str(dat)
set.seed(123)

#Removing Outliers (Unreasonable Observations)
dat$no_retweets<-NULL
dat <- dat[dat$Account_age < 5110,]
#Splitting Dataset
index <- sample(nrow(dat), 0.6*nrow(dat))
train <- data.frame(dat[index,])
dataremain<- data.frame(dat[-index,])
index1 <- sample(nrow(dataremain), 0.5*nrow(dataremain))
validation <- data.frame(dataremain[index1, ])
test <- data.frame(dataremain[-index1, ])


###########Bagging (Random forest with mtry=p)#######################
set.seed(123)
bag1<- randomForest(classification~., mtry=11, importance=TRUE, n.tree=1000, data=train)
bag1
importance(bag1)
```

```
#Confusion Matrix for Train Data
predtrainbag1<- predict(bag1, train)
CM_train_bag1<- confusionMatrix(predtrainbag1, train$classification, positive = "spammer")
CM_train_bag1
#Accuracy = 1
#Sensitivity = 1
#Specificity = 1

#Confusion Matrix for Test Data
predtestbag1<- predict(bag1, test)
CM_test_bag1<- confusionMatrix(predtestbag1, test$classification, positive = "spammer")
CM_test_bag1
#accuracy = .8754
#Sensitivity = .7537
#Specificity = .9425

#Error Rates Plotted
plot(bag1)

#ROC Curve of Train and Test for Bag 1

#Bagging with Different Parameters

bag2<- randomForest(classification~., mtry=11, importance=TRUE, n.tree=5000, data=train)
bag2
importance(bag2)
```

```r
#Confusion Matrix for Train Data
predtrainbag2<- predict(bag2, train)
CM_train_bag2<- confusionMatrix(predtrainbag2, train$classification, positive = "spammer")
CM_train_bag2
#Accuracy = 1
#Sensitivity = 1
#Specificity = 1

#Confusion Matrix for Test Data
predtestbag2<- predict(bag2, test)
CM_test_bag2<- confusionMatrix(predtestbag2, test$classification, positive = "spammer")
CM_test_bag2
#accuracy = .8781
#Sensitivity = .7587
#Specificity = .9438

#Error Rates Plotted
plot(bag2)

############################Random Forests##############################################
#Random Forest With 1000 trees and mtry = 4
rf1<- randomForest(classification~., mtry=4, importance=TRUE, n.tree=1000, data=train)
rf1
importance(rf1)

#Confusion Matrix for Train Data
predtrainrf1<- predict(rf1, train)
CM_train_rf1<- confusionMatrix(predtrainrf1, train$classification, positive = "spammer")
CM_train_rf1
#Accuracy = .9985
#Sensitivity = 1
```


```r
#Confusion Matrix for Train Data
predtrainbag2<- predict(bag2, train)
CM_train_bag2<- confusionMatrix(predtrainbag2, train$classification, positive = "spammer")
CM_train_bag2
#Accuracy = 1
#Sensitivity = 1
#Specificity = 1

#Confusion Matrix for Test Data
predtestbag2<- predict(bag2, test)
CM_test_bag2<- confusionMatrix(predtestbag2, test$classification, positive = "spammer")
CM_test_bag2
#accuracy = .8781
#Sensitivity = .7587
#Specificity = .9438

#Error Rates Plotted
plot(bag2)
```

```
############################Random Forests#################################################
#Random Forest With 1000 trees and mtry = 4
rf1<- randomForest(classification~., mtry=4, importance=TRUE, n.tree=1000, data=train)
rf1
importance(rf1)

#Confusion Matrix for Train Data
predtrainrf1<- predict(rf1, train)
CM_train_rf1<- confusionMatrix(predtrainrf1, train$classification, positive = "spammer")
CM_train_rf1
#Accuracy = .9985
#Sensitivity = 1
#Specificity = .9977

#Confusion Matrix for Test Data
predtestrf1<- predict(rf1, test)
CM_test_rf1<- confusionMatrix(predtestrf1, test$classification, positive = "spammer")
CM_test_rf1
#accuracy = .8799
#Sensitivity = .7587
#Specificity = .9466

#Error Rates Plotted
plot(rf1)

#ROC Curve of Train and Test for Bag 1

#RF with Different Parameters

rf2<- randomForest(classification~., mtry=3, importance=TRUE, n.tree=1000, data=train)
```

```
rf2<- randomForest(classification~., mtry=3, importance=TRUE, n.tree=1000, data=train)
rf2
importance(rf2)

#Confusion Matrix for Train Data
predtrainrf2<- predict(rf2, train)
CM_train_rf2<- confusionMatrix(predtrainrf2, train$classification, positive = "spammer")
CM_train_rf2
#Accuracy = .9956
#Sensitivity = .9984
#Specificity = .9940

#Confusion Matrix for Test Data
predtestrf2<- predict(rf2, test)
CM_test_rf2<- confusionMatrix(predtestrf2, test$classification, positive = "spammer")
CM_test_rf2
#accuracy = .8763
#Sensitivity = .7537
#Specificity = .9438

#Error Rates Plotted
plot(rf2)
```

CLASSIFICATION TREE AND BOOSTING

```{r}
tweets<-read.csv("BUDT758T Project Data.csv")
df <- data.frame(tweets)
head(df)

df[,15:26]<-NULL
df$Obs<-NULL
df$classification<- as.factor(df$classification)

#Splitting the dataset into 60-20-20

library(caret)
set.seed(123)
inTrain <- createDataPartition(df$classification, p=0.6, list=FALSE)
dftrain <- df[inTrain,] # with 60% of the data
dftemp <- df[-inTrain,] # with 40% of the data

##Now split the dftemp data set in two equal parts

inVal <- createDataPartition(dftemp$classification, p=0.5, list=FALSE)
dfvalidation <- dftemp[inVal,]
dftest <- dftemp[-inVal,]

library(tree)
tree.tweets=tree(classification~. ,dftrain)
summary(tree.tweets)
```

```
#Computing the training error
Prediction=predict(tree.tweets, dftrain,type="class")
Actual = dftrain$classification
# The confusion matrix and error rate
CM = table(Actual, Prediction)
(Train_Acc = (CM[1,1]+CM[2,2])/sum(CM))
Train_Error = 1 - Train_Acc
(Train_Error)

#Computing the test error (where dftest is 20% split)
Prediction.test=predict(tree.tweets, dftest,type="class")
Actual.test = dftest$classification
# The confusion matrix and error rate
CM2 = table(Actual.test,Prediction.test)
(Test_Acc = (CM2[1,1]+CM2[2,2])/sum(CM2))
Test_Error = 1 - Test_Acc
(Test_Error)

#Pruning the Tree
library(tree)
set.seed(123)
cv.tweets=cv.tree(tree.tweets, FUN=prune.misclass)
names(cv.tweets)

#plot(cv.tweets$size,cv.tweets$dev,type="b")
(i = which.min(cv.tweets$dev))
(z = cv.tweets$size[i])

#which.min(cv.tweets$size)```
#Now prune the tree
```

```r
#ENSEMBLE METHOD: BOOSTING
library(gbm)
set.seed(123)
#converting the classification variable to 0 and 1 where 1=spammer, 0=non-spammer
dftrain$class<- ifelse(dftrain$classification =="spammer",1,0)
#dftrain$class <- as.factor(dftrain$class)
dftest$class<- ifelse(dftest$classification=="spammer",1,0)
#dftest$class <- as.factor(dftest$class)

boost.tweets = gbm(class~. -classification ,
                   data=dftrain,
                   distribution="bernoulli",
                   n.trees=5000,
                   interaction.depth=4
                   )

#summary(boost.tweets)
#Computing the MSE using test data
yhat.tweets <- predict(boost.tweets, newdata=dftest, n.trees=5000, distribution="bernoulli", type="response")
mean((yhat.tweets - dftest$class)^2)


Actual.test2 <- dftest$class
predicted <- ifelse(yhat.tweets>=0.5,1,0)
CM4 = table( predicted, Actual.test2)
(Acc = (CM4[1,1]+CM4[2,2])/sum(CM4))
```

Appendix 3.1: Importance Measures

```
> importance(rf1)
               spammer non-spammer MeanDecreaseAccuracy MeanDecreaseGini
Account_age   51.56070    36.81608             68.00002        286.79735
no_followers  22.21622    40.83355             49.28473        154.43961
no_following  41.60341    32.95636             54.00411        124.31502
no_favorites  19.91712    24.39179             29.99978         68.14295
no_lists     112.37833    33.55785            101.44661        306.85802
no_tweets     37.93775    14.82828             39.18497         57.09263
no_hashtags   80.18338    51.82885             88.41721        171.76910
no_mention    31.37243    12.79633             34.62211         42.84449
no_urls       27.11242    24.84357             34.27106         21.17076
no_char       93.14383    14.23103             88.44491        216.38725
no_digits     62.55150    24.52343             62.90533        105.61942
```

```
> importance(bag1)
               spammer non-spammer MeanDecreaseAccuracy MeanDecreaseGini
Account_age   55.13706    63.82613             97.22821        391.43237
no_followers  20.44232    51.79572             58.53300        141.12952
no_following  40.23498    44.04665             61.90258        105.46279
no_favorites  29.16220    28.07516             39.76422         59.25111
no_lists     151.54454    42.41814            132.73324        271.77966
no_tweets     43.17718    14.14620             45.35248         56.37342
no_hashtags  114.59239    58.34831            122.92669        179.05853
no_mention    32.92486    20.42326             38.37003         39.24167
no_urls       30.21141    24.92921             35.68923         20.96679
no_char      109.36071    20.01780            109.23766        203.70618
no_digits     76.36877    23.95394             70.68344         96.24158

> summary(logistic_model)

Call:
glm(formula = classification ~ ., family = "binomial", data = train)

Deviance Residuals:
    Min       1Q   Median       3Q      Max
-3.2847  -1.0499   0.5283   0.9258   2.6139

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept)  -1.823e+00  2.675e-01  -6.817 9.31e-12 ***
Account_age  -4.590e-04  9.762e-05  -4.702 2.57e-06 ***
no_followers  3.830e-04  9.471e-05   4.043 5.27e-05 ***
no_following  3.468e-05  2.633e-05   1.317    0.188
no_favorites  3.309e-02  6.839e-03   4.839 1.30e-06 ***
no_lists      3.336e-05  3.545e-06   9.409  < 2e-16 ***
no_tweets    -1.546e-05  1.197e-05  -1.291    0.197
no_hashtags  -2.981e-01  3.714e-02  -8.027 9.98e-16 ***
no_mention    6.948e-02  5.910e-02   1.176    0.240
no_urls       1.220e+00  2.488e-01   4.904 9.38e-07 ***
no_char       1.843e-02  1.440e-03  12.795  < 2e-16 ***
no_digits    -1.739e-01  1.752e-02  -9.926  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 4439.0  on 3393  degrees of freedom
Residual deviance: 3785.3  on 3382  degrees of freedom
AIC: 3809.3

Number of Fisher Scoring iterations: 6
```