Assignment 1: Advanced Classification (100 points)
Student Name: Abdul Rahman Gulam.
Fall 2019

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]
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","","?"))

```
getwd()
setwd("C:/Users/Abdul Rahman/Documents/Train")
election_campaign_data <- read_csv("election_campaign_data.csv")
mydata <- read.csv("election_campaign_data.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))
nrow(mydata)
summary(mydata)
```

```
> mydata <- read.csv("election_campaign_data.csv", sep=",", header=T, strip.white = T, na.strings = c("NA","NaN","","?"))
> nrow(mydata)
[1] 941
> summary(mydata)
```

4. Drop the following variables from the data: "cand_id", "last_name", "first_name", "twitterbirth", "facebookdate", "facebookjan", "youtubebirth".

```
> mydata$cand_id <- NULL
> mydata$last_name <- NULL
> mydata$first_name <- NULL
> mydata$twitterbirth <-NULL
> mydata$facebookdate <-NULL
> mydata$facebookjan <-NULL
> mydata$youtubebirth <-NULL
>
```

5. Convert the following variables into factor variables using function as.factor(): "twitter", "facebook", "youtube", "cand_ici", and "gen_election".

[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

```
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)
```

6. 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

Untitled1* × mydata ×

Filter

	twitter	facebook	youtube	cand_ici	cand_pty_affiliation	ttl_receipts
1	0	0	0	C	DEM	48278.52
2	0	0	0	C	GRE	445.00
3	0	0	0	C	REP	446468.16
4	0	0	0	C	REP	87768.00

7. Remove all of the observations with any missing values.

```
> mydata <- mydata[complete.cases(mydata),]
> mydata <- mydata[complete.cases(mydata),]
> nrow(mydata)
[1] 929
```

8. Randomly assign 70% of the observations to train_data and the remaining observations to test_data.

```
> set.seed(32)
> n = nrow(mydata)
> trainIndex = sample(1:n,
                      size = round(0.7*n),
                      replace=FALSE)
> train_mydata = mydata[trainIndex,]
> test_mydata = mydata[-trainIndex,]
> summary(mydata)
```

9. Use train_data to build a random forest classifier with 10 trees.
   9.1. **(2 points)** What is the OOB estimate of error rate?
   OOB estimate of error rate = 8.61%
   9.2. **(2 points)** How many variables R tried at each split?
   R tried to split 5 variables.

```
Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      ntree = 10, importance = T, proximity = T, na.action = n
a.exclude)
               Type of random forest: classification
                     Number of trees: 10
No. of variables tried at each split: 5

        OOB estimate of  error rate: 8.61%
Confusion matrix:
    L   W class.error
L 309  28  0.08308605
W  27 275  0.08940397
```

9.3. **(4 points)** Now use 20 trees.

9.3.1. What is OOB estimate of error rate?

OOB estimate of error rate = 6.15%

9.3.2. How many variables R tried at each split?

R tried to split 5 variables.

```
Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      ntree = 20, importance = T, proximity = T, na.action = n
a.exclude)
               Type of random forest: classification
                     Number of trees: 20
No. of variables tried at each split: 5

        OOB estimate of  error rate: 6.15%
Confusion matrix:
    L   W class.error
L 321  22  0.06413994
W  18 289  0.05863192
```

9.4. **(4 points)** Now use 30 trees.

9.4.1. What is OOB estimate of error rate?

OOB estimate of error rate = 6.46%

9.4.2. How many variables R tried at each split?

R tried to split 5 variables.

```
Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      ntree = 30, importance = T, proximity = T, na.action = n
a.exclude)
               Type of random forest: classification
                     Number of trees: 30
No. of variables tried at each split: 5

        OOB estimate of  error rate: 6.46%
Confusion matrix:
    L   W class.error
L 318  25  0.07288630
W  17 290  0.05537459
```

9.5. **(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?

For n=40,50, 60…OOB error rate is 6.46%, 7.08%, 7.38%...I would recommend 30 or 40 trees as the OOB error rate is constant on both the scenario.

```
Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      ntree = 40, importance = T, proximity = T, na.action = n
a.exclude)
               Type of random forest: classification
                     Number of trees: 40
No. of variables tried at each split: 5

        OOB estimate of  error rate: 6.46%
Confusion matrix:
    L   W class.error
L 313  30  0.08746356
W  12 295  0.03908795
```

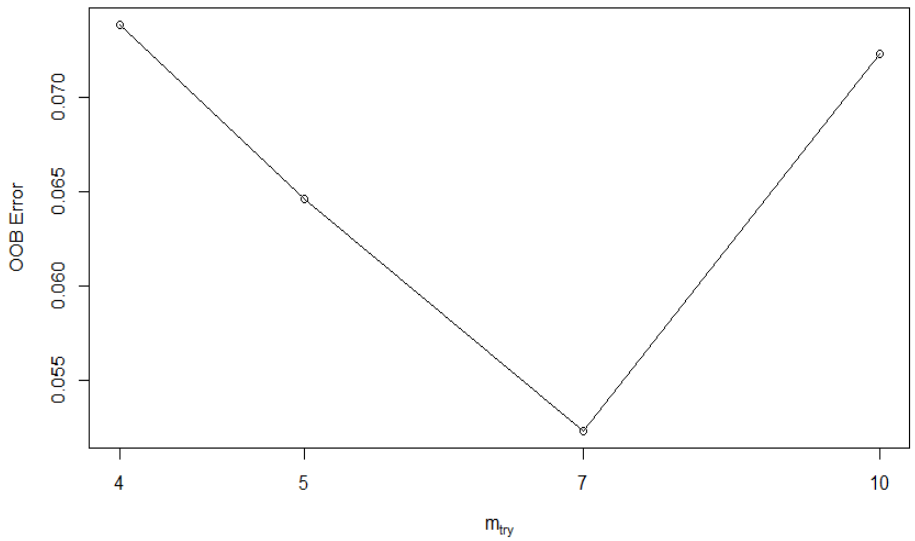
```
Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      ntree = 50, importance = T, proximity = T, na.action = n
a.exclude)
               Type of random forest: classification
                     Number of trees: 50
No. of variables tried at each split: 5

        OOB estimate of  error rate: 7.08%
Confusion matrix:
    L   W class.error
L 318  25  0.07288630
W  21 286  0.06840391
>
```

```
Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      ntree = 60, importance = T, proximity = T, na.action = n
a.exclude)
               Type of random forest: classification
                     Number of trees: 60
No. of variables tried at each split: 5

        OOB estimate of  error rate: 7.38%
Confusion matrix:
    L   W class.error
L 315  28  0.08163265
W  20 287  0.06514658
>
```

9.6. **(2 points)** Determine the best value for mtry (use the number of trees you recommended in 9.5). What is the recommended value for mtry?

I would recommend mtry as 7 as it has lowest OOB error.(5.23%) (Number of ntree used is 30)

```
[1] 10
> mtry <- tuneRF(train_mydata[-26], train_mydata$gen_election, ntreeTry=30,
+                stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE, na.action=na.exclude)
mtry = 5  OOB error = 6.46%
Searching left ...
mtry = 4        OOB error = 7.38%
-0.1428571 0.01
Searching right ...
mtry = 7        OOB error = 5.23%
0.1904762 0.01
mtry = 10       OOB error = 7.23%
-0.3823529 0.01
> best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
> print(mtry)
      mtry   OOBError
4.OOB    4 0.07384615
5.OOB    5 0.06461538
7.OOB    7 0.05230769
10.OOB  10 0.07230769
> print(best.m)
[1] 7
```



9.7. **(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?

The OOB estimate of error rate is 7.38% with ntree = 30 and mtry = 7

```
> set.seed(32)
> rf <-randomForest(gen_election~., data=train_mydata, mtry=7, importance=TRUE, ntree=30)
> print(rf)

Call:
 randomForest(formula = gen_election ~ ., data = train_mydata,      mtry = 7, importance = TRUE, ntree = 30)
               Type of random forest: classification
                     Number of trees: 30
No. of variables tried at each split: 7

        OOB estimate of  error rate: 7.38%
Confusion matrix:
    L   W class.error
L 313  30  0.08746356
W  18 289  0.05863192
> |
```

9.8. **(8 points)** Use library(caret)[2] 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. Here is the code from class example in "R_model_evaluation.html":

```
library(caret)
predicted_values <- predict(rf, test_data,type= "prob")
threshold <- 0.5
pred <- factor( ifelse(predicted_values[,2] > threshold, 1, 0) )
levels(test_data$salary.class)[2]
confusionMatrix(pred, test_data$salary.class,
```

[2] You may get an error that will require you to use library(e1071). In this case, install this library using install.packages("e1071").

```
            positive = levels(test_data$salary.class)[2])
```

		Actual	
		W	L
Predicted	W	**115**	**13**
	L	**13**	**138**

```
> predicted_values <- predict(rf, test_mydata,type= "prob")
> threshold <- 0.5
> pred <- factor( ifelse(predicted_values[,2] > threshold, "w", "L") )
> levels(test_mydata$gen_election)
[1] "L" "W"
> View(test_mydata$gen_election)
> confusionMatrix(pred,test_mydata$gen_election,
+                 positive = levels(test_mydata$gen_election)[2])
Confusion Matrix and Statistics

          Reference
Prediction   L   W
         L 138  13
         W  13 115
```

```
          Reference
Prediction   L   W
         L 138  13
         W  13 115

               Accuracy : 0.9068
                 95% CI : (0.8664, 0.9382)
    No Information Rate : 0.5412
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.8123

 Mcnemar's Test P-Value : 1

            Sensitivity : 0.8984
            Specificity : 0.9139
         Pos Pred Value : 0.8984
         Neg Pred Value : 0.9139
             Prevalence : 0.4588
```

9.8.1. What is the value of accuracy? = = = > **0.9068 or 90.68%**

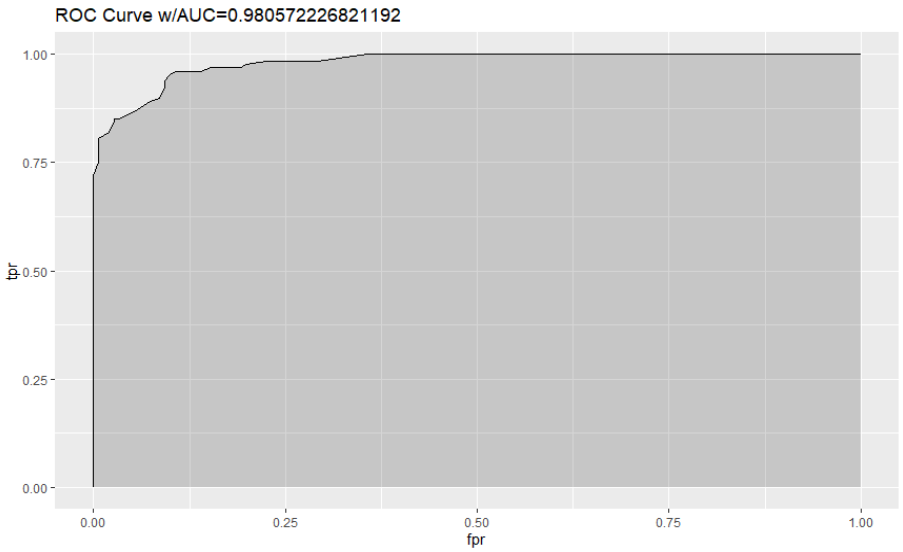
9.8.2. What is the value of TPR? = = = = = = > **TPR / Sensitivity = 0.8984 or 89.84%**

9.8.3. What is the value of FPR? = = = = = = = = > **FPR/ 1-Specifity = 1-0.9139 = 0.0861 = 8.6%**

9.9. **(4 points)** Use the code in "R_model_evaluation.html" to calculate AUC and create the ROC curve.

9.9.1. What is the value of AUC? = = = == = = > **AUC = 0.9805**

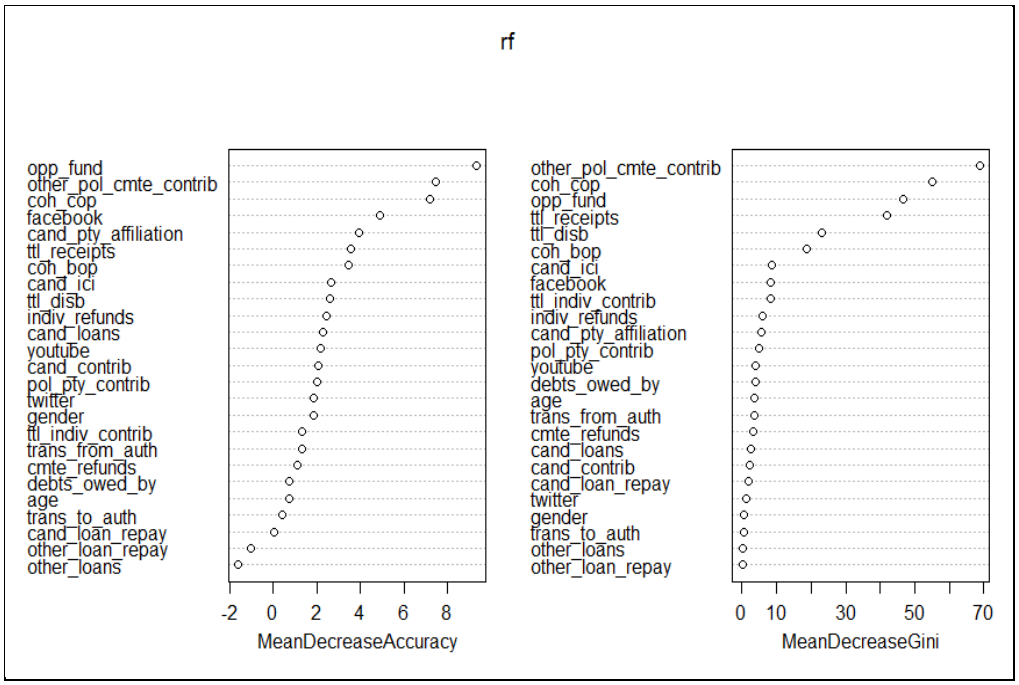
9.9.2. Paste the ROC curve in the space below:



9.10. **(4 points)** Use varImpPlot() to create the plot for variable importance. What are the type five important variables when we use MeanDecreaseAccuracy?

Top Five Important Variables:

1. Opp_Fund
2. Other_Pol_Cmte_Contrib
3. Coh_Cop
4. Facebook
5. Cand_Pty_Affliation



10. Build a neural network classifier.

10.1. **(20 points)** Use 5 hidden nodes in your ANN.

10.1.1. How many input nodes are in the ANN?

There are 39 input nodes.

10.1.2. How many weights are in the ANN?

There are 206 weights.

```
library("nnet")
ann<-nnet(gen_election~.,data=train_mydata,size=5,maxit=1000)
weights:  206
initial  value 648.414857
iter  10 value 222.019480
final  value 211.134819
converged
summary(ann)
39-5-1 network with 206 weights
options were - entropy fitting
 b->h1  i1->h1  i2->h1  i3->h1  i4->h1  i5->h1  i6->h1  i7->h1  i8->h1  i9->h1 i10->h1 i11->h1 i12->h1 i13->h1 i14->h1
 0.36    0.14   -0.51    0.04   -0.05   -0.01    0.38   -0.04    0.18    0.17    0.19    0.47   -0.19   -0.60   -0.17
i15->h1 i16->h1 i17->h1 i18->h1 i19->h1 i20->h1 i21->h1 i22->h1 i23->h1 i24->h1 i25->h1 i26->h1 i27->h1 i28->h1 i29->h1
 0.36   -0.51   -0.46    0.67    0.34    0.50   -0.62    0.41   -0.31    0.35    0.57    0.22    0.12    0.04    0.19
i30->h1 i31->h1 i32->h1 i33->h1 i34->h1 i35->h1 i36->h1 i37->h1 i38->h1 i39->h1
```

10.1.3. Use library(caret) 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.

		Actual	
		W	L
Predicted	W	**110**	**29**
	L	**18**	**122**

10.1.4. What is the value of sensitivity? = == = = = = > **0.8594**

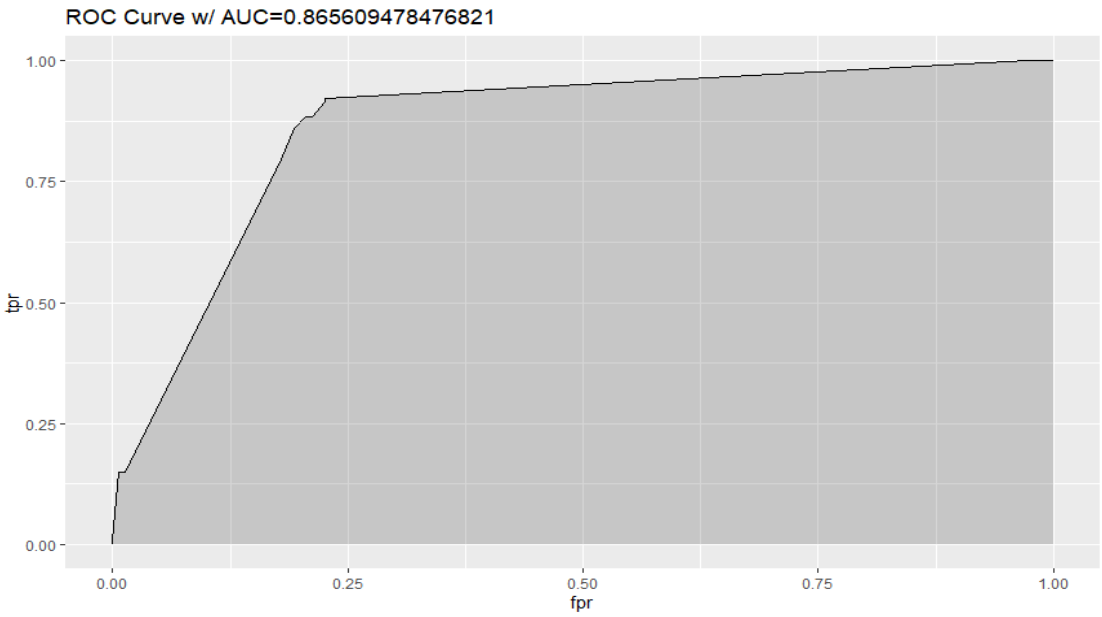
10.1.5. What is the value of specificity? = = = = = = = = = > **0.8079**

```
[1] "W"
> confusionMatrix(pred, test_mydata$gen_election,
+                 positive = levels(test_mydata$gen_election)[2])
Confusion Matrix and Statistics

          Reference
Prediction   L   W
         L 122  18
         W  29 110

               Accuracy : 0.8315
                 95% CI : (0.7824, 0.8735)
    No Information Rate : 0.5412
    P-Value [Acc > NIR] : <2e-16

                  Kappa : 0.663

 Mcnemar's Test P-Value : 0.1447

            Sensitivity : 0.8594
            Specificity : 0.8079
         Pos Pred Value : 0.7914
         Neg Pred Value : 0.8714
             Prevalence : 0.4588
```

10.1.6. Use the code in "R_model_evaluation.html" to calculate AUC and create the ROC curve.

10.1.6.1. What is the value of AUC? = = = = = = == = > **AUC = 0.8656**

10.1.6.2. Paste the ROC curve in the space below:



10.2. **(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.

10.2.1. What is the maximum number of hidden nodes that we could use?

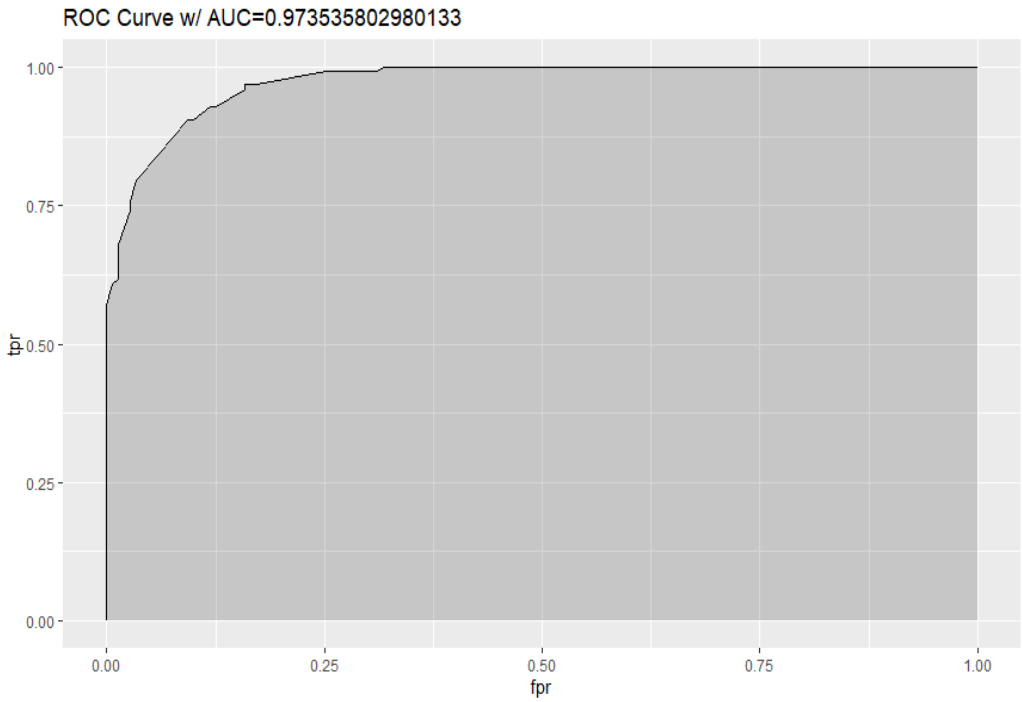
The maximum number of hidden nodes are 24.

```
> ann<-nnet(gen_election~.,data=train_mydata,size=25,maxit=1000)
Error in nnet.default(x, y, w, entropy = TRUE, ...) :
  too many (1026) weights
> ann<-nnet(gen_election~.,data=train_mydata,size=24,maxit=1000)
# weights:  985
initial  value 729.122896
iter  10 value 157.803051
iter  20 value 138.078201
iter  30 value 137.158541
iter  40 value 137.137932
iter  50 value 137.122980
final  value 137.122739
converged
>
```

10.2.2. Use the code in "R_model_evaluation.html" to calculate AUC and create the ROC curve.

10.2.2.1. What is the value of AUC? = = = = = = > **AUC = 0.9735**

10.2.2.2. Paste the ROC curve in the space below:



11. **(5 points)** Among the three classifiers that you built, which classifier would you finally use for predicting the election's outcome? Please explain.

    I would choose the classifier that has higher accuracy and AUC. Based on that, *the accuracy and AUC for Random forest are 90.68% and 0.9805 or 98.5% respectively*, which are higher than the other methods. Hence, I would recommend random forest classifier.

12. **(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.

    From, Ftable we can see that the candidates who invested more in social media campaign has won relatively better than those who have not invested. Of the three-social media, those invested in Facebook have greater chance of winning. Hence, I would recommend the candidates to invest in Facebook as probability of win is high.

```
1                    0 0
> ftable(xtabs(~facebook+gen_election, data=mydata))
          gen_election   L   W
facebook
0                      481 206
1                       13 229
> ftable(xtabs(~twitter+gen_election, data=mydata))
         gen_election   L   W
twitter
0                     480 250
1                      14 185
> ftable(xtabs(~youtube+gen_election, data=mydata))
         gen_election   L   W
youtube
0                     482 209
1                      12 226
> |
```

13. **(10 points)** Given your analysis, would you agree with this statement: "Money Buys Political Power"? Please explain.

    Yes. I agree with the statement. Based on our analysis from the model, we can determine that candidates who have good funding and invested the fund appropriately have won the general election. Having more funds enable the candidates to invest more in the appropriate mode of campaign and have a greater reach. For instance, in the given data, we can see that investing in the social media, which require huge funding, had greater impact on winning the election. Thus, the statement, "Money Buys Political Power" holds true for this data.

14. **(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.

    The dataset reveals that the funding and Investing that fund in the appropriate mode of campaign are the important factors that determined the wining. In other words, the probability of candidates who gets larger amount of fund and invest more on social media campaign had a higher winning rate. The important variables as per our mean_decrease_accuracy is Opp_Fund, Other_Pol_Cmte_Contrib, Coh_Cop, Facebook, Cand_Pty_Affliation also confirms the same. So, I would suggest the candidate to get more funding and invest more on social media campaign, especially in Facebook.

15. **(5 points)** Please paste your R code in the space below:

```
getwd()
setwd("C:/Users/Abdul Rahman/Documents/Train")
election_campaign_data <- read_csv("election_campaign_data.csv")
mydata <- read.csv("election_campaign_data.csv", sep=",", header=T, strip.white = T, na.strings =
c("NA","NaN","","?"))
nrow(mydata)
summary(mydata)
mydata$cand_id <- NULL
mydata$last_name <- NULL
mydata$first_name <- NULL
mydata$twitterbirth <-NULL
mydata$facebookdate <-NULL
mydata$facebookjan <-NULL
mydata$youtubebirth <-NULL

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)
summary(mydata)

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


ftable(xtabs(~facebook+gen_election, data=mydata))
ftable(xtabs(~twitter+gen_election, data=mydata))
ftable(xtabs(~youtube+gen_election, data=mydata))




set.seed(32)
n = nrow(mydata)
trainIndex = sample(1:n,
                    size = round(0.7*n),
                    replace=FALSE)
train_mydata = mydata[trainIndex,]
test_mydata = mydata[-trainIndex,]
summary(mydata)
summary(test_mydata)
summary(train_mydata)
View(mydata)
View(test_mydata)
View(train_mydata)

rf <-randomForest(gen_election~., data=train_mydata, ntree=10, na.action=na.exclude, importance=T,
                  proximity=T)
print(rf)

rf <-randomForest(gen_election~., data=train_mydata, ntree=20, na.action=na.exclude, importance=T,
                  proximity=T)
print(rf)

rf <-randomForest(gen_election~., data=train_mydata, ntree=30, na.action=na.exclude, importance=T,
                  proximity=T)
print(rf)

rf <-randomForest(gen_election~., data=train_mydata, ntree=40, na.action=na.exclude, importance=T,
```

```
            proximity=T)
print(rf)

rf <-randomForest(gen_election~., data=train_mydata, ntree=50, na.action=na.exclude, importance=T,
            proximity=T)
print(rf)

rf <-randomForest(gen_election~., data=train_mydata, ntree=60, na.action=na.exclude, importance=T,
            proximity=T)
print(rf)

rf <-randomForest(gen_election~., data=train_mydata, ntree=70, na.action=na.exclude, importance=T,
            proximity=T)
print(rf)

mtry <- tuneRF(train_mydata[-26], train_mydata$gen_election, ntreeTry=30,
          stepFactor=1.5, improve=0.01, trace=TRUE, plot=TRUE, na.action=na.exclude)
best.m <- mtry[mtry[, 2] == min(mtry[, 2]), 1]
print(mtry)
print(best.m)

set.seed(32)
rf <-randomForest(gen_election~., data=train_mydata, mtry=7, importance=TRUE, ntree=30)
print(rf)

library(caret)
library(e1071)
predicted_values <- predict(rf, test_mydata,type= "prob")
threshold <- 0.5
pred <- factor( ifelse(predicted_values[,2] > threshold, "W", "L") )
str(pred)
levels(test_mydata$gen_election)
View(test_mydata$gen_election)
confusionMatrix(pred,test_mydata$gen_election,
            positive = levels(test_mydata$gen_election)[2])

library(ROCR)
library(ggplot2)
predicted_values <- predict(rf, test_mydata,type= "prob")[,2]
pred <- prediction(predicted_values, test_mydata$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))
importance(rf)
varImpPlot(rf)

library("nnet")
ann<-nnet(gen_election~.,data=train_mydata,size=5,maxit=1000)
summary(ann)
print(ann)

predicted_values <- predict(ann, test_mydata,type= "raw")
head(predicted_values)
threshold <- 0.5
pred <- factor( ifelse(predicted_values[,1] > threshold, "W", "L") )
levels(test_mydata$gen_election)[2]
confusionMatrix(pred, test_mydata$gen_election,
                positive = levels(test_mydata$gen_election)[2])

ann<-nnet(gen_election~.,data=train_mydata,size=24,maxit=1000)


library(ggplot2)
predicted_values <- predict(ann, test_mydata,type= "raw")
pred <- prediction(predicted_values, test_mydata$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="ANN")
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))


ann_24 <- nnet(gen_election~.,data=test_mydata,size=24,maxit=1000)
```

```
predicted_values <- predict(ann_24, test_mydata,type= "raw")
pred <- prediction(predicted_values, test_mydata$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="ANN")
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))
```