**Assignment 1: Advanced Classification (100 points)**

**Student Name: Alekhya Akkinepally**

**Purpose:** To build and test advanced classifiers and prescribe strategies

**Description:** Using data from 2010 Congressional elections, we intend to build a classifier that would predict the election’s outcome. The data set includes information about the campaign funds, social media (Twitter, Facebook, and YouTube) campaigns, and demographics (age, gender) of 941 candidates who were in race in the general elections for The 112th House of Representatives seats in The U.S. Congress.[[1]](#footnote-1)

**Instructions:** You need to follow these steps:

1. In Canvas, navigate to Assignments and then Assignment1
2. Download and save the data set election\_campaign\_data.csv
3. Read the file: data <- read.csv("election\_campaign\_data.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))

**Ans) mydata <- read.csv("election\_campaign\_data.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))**

1. Drop the following variables from the data: "cand\_id", "last\_name", "first\_name", "twitterbirth", "facebookdate", "facebookjan", "youtubebirth".

**Ans) mydata$cand\_id <- NULL**

**mydata$first\_name <- NULL**

**mydata$last\_name <- NULL**

**mydata$twitterbirth <- NULL**

**mydata$facebookdate <- NULL**

**mydata$facebookjan <- NULL**

**mydata$youtubebirth <- NULL**

1. Convert the following variables into factor variables using function as.factor(): “twitter”, “facebook”, “youtube”, “cand\_ici”, and “gen\_election”.

**Ans) mydata$twitter <- as.factor(mydata$twitter)**

**mydata$facebook<- as.factor(mydata$facebook)**

**mydata$youtube <- as.factor(mydata$youtube)**

**mydata$cand\_ici <- as.factor(mydata$cand\_ici)**

**mydata$gen\_election <- as.factor(mydata$gen\_election)**

1. Bear in mind that “twitter” equals 1 if the candidate had a Twitter campaign during the election and zero otherwise. The same would apply for “facebook” and “youtube”. “opp\_fund” is the total campaign fund of the opposing candidate. “gen\_election” is our target variable which takes value of “L” when the candidate lost the election and “W” when the candidate won the election. For descriptions of other variables in the data, please refer to: <http://www.fec.gov/finance/disclosure/metadata/DataDictionaryWEBALL.shtml#search=%22trans_from_auth%22>
2. Remove all of the observations with any missing values using function complete.cases()

**Ans) mydata <- mydata[complete.cases(mydata),]**

1. Randomly assign 70% of the observations to train\_data and the remaining observations to test\_data (Refer to Module 6 for the code).

**Ans) trainIndex = sample(1:n, size = round(0.7\*n), replace=FALSE)**

**train\_data = mydata[trainIndex,]**

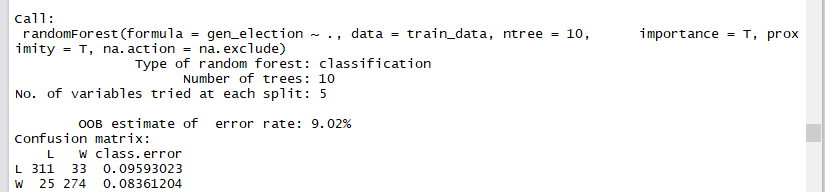
**test\_data = mydata[-trainIndex ,]**

1. Use train\_data to build a random forest classifier with 10 trees. Use library(randomForest).

**Ans) rf <- randomForest(gen\_election ~., data = train\_data, ntree= 10, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**



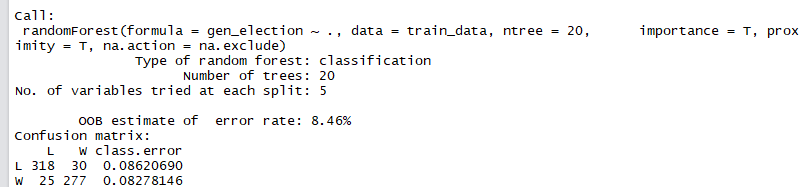
* 1. **(2 points)** What is the OOB estimate of error rate?

**Ans) 9.02**

* 1. **(2 points)** How many variables R tried at each split?

**Ans) 5**

* 1. **(4 points)** Now use 20 trees.



* + 1. What is OOB estimate of error rate?

**Ans) 8.46**

* + 1. How many variables R tried at each split?

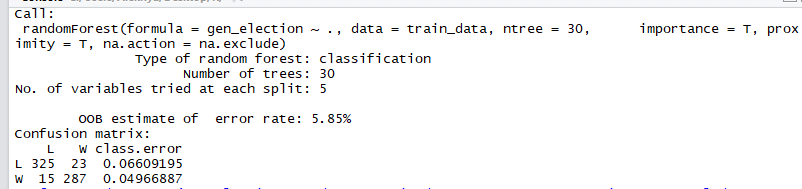
**Ans) 5**

* 1. **(4 points)** Now use 30 trees.

**Ans) rf <- randomForest(gen\_election ~., data = train\_data, ntree= 30, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**



* + 1. What is OOB estimate of error rate?

**Ans) 5.85**

* + 1. How many variables R tried at each split?

**Ans) 5**

* 1. **(2 points)** Increase the number of trees in 10 increments (e.g. 40, 50, …). Using OOB error rate to evaluate your random forest classifier, how many trees would you recommend?

**Ans) rf <- randomForest(gen\_election ~., data = train\_data, ntree= 40, na.action = na.exclude,**

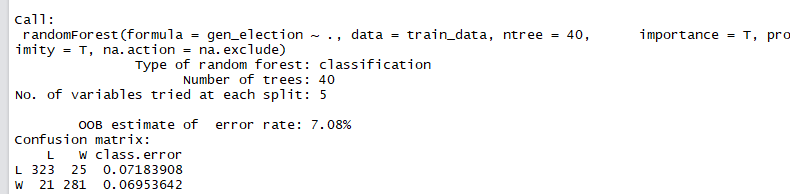
**importance = T, proximity = T)**

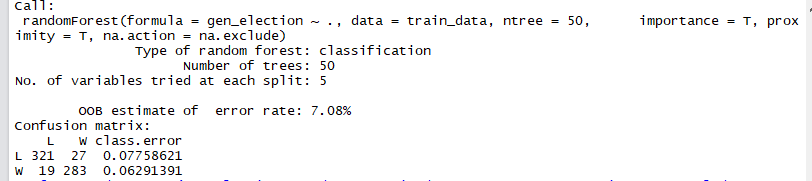
**print(rf)**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 50, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**





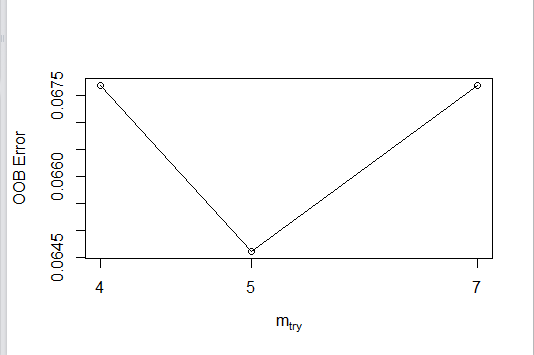
**I would choose number of trees as 3 where I have got a minimum OOB error as 5.85**

* 1. **(2 points)** Use tuneRF() function to find the best value for mtry. Here is the code:

*mtry <- tuneRF(train\_data[-26], train\_data$gen\_election, ntreeTry=n, stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE, , na.action=na.exclude).* Replace n with the number of trees you recommended in 9.5. What is the recommended value for mtry?

**Ans) mtry <- tuneRF(train\_data[-26], train\_data$gen\_election, ntreeTry=30,**

**stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE, na.action = na.exclude)**



* 1. **(2 points)** Use your recommended number of trees and mtry value to build a new random forest classifier using train\_data. What is OOB estimate of error rate?

**Ans) 6.46%**

**newrf <-randomForest(gen\_election~., data=train\_data, mtry=5, importance=TRUE, ntree=30)**

**print(rf)**

* 1. **(8 points)** Use library(caret)[[2]](#footnote-2) and the code in Module 6 to create the confusion matrix for test\_data. Fill out the confusion matrix in below. Use “W” as the value of option positive in confusionMatrix() function.

**Ans) library(caret)**

**predicted\_values <- predict(newrf, test\_data,type= "prob")**

**head(predicted\_values)**

**threshold <- 0.5**

**pred <- factor( ifelse(predicted\_values[,2] > threshold, "W", "L") )**

**levels(test\_data$gen\_election)[2]**

**confusionMatrix(pred, test\_data$gen\_election,**

**positive = levels(test\_data$gen\_election)[2])**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual | | |
| Predicted |  | W | L |
| W | **126** | **10** |
| L | **7** | **136** |

* + 1. What is the value of accuracy?

**Ans) 0.9391**

* + 1. What is the value of TPR?

**Ans) 0.9474**

* + 1. What is the value of FPR?

**Ans) 0.048**

* 1. **(4 points)** Use the code in Module 6 to calculate AUC and create the ROC curve.

**Ans) library(ROCR)**

**library(ggplot2)**

**predicted\_values <- predict(newrf, test\_data,type= "prob")[,2]**

**pred <- prediction(predicted\_values, test\_data$gen\_election)**

**perf <- performance(pred, measure = "tpr", x.measure = "fpr")**

**auc <- performance(pred, measure = "auc")**

**auc <- auc@y.values[[1]]**

**roc.data <- data.frame(fpr=unlist(perf@x.values),**

**tpr=unlist(perf@y.values),**

**model="RF")**

**ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +**

**geom\_ribbon(alpha=0.2) +**

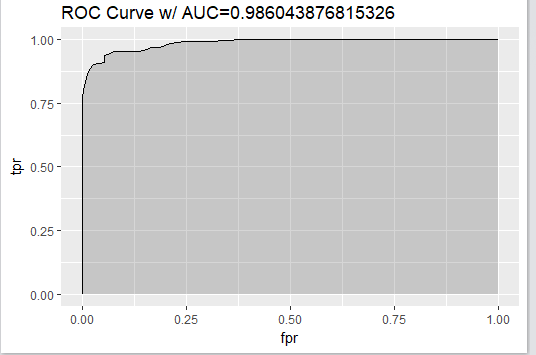
**geom\_line(aes(y=tpr)) +**

**ggtitle(paste0("ROC Curve w/ AUC=", auc))**

* + 1. What is the value of AUC?

**Ans) 0.986**

* + 1. Paste the ROC curve in the space below:



* 1. **(4 points)** Use varImpPlot() to create the plot for variable importance. What are the type five important variables when we use MeanDecreaseAccuracy?

**Ans ) varImpPlot(newrf)**

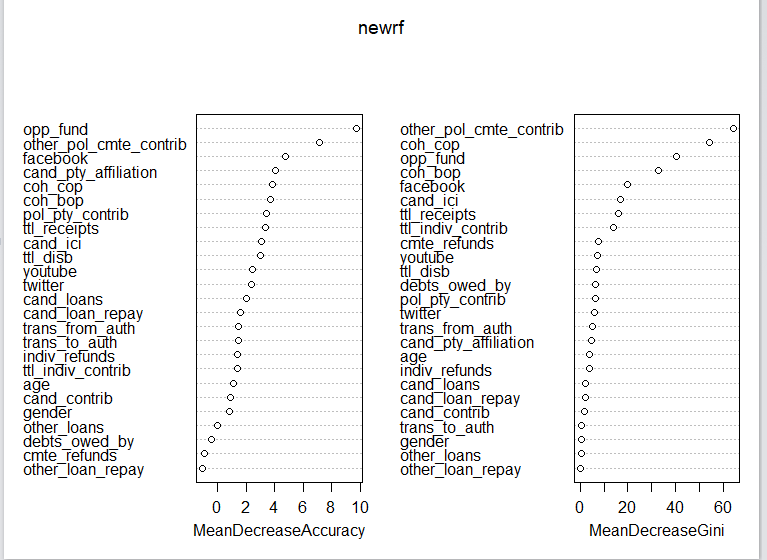
**opp\_fund**

**other\_pol\_cmte\_contrib**

**facebook**

**cand\_pty\_affiliation**

**coh\_cop**



1. Use library(nnet) and the code in Module 6 to build a neural network classifier.
   1. **(20 points)** Use 5 hidden nodes in your ANN.

**library(nnet)**

**ann <- nnet(gen\_election ~ ., data=train\_data, size=5, maxit=1000)**

**summary(ann)**

* + 1. How many input nodes are in the ANN?

**Ans) 39**

* + 1. How many weights are in the ANN?

**Ans) 206**

* + 1. Use library(caret) and the code in Module 6 to create the confusion matrix for test\_data. Fill out the confusion matrix in below. Use “W” as the value of option positive in confusionMatrix() function.

**Ans) predicted\_values1 <- predict(ann, test\_data,type= "raw")**

**head(predicted\_values1)**

**threshold <- 0.5**

**pred1 <- factor( ifelse(predicted\_values1[,1] > threshold, "W", "L") )**

**head(pred1)**

**levels(test\_data$gen\_election)[2]**

**confusionMatrix(pred1, test\_data$gen\_election,**

**positive = levels(test\_data$gen\_election)[2])**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Actual | | |
| Predicted |  | W | L |
| W | **120** | **26** |
| L | **8** | **120** |

* + 1. What is the value of sensitivity?

**Ans) 0.9398**

* + 1. What is the value of specificity?

**Ans) 0.8219**

* + 1. Use the code in Module 6 to calculate AUC and create the ROC curve.

**Ans) library(ggplot2)**

**predicted\_values1 <- predict(ann, test\_data,type= "raw")**

**pred1 <- prediction(predicted\_values1, test\_data$gen\_election)**

**perf1 <- performance(pred1, measure = "tpr", x.measure = "fpr")**

**auc1 <- performance(pred1, measure = "auc")**

**auc1 <- auc1@y.values[[1]]**

**roc.data1 <- data.frame(fpr=unlist(perf1@x.values),**

**tpr=unlist(perf1@y.values),**

**model="ANN")**

**ggplot(roc.data1, aes(x=fpr, ymin=0, ymax=tpr)) +**

**geom\_ribbon(alpha=0.2) +**

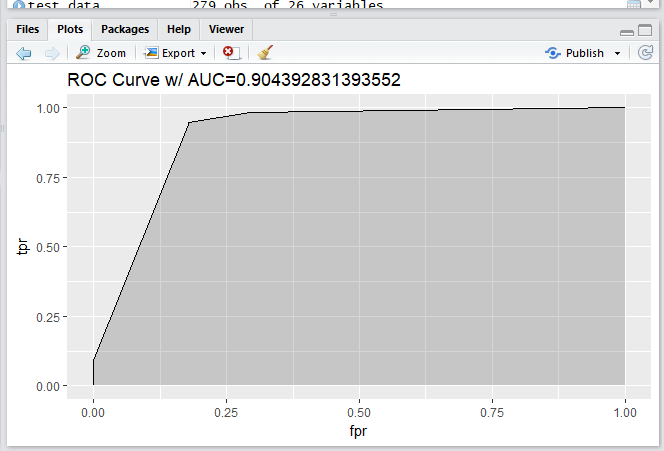
**geom\_line(aes(y=tpr)) +**

**ggtitle(paste0("ROC Curve w/ AUC=", auc1))**

* + - 1. What is the value of AUC?

**Ans) 0.9043**

* + - 1. Paste the ROC curve in the space below:



* 1. **(6 points)** Increase the number of hidden nodes until you get the following error: “*Error in nnet.default(x, y, w, entropy = TRUE, ...): too many (1026) weights.”* Use the maximum number of hidden nodes that you can use to build your ANN classifier.

**Ans) new\_ann <- nnet(gen\_election ~ ., data=train\_data, size=24, maxit=1000)**

* + 1. What is the maximum number of hidden nodes that we could use?

**Ans) 24**

* + 1. Use the code in Module 6 to calculate AUC and create the ROC curve.

**Ans) predicted\_values2 <- predict(new\_ann, test\_data,type= "raw")**

**pred2 <- prediction(predicted\_values2, test\_data$gen\_election)**

**perf2 <- performance(pred2, measure = "tpr", x.measure = "fpr")**

**auc2 <- performance(pred2, measure = "auc")**

**auc2 <- auc2@y.values[[1]]**

**roc.data2 <- data.frame(fpr=unlist(perf2@x.values),**

**tpr=unlist(perf2@y.values),**

**model="ANN")**

**ggplot(roc.data2, aes(x=fpr, ymin=0, ymax=tpr)) +**

**geom\_ribbon(alpha=0.2) +**

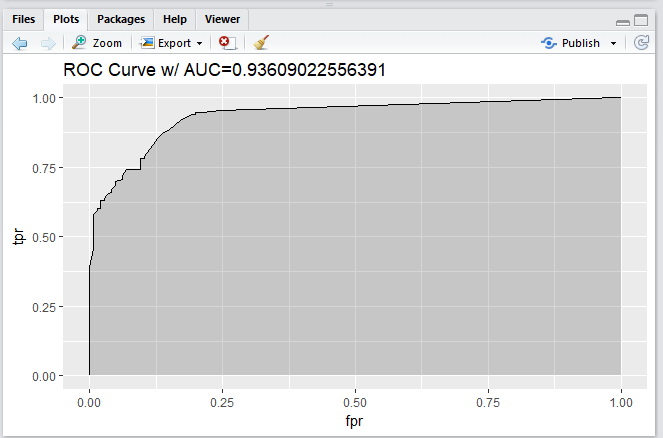
**geom\_line(aes(y=tpr)) +**

**ggtitle(paste0("ROC Curve w/ AUC=", auc2))**

* + - 1. What is the value of AUC?

**Ans) 0.936**

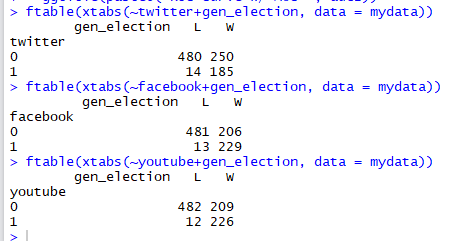
* + - 1. Paste the ROC curve in the space below:



1. **(5 points)** Among the three classifiers that you built, which classifier would you finally use for predicting the election’s outcome? Please explain.

**Ans) Among the three classifiers I would prefer using the Random Forest classifier because its accuracy and the AUC values are good when compared to the both the Neural Networks classifiers whose AUC values are less. It has a better choice of data.**

1. **(10 points)** The buzz from the 2008 election motivated the candidates for political offices to employ social media campaigns to get their message across. Imagine that you are an advisor to a candidate who is running for a Congressional seat. Based on your analysis, would you recommend sparing money and resources to create social media campaigns? If so, among the three social media platforms (Facebook, Twitter, and YouTube), which platform would you recommend to invest in? Please explain. You can use function ftable() in Module 3 to support your recommendation.

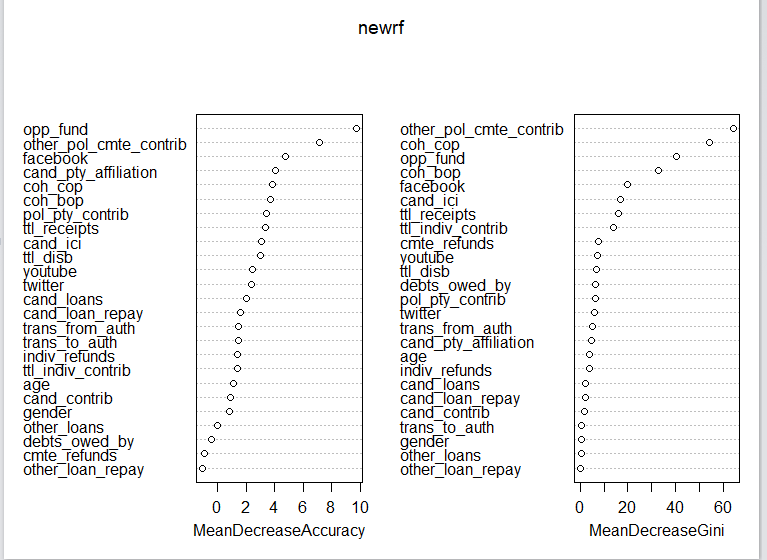


**Ans) From all the analysis we could conclude how important social media is for the campaigns. As per the above analysis winning candidates have used social media a lot than the candidates who have lost. And if given an opportunity of choosing which media platform to choose the numbers show it very clearly that Facebook is a very good platform for a candidate to invest in to increase their chance of winning.**

1. **(10 points)** Given your analysis, would you agree with this statement: “Money Buys Political Power”? Please explain.

**Ans) yes, I agree with the statement “Money Buys Political power” because campaigning and advertising in media require a lot of investment. Candidates who spend more money has a huge chance of winning.**

1. **(10 points)** Imagine that you are an advisor to a candidate who is running for a Congressional seat. Based on your analysis, what are your prescriptions for success for your candidate? Please explain.



**Ans) As per the varlmpPlot, there were five variables which would influence the output.**

**The variables opposition fund, Facebook and Contributions from other political committees, ending cash, these are all related to money. If the parties have more money they could do spend that in campaigning and advertising especially Facebook which is the one of the top factors which was a reason for a candidate to win.**

1. **(5 points)** Please paste your R code in the space below:

**Ans) setwd("/Users/Alekhya/Desktop/R")**

**# Read file**

**mydata <- read.csv("election\_campaign\_data.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))**

**summary(mydata)**

**#dropping the following variables**

**mydata$cand\_id <- NULL**

**mydata$first\_name <- NULL**

**mydata$last\_name <- NULL**

**mydata$twitterbirth <- NULL**

**mydata$facebookdate <- NULL**

**mydata$facebookjan <- NULL**

**mydata$youtubebirth <- NULL**

**#Converting variables as factors**

**mydata$twitter <- as.factor(mydata$twitter)**

**mydata$facebook<- as.factor(mydata$facebook)**

**mydata$youtube <- as.factor(mydata$youtube)**

**mydata$cand\_ici <- as.factor(mydata$cand\_ici)**

**mydata$gen\_election <- as.factor(mydata$gen\_election)**

**# Remove all the observations with any missing values**

**mydata <- mydata[complete.cases(mydata),]**

**# Randomly assign 70% train data**

**set.seed(32)**

**n=nrow(mydata)**

**trainIndex = sample(1:n, size = round(0.7\*n), replace=FALSE)**

**train\_data = mydata[trainIndex,]**

**test\_data = mydata[-trainIndex ,]**

**#Installing and loading randomForest**

**install.packages("randomForest")**

**library(randomForest)**

**# Building a randomForest classifier**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 10, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**# Building a randomforest classifier with n= 20**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 20, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**# Building a randomforest classifier with n= 30**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 30, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**# Building a randomforest classifier with n= 40**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 40, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**# Building a randomforest classifier with n= 50**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 50, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**# Building a randomforest classifier with n= 60**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 60, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**# Building a randomforest classifier with n= 70**

**rf <- randomForest(gen\_election ~., data = train\_data, ntree= 70, na.action = na.exclude,**

**importance = T, proximity = T)**

**print(rf)**

**#Tune mtry**

**mtry <- tuneRF(train\_data[-26], train\_data$gen\_election, ntreeTry=30,**

**stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE, na.action = na.exclude)**

**# New RandomForest Classifier**

**newrf <-randomForest(gen\_election~., data=train\_data, mtry=5, importance=TRUE, ntree=30)**

**print(rf)**

**#Confusion Matrix**

**install.packages("caret")**

**library(caret)**

**predicted\_values <- predict(newrf, test\_data,type= "prob")**

**head(predicted\_values)**

**threshold <- 0.5**

**pred <- factor( ifelse(predicted\_values[,2] > threshold, "W", "L") )**

**levels(test\_data$gen\_election)[2]**

**confusionMatrix(pred, test\_data$gen\_election,**

**positive = levels(test\_data$gen\_election)[2])**

**# AUC & ROC**

**install.packages("ROCR")**

**library(ROCR)**

**library(ggplot2)**

**predicted\_values <- predict(newrf, test\_data,type= "prob")[,2]**

**pred <- prediction(predicted\_values, test\_data$gen\_election)**

**perf <- performance(pred, measure = "tpr", x.measure = "fpr")**

**auc <- performance(pred, measure = "auc")**

**auc <- auc@y.values[[1]]**

**roc.data <- data.frame(fpr=unlist(perf@x.values),**

**tpr=unlist(perf@y.values),**

**model="RF")**

**ggplot(roc.data, aes(x=fpr, ymin=0, ymax=tpr)) +**

**geom\_ribbon(alpha=0.2) +**

**geom\_line(aes(y=tpr)) +**

**ggtitle(paste0("ROC Curve w/ AUC=", auc))**

**# Varmplot**

**varImpPlot(newrf)**

**#ANN**

**install.packages("nnet")**

**library(nnet)**

**ann <- nnet(gen\_election ~ ., data=train\_data, size=5, maxit=1000)**

**summary(ann)**

**#Confusion Matrix for ANNvarImpPlot(newrf)**

**predicted\_values1 <- predict(ann, test\_data,type= "raw")**

**head(predicted\_values1)**

**threshold <- 0.5**

**pred1 <- factor( ifelse(predicted\_values1[,1] > threshold, "W", "L") )**

**head(pred1)**

**levels(test\_data$gen\_election)[2]**

**confusionMatrix(pred1, test\_data$gen\_election,**

**positive = levels(test\_data$gen\_election)[2])**

**#ROC &AUC**

**library(ggplot2)**

**predicted\_values1 <- predict(ann, test\_data,type= "raw")**

**pred1 <- prediction(predicted\_values1, test\_data$gen\_election)**

**perf1 <- performance(pred1, measure = "tpr", x.measure = "fpr")**

**auc1 <- performance(pred1, measure = "auc")**

**auc1 <- auc1@y.values[[1]]**

**roc.data1 <- data.frame(fpr=unlist(perf1@x.values),**

**tpr=unlist(perf1@y.values),**

**model="ANN")**

**ggplot(roc.data1, aes(x=fpr, ymin=0, ymax=tpr)) +**

**geom\_ribbon(alpha=0.2) +**

**geom\_line(aes(y=tpr)) +**

**ggtitle(paste0("ROC Curve w/ AUC=", auc1))**

**new\_ann <- nnet(gen\_election ~ ., data=train\_data, size=24, maxit=1000)**

**predicted\_values2 <- predict(new\_ann, test\_data,type= "raw")**

**pred2 <- prediction(predicted\_values2, test\_data$gen\_election)**

**perf2 <- performance(pred2, measure = "tpr", x.measure = "fpr")**

**auc2 <- performance(pred2, measure = "auc")**

**auc2 <- auc2@y.values[[1]]**

**roc.data2 <- data.frame(fpr=unlist(perf2@x.values),**

**tpr=unlist(perf2@y.values),**

**model="ANN")**

**ggplot(roc.data2, aes(x=fpr, ymin=0, ymax=tpr)) +**

**geom\_ribbon(alpha=0.2) +**

**geom\_line(aes(y=tpr)) +**

**ggtitle(paste0("ROC Curve w/ AUC=", auc2))**

**# Confusion matrix**

**ftable(xtabs(~twitter+gen\_election, data = mydata))**

**ftable(xtabs(~facebook+gen\_election, data = mydata))**

**ftable(xtabs(~youtube+gen\_election, data = mydata))**

1. To read more about the general election, please refer to: <http://www.wwnorton.com/college/polisci/campaignsandelections/ch/09/outline.aspx>, and for information about the U.S. Congress, please refer to: <https://en.wikipedia.org/wiki/United_States_Congress> [↑](#footnote-ref-1)
2. You may get an error that will require you to use library(e1071). In this case, install this library using install.packages(“e1071”). [↑](#footnote-ref-2)