# Classification Trees Assignment

## BAN502 Predictive Analytics

### Susan Wiggins

The following libraries needed:

library(tidyverse)

## -- Attaching packages ---------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.1.1 v purrr 0.3.2  
## v tibble 2.1.1 v dplyr 0.8.1  
## v tidyr 0.8.3 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts ------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(rpart)  
library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

Reading in the data set:

library(readr)  
parole <- read\_csv("parole.csv")

## Parsed with column specification:  
## cols(  
## male = col\_double(),  
## race = col\_double(),  
## age = col\_double(),  
## state = col\_double(),  
## time.served = col\_double(),  
## max.sentence = col\_double(),  
## multiple.offenses = col\_double(),  
## crime = col\_double(),  
## violator = col\_double()  
## )

View(parole)

parole = parole %>% mutate(male = as\_factor(as.character(male)))%>%   
mutate(male = fct\_recode(male, "male" = "1", "female" = "0"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <dbl> 1, 1, 2, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 1,...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <dbl> 4, 3, 3, 1, 1, 4, 3, 1, 3, 2, 1, 1, 3, 3, 3,...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(race = as\_factor(as.character(race)))%>%   
mutate(race = fct\_recode(race, "white" = "1", "otherwise" = "2"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <dbl> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <dbl> 4, 3, 3, 1, 1, 4, 3, 1, 3, 2, 1, 1, 3, 3, 3,...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(state = as\_factor(as.character(state)))%>%   
mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "other state" = "1"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <dbl> 4, 3, 3, 1, 1, 4, 3, 1, 3, 2, 1, 1, 3, 3, 3,...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(crime = as\_factor(as.character(crime)))%>%   
mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related crime" = "3", "driving-related crime" = "4", "other crime" = "1"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <fct> driving-related crime, drug-related crime, d...  
## $ violator <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,...

parole = parole %>% mutate(violator = as\_factor(as.character(violator)))%>%   
mutate(violator = fct\_recode(violator, "Yes" = "1", "No" = "0"))  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <dbl> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,...  
## $ crime <fct> driving-related crime, drug-related crime, d...  
## $ violator <fct> No, No, No, No, No, No, No, No, No, No, No, ...

parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses)))%>%   
mutate(multiple.offenses = fct\_recode(multiple.offenses, "Multiple Offenses" = "1", "Otherwise" = "0"))  
  
  
glimpse(parole)

## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> other state, other state, other state, other...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <dbl> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <fct> Otherwise, Otherwise, Otherwise, Otherwise, ...  
## $ crime <fct> driving-related crime, drug-related crime, d...  
## $ violator <fct> No, No, No, No, No, No, No, No, No, No, No, ...

### Task 1

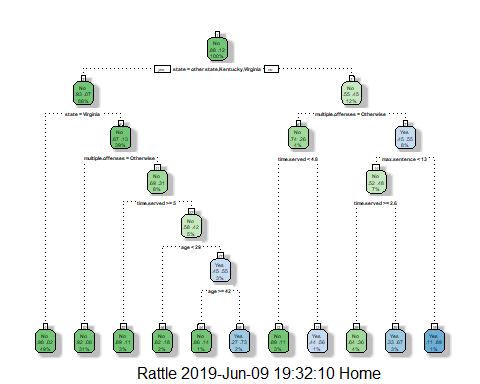
Split the data into training and testing sets. Your training set should have 70% of the data. Use a random number (set.seed) of 12345.

set.seed(12345)   
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE) #70% in training  
train = parole[train.rows,]   
test = parole[-train.rows,]

### Task 2

Create a classification tree to predict “violator” in the training set. Plot the tree.

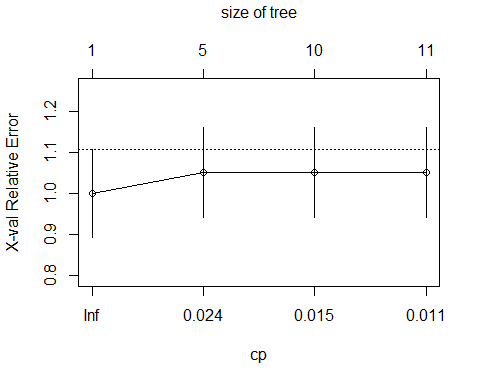
tree1 = rpart(violator~., parole, method="class")  
fancyRpartPlot(tree1)



#Task4  
  
printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = parole, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age max.sentence multiple.offenses state   
## [5] time.served   
##   
## Root node error: 78/675 = 0.11556  
##   
## n= 675   
##   
## CP nsplit rel error xerror xstd  
## 1 0.032051 0 1.00000 1.0000 0.10648  
## 2 0.017949 4 0.83333 1.0513 0.10881  
## 3 0.012821 9 0.74359 1.0513 0.10881  
## 4 0.010000 10 0.73077 1.0513 0.10881

plotcp(tree1)



### Task 3

For the tree created in Task 2, how would you classify a 40 year-old parolee from Louisiana who served a 5 year prison sentence? Describe how you “walk through” the classification tree to arrive at your answer.

State = other state, Kentucky, and Virginia, this is the root of the tree so I said No to this statement and went to the right of the tree structure. Next, I assumed that the person did not serve for mutiple offenses so I choose No then moved to the left side of the tree at this leaf. Next, time server < 4.5 was more than this value so I this was no then moved to the next left indicated as no: therefore, the 40-year old parolee from Louisiana who served a 5 year sentence would not violate his or her parole.

### Task 4

Use the plotcp and printcp functions to evaluate tree performance as a function of the complexity parameter (cp). Pay close attention as this cp plot will look different than others we have seen. What cp value should be selected?

The optimal cp value looks like it should be .01

Just wanted to practice:

#tree2 = rpart(violator ~., parole, cp=0.0001, method="class")  
#fancyRpartPlot(tree2)

#printcp(tree2)  
#plotcp(tree2)

### Task 5

Prune the tree from Task 2 back to the cp value that you selected in Task 4. Do not attempt to plot the tree. The resulting tree is known as a “root”. A tree that takes the form of a root is essentially a naive model that assumes that the prediction for all observations is the majority class. Which class (category) in the training set is the majority class (i.e., has the most observations)?

I believe the class with the most observations are the observation listed as the root node which is state in this tree model.

tree2 = prune(tree1,cp= tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
  
summary(tree2)

## Call:  
## rpart(formula = violator ~ ., data = parole, method = "class")  
## n= 675   
##   
## CP nsplit rel error xerror xstd  
## 1 0.03205128 0 1 1 0.1064849  
##   
## Node number 1: 675 observations  
## predicted class=No expected loss=0.1155556 P(node) =1  
## class counts: 597 78  
## probabilities: 0.884 0.116

### Task 6

Use the unpruned tree from Task 2 to develop predictions for the training data. Use caret’s confusionMatrix function to calculate the accuracy, specificity, and sensitivty of this tree on the training data.

The confusionMatrix function to calculate the accuracy as 0.9197 which is 91.97% accuarate.

The confusionMatrix function to calculate the specificity as 0.5091 which is 50.91% specificity.

The confusionMatrix function to calculate the sensitivty as 0.9737 which has 97.37% sensitivity.

treepred\_parole = predict(tree1, train, type = "class")  
head(treepred\_parole)

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

confusionMatrix(treepred\_parole,train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 407 27  
## Yes 11 28  
##   
## Accuracy : 0.9197   
## 95% CI : (0.8914, 0.9425)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.006833   
##   
## Kappa : 0.5526   
##   
## Mcnemar's Test P-Value : 0.014961   
##   
## Sensitivity : 0.9737   
## Specificity : 0.5091   
## Pos Pred Value : 0.9378   
## Neg Pred Value : 0.7179   
## Prevalence : 0.8837   
## Detection Rate : 0.8605   
## Detection Prevalence : 0.9175   
## Balanced Accuracy : 0.7414   
##   
## 'Positive' Class : No   
##

### Task 7

Use the unpruned tree from Task 2 to develop predictions for the testing data. Use caret’s confusionMatrix function to calculate the accuracy, specificity, and sensitivty of this tree on the testing data. Comment on the quality of the model.

The confusionMatrix function to calculate the accuracy as 0.9059 which is 90.59% accuarate.

The confusionMatrix function to calculate the specificity as 0.3478 which is 34.78% specificity.

The confusionMatrix function to calculate the sensitivty as 0.9777 which is 97.77% sensitivity.

treepred\_parole\_test = predict(tree1, test, type = "class")  
head(treepred\_parole\_test)

## 1 2 3 4 5 6   
## No No No No No No   
## Levels: No Yes

confusionMatrix(treepred\_parole\_test,test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 175 15  
## Yes 4 8  
##   
## Accuracy : 0.9059   
## 95% CI : (0.857, 0.9424)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.22246   
##   
## Kappa : 0.4112   
##   
## Mcnemar's Test P-Value : 0.02178   
##   
## Sensitivity : 0.9777   
## Specificity : 0.3478   
## Pos Pred Value : 0.9211   
## Neg Pred Value : 0.6667   
## Prevalence : 0.8861   
## Detection Rate : 0.8663   
## Detection Prevalence : 0.9406   
## Balanced Accuracy : 0.6627   
##   
## 'Positive' Class : No   
##

### Task 8

Read in the “Blood.csv” dataset. The dataset contains five variables: Mnths\_Since\_Last: Months since last donation TotalDonations: Total number of donation Total\_Donated: Total amount of blood donated Mnths\_Since\_First: Months since first donation DonatedMarch: Binary variable representing whether he/she donated blood in March (1 = Yes, 0 = No) Convert the DonatedMarch variable to a factor and recode the variable so 0 = “No” and 1 = “Yes”.

Read the Blood.csv file into the assignment:

library(readr)  
Blood <- read\_csv("Blood.csv")

## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

View(Blood)  
  
Blood = Blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch)))%>%   
mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1"))  
  
glimpse(Blood)

## Observations: 748  
## Variables: 5  
## $ Mnths\_Since\_Last <dbl> 2, 0, 1, 2, 1, 4, 2, 1, 2, 5, 4, 0, 2, 1, 2,...  
## $ TotalDonations <dbl> 50, 13, 16, 20, 24, 4, 7, 12, 9, 46, 23, 3, ...  
## $ Total\_Donated <dbl> 12500, 3250, 4000, 5000, 6000, 1000, 1750, 3...  
## $ Mnths\_Since\_First <dbl> 98, 28, 35, 45, 77, 4, 14, 35, 22, 98, 58, 4...  
## $ DonatedMarch <fct> Yes, Yes, Yes, Yes, No, No, Yes, No, Yes, Ye...

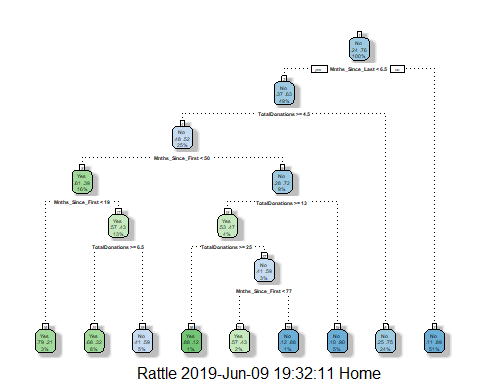
### Task 9

Split the dataset into training (70%) and testing (30%) sets. You may wish to name your training and testing sets “train2” and “test2” as to not confuse them with the parole datsets Use set.seed of 1234. Then develop a classification tree on the training set to predict “DonatedMarch”. Evaluate the complexity parameter (cp) selection for this model.

The optimal cp value looks like it should be .01

set.seed(1234)   
train.rows = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
trainblood = Blood[train.rows,]   
testblood = Blood[-train.rows,]

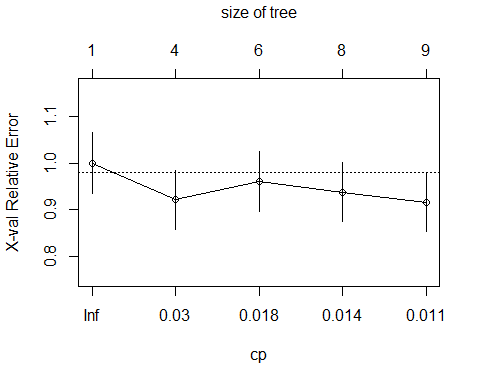
treeblood = rpart(DonatedMarch~., Blood, method="class")  
fancyRpartPlot(treeblood)



printcp(treeblood)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = Blood, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 178/748 = 0.23797  
##   
## n= 748   
##   
## CP nsplit rel error xerror xstd  
## 1 0.046816 0 1.00000 1.00000 0.065430  
## 2 0.019663 3 0.85955 0.92135 0.063571  
## 3 0.016854 5 0.82022 0.96067 0.064523  
## 4 0.011236 7 0.78652 0.93820 0.063985  
## 5 0.010000 8 0.77528 0.91573 0.063431

plotcp(treeblood)

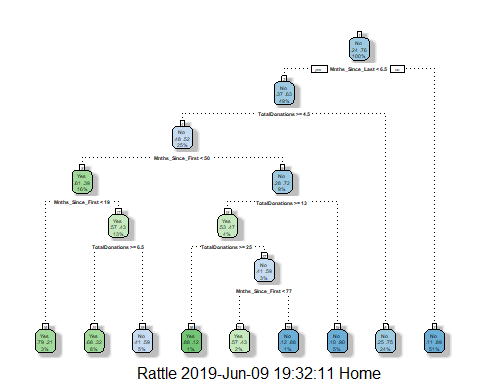


### Task 10

Prune the tree back to the optimal cp value, make predictions, and use the confusionMatrix function on the both training and testing sets. Comment on the quality of the predictions.

Training set:

tree3 = prune(treeblood,cp= treeblood$cptable[which.min(treeblood$cptable[,"xerror"]),"CP"])  
#most of the code in the line above can be left untouched. Just change tree1 to the name of your tree model (if it's not called tree1)  
fancyRpartPlot(tree3)



treepred = predict(tree3, trainblood, type = "class")  
head(treepred)

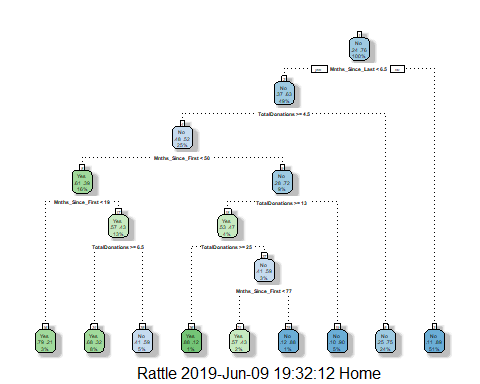
## 1 2 3 4 5 6   
## Yes Yes No No Yes Yes   
## Levels: Yes No

confusionMatrix(treepred,trainblood$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 53 19  
## No 72 380  
##   
## Accuracy : 0.8263   
## 95% CI : (0.7911, 0.8578)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.0001954   
##   
## Kappa : 0.4405   
##   
## Mcnemar's Test P-Value : 5.006e-08   
##   
## Sensitivity : 0.4240   
## Specificity : 0.9524   
## Pos Pred Value : 0.7361   
## Neg Pred Value : 0.8407   
## Prevalence : 0.2385   
## Detection Rate : 0.1011   
## Detection Prevalence : 0.1374   
## Balanced Accuracy : 0.6882   
##   
## 'Positive' Class : Yes   
##

Testing set:

tree4 = prune(treeblood,cp= treeblood$cptable[which.min(treeblood$cptable[,"xerror"]),"CP"])  
#most of the code in the line above can be left untouched. Just change tree1 to the name of your tree model (if it's not called tree1)  
fancyRpartPlot(tree3)



treepred\_blood\_test = predict(tree4, testblood, type = "class")  
head(treepred)

## 1 2 3 4 5 6   
## Yes Yes No No Yes Yes   
## Levels: Yes No

confusionMatrix(treepred\_blood\_test,testblood$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 17 11  
## No 36 160  
##   
## Accuracy : 0.7902   
## 95% CI : (0.7309, 0.8416)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.1945593   
##   
## Kappa : 0.3063   
##   
## Mcnemar's Test P-Value : 0.0004639   
##   
## Sensitivity : 0.32075   
## Specificity : 0.93567   
## Pos Pred Value : 0.60714   
## Neg Pred Value : 0.81633   
## Prevalence : 0.23661   
## Detection Rate : 0.07589   
## Detection Prevalence : 0.12500   
## Balanced Accuracy : 0.62821   
##   
## 'Positive' Class : Yes   
##

### Task 10 (Continued)

Comment on the quality of the predictions.

The training set tree has a 82.63% accuracy for the model to be correct whereas a naive model would be 76.15% assuming that everyone gave blood or did not give blood. The testing set tree has a 79.02% accuracy for the model to be correct and the naive model would be at 76.34% for the same assumption previously stated. The training set seems to have a significant difference over the testing set when looking at the p-value due to being less than .05. The training set is 0.0001954 for the p-value and the testing set is listed at 0.1945593.