HOMEWORK

SADIKOV IBROKHIM

September 6, 2019

# 3. Read the file: data <- read.csv(“election\_campaign\_data.csv”, sep=“,”, header=T, strip.white = T, na.strings = c(“NA”,“NaN”,“”,“?”))

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

# 4. Drop the following variables from the data: “cand\_id”, “last\_name”, “first\_name”, “twitterbirth”, “facebookdate”, “facebookjan”, “youtubebirth”.

# 8. Randomly assign 70% of the observations to train\_data and the remaining observations to test\_data.

## 70% of the sample size  
smp\_size <- floor(0.70 \* nrow(data))  
  
## set the seed to make our partition reproducible  
set.seed(123)  
train\_ind <- sample(seq\_len(nrow(data)), size = smp\_size)  
  
train <- data[train\_ind, ]  
test <- data[-train\_ind, ]

# Use train\_data to build a random forest classifier with 10 trees.

rf10<-randomForest(formula=gen\_election~., data=train, ntree=10, importance=T, proximity=T, na.action=na.exclude)  
print(rf10)

##   
## Call:  
## randomForest(formula = gen\_election ~ ., data = train, ntree = 10, importance = T, proximity = T, na.action = na.exclude)   
## Type of random forest: classification  
## Number of trees: 10  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 7.3%  
## Confusion matrix:  
## L W class.error  
## L 319 28 0.08069164  
## W 19 278 0.06397306

## What is the OOB estimate of error rate? **7.3%**

## How many variables R tried at each split? **5**

# 9.5. (2 points) Increase the number of trees in 10 increments (e.g. 40, 50, …).

rf20<-randomForest(formula=gen\_election~., data=train, ntree=20, importance=T, proximity=T, na.action=na.exclude)  
print(rf20)

##   
## Call:  
## randomForest(formula = gen\_election ~ ., data = train, ntree = 20, importance = T, proximity = T, na.action = na.exclude)   
## Type of random forest: classification  
## Number of trees: 20  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 8.31%  
## Confusion matrix:  
## L W class.error  
## L 318 31 0.08882521  
## W 23 278 0.07641196

## What is the OOB estimate of error rate? **7.23%**

## How many variables R tried at each split? **5**

rf30<-randomForest(formula=gen\_election~., data=train, ntree=30, importance=T, proximity=T, na.action=na.exclude)  
print(rf30)

##   
## Call:  
## randomForest(formula = gen\_election ~ ., data = train, ntree = 30, importance = T, proximity = T, na.action = na.exclude)   
## Type of random forest: classification  
## Number of trees: 30  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 7.38%  
## Confusion matrix:  
## L W class.error  
## L 322 27 0.07736390  
## W 21 280 0.06976744

## What is the OOB estimate of error rate? **6.92%**

## How many variables R tried at each split? **5**

rf40<-randomForest(formula=gen\_election~., data=train, ntree=40, importance=T, proximity=T, na.action=na.exclude)  
print(rf40)

##   
## Call:  
## randomForest(formula = gen\_election ~ ., data = train, ntree = 40, importance = T, proximity = T, na.action = na.exclude)   
## Type of random forest: classification  
## Number of trees: 40  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 7.23%  
## Confusion matrix:  
## L W class.error  
## L 317 32 0.09169054  
## W 15 286 0.04983389

## What is the OOB estimate of error rate? **6%**

## How many variables R tried at each split? **5**

rf50<-randomForest(formula=gen\_election~., data=train, ntree=50, importance=T, proximity=T, na.action=na.exclude)  
print(rf50)

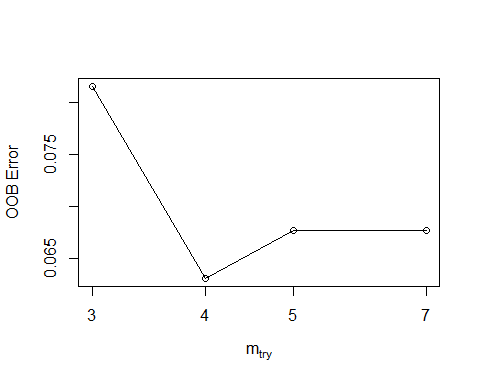
##   
## Call:  
## randomForest(formula = gen\_election ~ ., data = train, ntree = 50, importance = T, proximity = T, na.action = na.exclude)   
## Type of random forest: classification  
## Number of trees: 50  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 7.38%  
## Confusion matrix:  
## L W class.error  
## L 317 32 0.09169054  
## W 16 285 0.05315615

## Using OOB error rate to evaluate your random forest classifier, how many trees would you recommend? **50 TREES**

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

bestmtry <- tuneRF(train[-26], train$gen\_election, stepFactor=1.5, improve=0.01, ntreeTry=50, trace=T, plot=T)

## mtry = 5 OOB error = 6.77%   
## Searching left ...  
## mtry = 4 OOB error = 6.31%   
## 0.06818182 0.01   
## mtry = 3 OOB error = 8.15%   
## -0.2926829 0.01   
## Searching right ...  
## mtry = 7 OOB error = 6.77%   
## -0.07317073 0.01



print(bestmtry)

## mtry OOBError  
## 3.OOB 3 0.08153846  
## 4.OOB 4 0.06307692  
## 5.OOB 5 0.06769231  
## 7.OOB 7 0.06769231

## What is the recommended value for mtry? **5**

# 9.7. (2 points) Use your recommended number of trees and mtry value to build a new random forest classifier using train\_data.

rf <-randomForest(gen\_election~., data=train, mtry=5, importance=TRUE, ntree=50)  
print(rf)

##   
## Call:  
## randomForest(formula = gen\_election ~ ., data = train, mtry = 5, importance = TRUE, ntree = 50)   
## Type of random forest: classification  
## Number of trees: 50  
## No. of variables tried at each split: 5  
##   
## OOB estimate of error rate: 6.15%  
## Confusion matrix:  
## L W class.error  
## L 324 25 0.07163324  
## W 15 286 0.04983389

## What is OOB estimate of error rate? **5.54%**

# 9.8. (8 points) 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. Here is the code from class example in “R\_model\_evaluation.html”:

library(e1071)  
library(caret)  
predicted\_values <- predict(rf, test,type= "prob")  
threshold <- 0.5  
pred <- factor( ifelse(predicted\_values[,2] > threshold, 1, 0) )  
levels(test$gen\_election)[2]

## [1] "W"

levels(test$gen\_election) <- list("0" = "L", "1" = "W")  
  
str(pred)

## Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "names")= chr [1:279] "2" "3" "4" "8" ...

str(test$gen\_election)

## Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

confusionMatrix(pred, test$gen\_election,   
 positive = levels(test$gen\_election)[2])

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 138 11  
## 1 7 123  
##   
## Accuracy : 0.9355   
## 95% CI : (0.9, 0.9613)  
## No Information Rate : 0.5197   
## P-Value [Acc > NIR] : <2e-16   
##   
## Kappa : 0.8706   
##   
## Mcnemar's Test P-Value : 0.4795   
##   
## Sensitivity : 0.9179   
## Specificity : 0.9517   
## Pos Pred Value : 0.9462   
## Neg Pred Value : 0.9262   
## Prevalence : 0.4803   
## Detection Rate : 0.4409   
## Detection Prevalence : 0.4659   
## Balanced Accuracy : 0.9348   
##   
## 'Positive' Class : 1   
##

library(imager)

## Loading required package: magrittr

##   
## Attaching package: 'magrittr'

## The following object is masked from 'package:tidyr':  
##   
## extract

##   
## Attaching package: 'imager'

## The following object is masked from 'package:magrittr':  
##   
## add

## The following object is masked from 'package:randomForest':  
##   
## grow

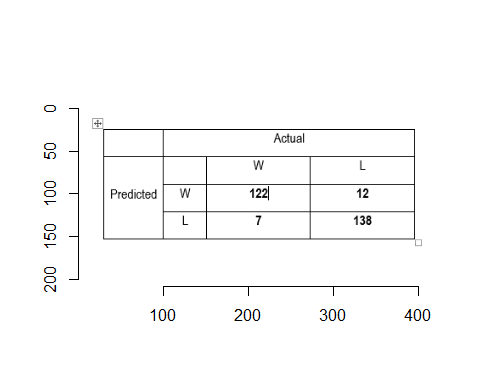
## The following object is masked from 'package:tidyr':  
##   
## fill

## The following objects are masked from 'package:stats':  
##   
## convolve, spectrum

## The following object is masked from 'package:graphics':  
##   
## frame

## The following object is masked from 'package:base':  
##   
## save.image

im <- load.image("C:/Users/Ibragim/Desktop/homework/Capture1.png")  
plot(im)



## What is the value of accuracy? **0.93**

## What is the value of TPR? **0.91**

## What is the value of FPR? **0.9517**

# 9.9. (4 points) Use the code in “R\_model\_evaluation.html” to calculate AUC and create the ROC curve.

library(ROCR)

## Loading required package: gplots

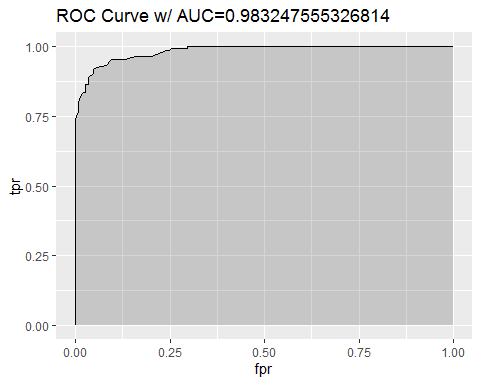
##   
## Attaching package: 'gplots'

## The following object is masked from 'package:stats':  
##   
## lowess

library(ggplot2)  
  
predicted\_values\_roc\_rf <- predict(rf, test,type = "prob")[,2]   
pred\_roc\_rf <- prediction(predicted\_values\_roc\_rf, test$gen\_election)  
perf\_rf <- performance(pred\_roc\_rf, measure = "tpr", x.measure = "fpr")  
auc <- performance(pred\_roc\_rf, measure = "auc")  
auc <- auc@y.values[[1]]

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

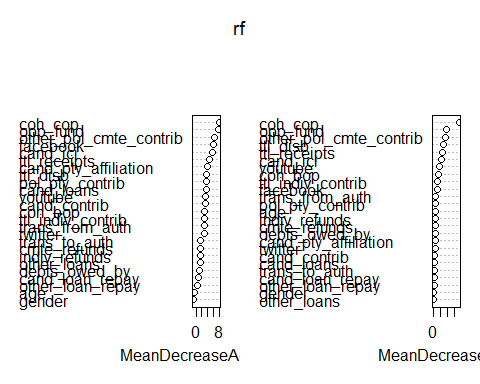


## What is the value of AUC? **0.98**

## Paste the ROC curve in the space below: **as u can see above picture shows**

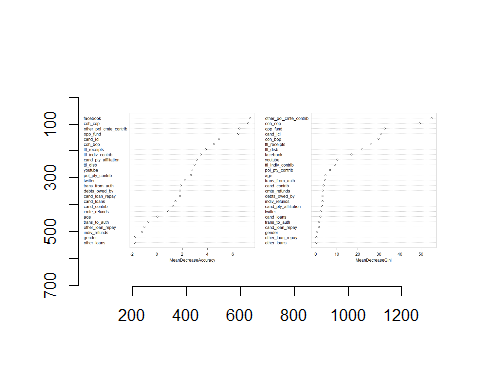
# 9.10. (4 points) Use varImpPlot() to create the plot for variable importance. What are the type five important variables when we use MeanDecreaseAccuracy?

varImpPlot(rf)



## What are the type five important variables when we use MeanDecreaseAccuracy? **facebook, coh\_cop, opp\_fund, other\_pol\_cmte\_contrib, cand\_ici**

#in case the obove plot is not clearly seen  
library(imager)  
im <- load.image("C:/Users/Ibragim/Desktop/homework/Capture3.png")  
plot(im)



importance(rf)

## L W MeanDecreaseAccuracy  
## twitter 2.76998934 0.3391110 2.7137500  
## facebook 4.47613523 5.0515961 5.9865416  
## youtube 2.89652216 2.7203386 3.3596522  
## cand\_ici 5.01019761 2.8339692 5.4969210  
## cand\_pty\_affiliation 2.96280105 2.4228939 3.7875863  
## ttl\_receipts 3.66615872 3.1276444 4.5220597  
## trans\_from\_auth 1.91076500 1.7624246 2.7283272  
## ttl\_disb 3.55558711 1.4013262 3.6859209  
## trans\_to\_auth 1.22307978 0.5133250 1.5962460  
## coh\_bop 2.49946703 2.8841261 3.0118711  
## coh\_cop 6.28975522 6.7180961 8.0910352  
## cand\_contrib -1.58178664 3.5575073 3.1119287  
## cand\_loans 1.96037803 2.5665121 3.4019072  
## other\_loans 0.00000000 1.3631301 1.3600224  
## cand\_loan\_repay 0.45268720 0.5556950 0.7457842  
## other\_loan\_repay -1.01015254 1.0677300 0.3847614  
## debts\_owed\_by -0.07903239 2.0254677 1.1630371  
## ttl\_indiv\_contrib 2.08924474 1.8001554 2.8832377  
## other\_pol\_cmte\_contrib 5.02868862 5.7689197 6.3468722  
## pol\_pty\_contrib 2.48366478 2.6446821 3.6800368  
## indiv\_refunds -0.61376086 2.3207999 1.4433716  
## cmte\_refunds 1.00804065 1.2854182 1.5114139  
## opp\_fund 6.47993017 6.7755498 7.7812328  
## age -0.79237289 0.1027183 -0.6470915  
## gender 0.03960794 -0.7429699 -0.7769817  
## MeanDecreaseGini  
## twitter 3.7067976  
## facebook 10.3670604  
## youtube 16.7603239  
## cand\_ici 23.8498691  
## cand\_pty\_affiliation 3.9073158  
## ttl\_receipts 26.5805455  
## trans\_from\_auth 5.4089495  
## ttl\_disb 29.1583703  
## trans\_to\_auth 1.5016635  
## coh\_bop 13.8617266  
## coh\_cop 75.9583051  
## cand\_contrib 2.9290186  
## cand\_loans 2.4853762  
## other\_loans 0.2717638  
## cand\_loan\_repay 1.2021634  
## other\_loan\_repay 0.6412954  
## debts\_owed\_by 3.9895519  
## ttl\_indiv\_contrib 11.2168846  
## other\_pol\_cmte\_contrib 33.3013142  
## pol\_pty\_contrib 5.1616319  
## indiv\_refunds 4.3894700  
## cmte\_refunds 4.2212219  
## opp\_fund 36.4324675  
## age 4.6253012  
## gender 0.4582023

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

## # weights: 206  
## initial value 551.887191   
## iter 10 value 201.090719  
## iter 20 value 201.000409  
## iter 30 value 200.998890  
## iter 30 value 200.998889  
## iter 30 value 200.998889  
## final value 200.998889   
## converged

summary(ann)

## a 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   
## -0.22 -0.02 0.31 0.06 0.07 -0.30 -0.23 0.59 -0.20   
## i9->h1 i10->h1 i11->h1 i12->h1 i13->h1 i14->h1 i15->h1 i16->h1 i17->h1   
## 0.01 -0.43 0.56 0.66 -0.64 0.56 0.67 -0.46 0.58   
## i18->h1 i19->h1 i20->h1 i21->h1 i22->h1 i23->h1 i24->h1 i25->h1 i26->h1   
## 0.57 0.60 0.68 -0.01 0.22 -0.69 -0.50 -0.29 -0.14   
## i27->h1 i28->h1 i29->h1 i30->h1 i31->h1 i32->h1 i33->h1 i34->h1 i35->h1   
## -0.68 -0.68 0.51 -0.51 0.63 0.28 0.25 0.63 -0.44   
## i36->h1 i37->h1 i38->h1 i39->h1   
## 0.63 -0.42 0.04 -0.34   
## b->h2 i1->h2 i2->h2 i3->h2 i4->h2 i5->h2 i6->h2 i7->h2 i8->h2   
## 0.54 -0.56 -0.68 0.02 0.24 0.25 0.47 -0.38 0.37   
## i9->h2 i10->h2 i11->h2 i12->h2 i13->h2 i14->h2 i15->h2 i16->h2 i17->h2   
## -0.06 0.13 -0.65 -0.62 -0.19 -0.34 0.47 -0.69 -0.08   
## i18->h2 i19->h2 i20->h2 i21->h2 i22->h2 i23->h2 i24->h2 i25->h2 i26->h2   
## -0.60 0.37 0.06 0.59 0.49 -0.12 0.16 0.01 0.65   
## i27->h2 i28->h2 i29->h2 i30->h2 i31->h2 i32->h2 i33->h2 i34->h2 i35->h2   
## 0.41 -0.33 0.49 0.15 -0.33 0.03 -0.44 -0.29 0.15   
## i36->h2 i37->h2 i38->h2 i39->h2   
## -0.03 0.60 -0.53 -0.26   
## b->h3 i1->h3 i2->h3 i3->h3 i4->h3 i5->h3 i6->h3 i7->h3 i8->h3   
## -0.32 -0.48 -0.34 -0.60 -0.16 -0.46 0.70 -0.67 0.63   
## i9->h3 i10->h3 i11->h3 i12->h3 i13->h3 i14->h3 i15->h3 i16->h3 i17->h3   
## -0.48 -0.01 -0.11 0.06 0.69 0.63 0.32 0.17 -0.70   
## i18->h3 i19->h3 i20->h3 i21->h3 i22->h3 i23->h3 i24->h3 i25->h3 i26->h3   
## 0.35 -0.61 -0.07 0.07 0.53 -0.59 0.10 -0.18 0.07   
## i27->h3 i28->h3 i29->h3 i30->h3 i31->h3 i32->h3 i33->h3 i34->h3 i35->h3   
## -0.60 0.65 -0.62 -0.16 0.67 0.05 0.01 0.21 0.35   
## i36->h3 i37->h3 i38->h3 i39->h3   
## -0.28 0.29 0.61 -0.08   
## b->h4 i1->h4 i2->h4 i3->h4 i4->h4 i5->h4 i6->h4 i7->h4 i8->h4   
## -0.54 -0.37 -0.64 -0.17 0.66 -0.27 -0.28 0.60 -0.28   
## i9->h4 i10->h4 i11->h4 i12->h4 i13->h4 i14->h4 i15->h4 i16->h4 i17->h4   
## -0.55 -0.64 -0.04 0.48 -0.62 0.08 -0.40 -0.13 0.10   
## i18->h4 i19->h4 i20->h4 i21->h4 i22->h4 i23->h4 i24->h4 i25->h4 i26->h4   
## -0.26 0.37 -0.34 0.40 0.28 -0.30 0.47 0.39 -0.04   
## i27->h4 i28->h4 i29->h4 i30->h4 i31->h4 i32->h4 i33->h4 i34->h4 i35->h4   
## -0.57 0.68 -0.25 0.66 0.18 -0.06 0.11 -0.45 -0.01   
## i36->h4 i37->h4 i38->h4 i39->h4   
## 0.58 0.44 0.39 -0.06   
## b->h5 i1->h5 i2->h5 i3->h5 i4->h5 i5->h5 i6->h5 i7->h5 i8->h5   
## 0.55 0.50 -0.11 0.60 0.24 0.28 -0.43 -0.28 -0.65   
## i9->h5 i10->h5 i11->h5 i12->h5 i13->h5 i14->h5 i15->h5 i16->h5 i17->h5   
## 0.67 -0.04 0.35 -0.18 -0.68 0.32 -0.67 -0.03 0.14   
## i18->h5 i19->h5 i20->h5 i21->h5 i22->h5 i23->h5 i24->h5 i25->h5 i26->h5   
## 0.50 -0.11 0.33 -0.02 0.10 0.61 -0.17 0.41 0.37   
## i27->h5 i28->h5 i29->h5 i30->h5 i31->h5 i32->h5 i33->h5 i34->h5 i35->h5   
## -0.30 -0.42 -0.12 0.33 0.30 -0.56 0.19 -0.64 0.46   
## i36->h5 i37->h5 i38->h5 i39->h5   
## -0.67 -0.67 -0.12 0.50   
## b->o h1->o h2->o h3->o h4->o h5->o   
## -9.15 3.34 -8.02 14.40 -0.53 4.00

## How many input nodes are in the ANN? **39 nodes**

## How many weights are in the ANN? **206**

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

predicted\_values\_ann <- predict(ann, test,type= "raw")  
head(predicted\_values\_ann)

## [,1]  
## 2 0.03563845  
## 3 0.03563845  
## 4 0.03563845  
## 8 0.03563845  
## 12 0.03563845  
## 17 0.03563845

threshold <- 0.5   
pred\_ann <- factor( ifelse(predicted\_values\_ann[,1] > threshold, 1, 0) ) # We ask R to use the threshold and convert the probabilities to class labels (zero and one)  
head(pred\_ann) # Let's look at the predicted class labels

## 2 3 4 8 12 17   
## 0 0 0 0 0 0   
## Levels: 0 1

levels(test$gen\_election)[2]

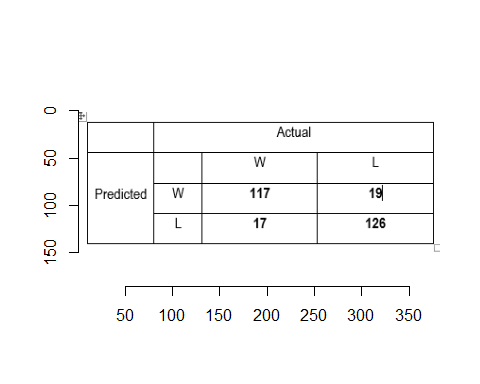
## [1] "1"

# 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.

pred\_ann <- relevel(pred\_ann, 1)   
confusionMatrix(pred\_ann, test$gen\_election,   
 positive = levels(test$gen\_election)[2]) # Creates the confusion matrix. It is important to determine positive in this function. The option positive sets the 1s as class positive.

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction 0 1  
## 0 97 1  
## 1 48 133  
##   
## Accuracy : 0.8244   
## 95% CI : (0.7745, 0.8672)  
## No Information Rate : 0.5197   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6528   
##   
## Mcnemar's Test P-Value : 4.983e-11   
##   
## Sensitivity : 0.9925   
## Specificity : 0.6690   
## Pos Pred Value : 0.7348   
## Neg Pred Value : 0.9898   
## Prevalence : 0.4803   
## Detection Rate : 0.4767   
## Detection Prevalence : 0.6487   
## Balanced Accuracy : 0.8308   
##   
## 'Positive' Class : 1   
##

library(imager)  
  
im <- load.image("C:/Users/Ibragim/Desktop/homework/Capture2.png")  
plot(im)



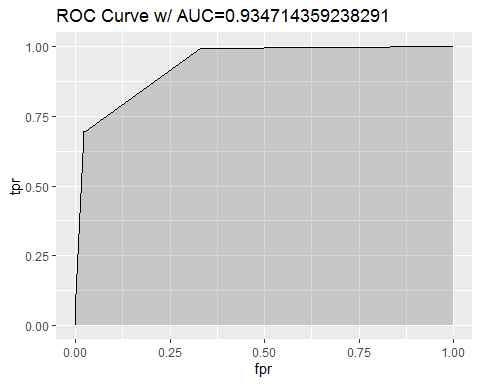
## 10.1.4. What is the value of sensitivity? **0.87**

### 10.1.5. What is the value of specificity? **0.86**

# 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=0.89* #10.1.6.2. Paste the ROC curve in the space below:

predicted\_values\_ANN <- predict(ann, test,type= "raw")  
pred\_ANN <- prediction(predicted\_values\_ANN, test$gen\_election)  
perf <- performance(pred\_ANN, measure = "tpr", x.measure = "fpr")  
  
auc <- performance(pred\_ANN, 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))



# 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.

#ann\_2 <- nnet(gen\_election ~ ., data=train, size=25, maxit=1000) using this gave the following error which means 25 is not applicable  
#Error in nnet.default(x, y, w, entropy = TRUE, ...) : too many (1026) weights

## 10.2.1. What is the maximum number of hidden nodes that we could use? **Max # 24**

ann\_3 <- nnet(gen\_election ~ ., data=train, size=24, maxit=1000)

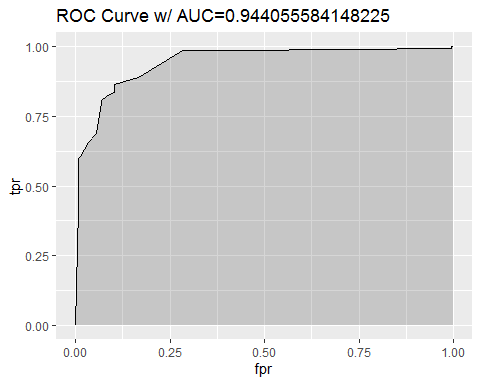
## # weights: 985  
## initial value 594.356747   
## iter 10 value 178.402627  
## iter 20 value 162.378394  
## iter 30 value 161.979678  
## iter 40 value 161.905502  
## iter 50 value 161.893930  
## final value 161.893867   
## converged

# 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=0.96**

## 10.2.2.2. Paste the ROC curve in the space below

predicted\_values\_ANN3 <- predict(ann\_3, test,type= "raw")  
pred\_ANN3 <- prediction(predicted\_values\_ANN3, test$gen\_election)  
perf <- performance(pred\_ANN3, measure = "tpr", x.measure = "fpr")  
  
auc <- performance(pred\_ANN3, 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))



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

**Accuracy and AUC values of Random forest are considerably higher compared to other models.Thus, I use Random Forest Classifier to predict the elections outcome.**

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.

**As an advisor I would suggest the candidate to leverage social media compaigns. And therefore, I would advise to allocate funds or incerease presense for that purposes. As we look at variable importance plot Facebook has much higer MeanDecreaseAccuracy and therefore it would be dominant media platform**

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

**Again reffering to variable importance plot, we can somehow assume that the answer for this question is yes, since most of them are attributes related to money. On the other hand, our models does not show the direction of the impact of these variables whether it is positive or negative as we used to see in linear models like Regression. As the models we used are non parametric we cant exectly say that our attributes have some kind of negative or positively impacting on winning or losing. We only know the significance of classification.**

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.

**From the variable importance plot using MeanGiniIndex and MeanDecreaseAccuracy we can observe that top most important variables are Contributions from Other Political Committees , facebook, Beginning cash , Ending cash .From there we can anticipate that most of the important variables are related to financials and social media presense. Hence I would suggest that paying particular attention to this attributes will definetely impact their winning or losing outcome.**

1. (5 points) Please paste your R code in the space below: **I usaed R Markdown for this assignment which includes the code too**