## Module 4 Assignment 1 - Classification Trees

### Nicholas Barrett, 02/16/19

options(tidyverse.quiet = TRUE)  
library(tidyverse)  
library(caret)

## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(rpart)

## Warning: package 'rpart' was built under R version 3.5.2

library(rattle)

## Warning: package 'rattle' was built under R version 3.5.2

## 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)

## Warning: package 'RColorBrewer' was built under R version 3.5.2

parole = read\_csv("parole.csv")

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

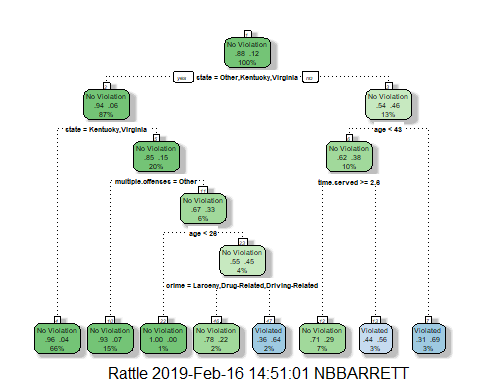
parole = parole %>% mutate(male = as.factor(male)) %>%   
 mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1" ))  
parole = parole %>%mutate(race = as.factor(race)) %>%  
 mutate(race = fct\_recode(race, "Other" = "2", "White" = "1" ))  
parole = parole %>% mutate(state = as.factor(state))%>%  
 mutate(state = fct\_recode(state, "Kentucky"= "2", "Lousiana"= "3", "Virginia"= "4", "Other"= "1" ))  
parole = parole %>% mutate(crime = as.factor(crime))%>%  
 mutate(crime = fct\_recode(crime, "Larceny"= "2", "Drug-Related"= "3", "Driving-Related"= "4", "Other"= "1" ))  
parole = parole %>% mutate(multiple.offenses = as.factor(multiple.offenses))%>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "Incarcerated" = "1", "Other" = "0" ))  
parole = parole %>% mutate(violator = as.factor(violator))%>%  
 mutate(violator = fct\_recode(violator, "Violated" = "1", "No Violation" = "0" ))

### Task 1 - Split the training and testing sets. Training set should have 70% of data and use a random number of 12345

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

### Task 2 - Create a classification tree to predict “violator” in training set. Plot tree.

traintree1 = rpart(violator ~., train, method = "class")  
fancyRpartPlot(traintree1)



### Task 3 - For above tree, classify a 40-year-old parolee from Lousiana who served a 5 year prison sentence. Walk through the classification tree to describe the classfication of the LA parolee under those parameters.

As 40-year-old parolee from Louisiana who served a 5 year prison sentence would be predicted to have a 3% chance of violating parole.

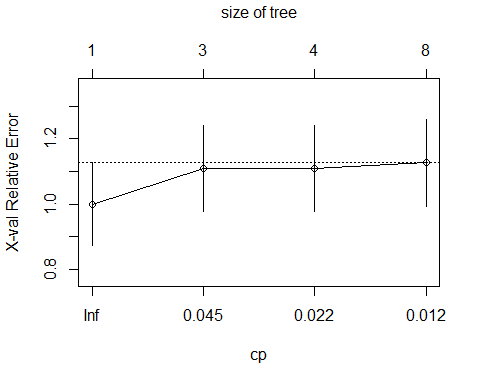
13% of parolees are from states not Other, Kentucky or Virgina. 10% of those parolees are younger than 43-years-old. 3% of those who served more than or equal to 2.6 years violated parole.

### Task 4 - Use plotcp and printcp to evaluate tree performance as a function of the complexity parameter (cp). Determine what cp value should be selected.

printcp(traintree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age crime multiple.offenses state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.054545 0 1.00000 1.0000 0.12676  
## 2 0.036364 2 0.89091 1.1091 0.13253  
## 3 0.013636 3 0.85455 1.1091 0.13253  
## 4 0.010000 7 0.80000 1.1273 0.13345

plotcp(traintree1)



The cp value that has the lowest cross-validated error is a cp of 0.054.

### Task 5 - Prune the tree from Task 2 with the best cp value without plotting the tree.

traintree2 = prune(traintree1, cp = traintree1$cptable[which.min(traintree1$cptable[,"xerror"]), "CP"])  
summary(traintree2)

## Call:  
## rpart(formula = violator ~ ., data = train, method = "class")  
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.05454545 0 1 1 0.1267582  
##   
## Node number 1: 473 observations  
## predicted class=No Violation expected loss=0.1162791 P(node) =1  
## class counts: 418 55  
## probabilities: 0.884 0.116

The majority class in the training set is No Violation with 418 observations.

### Task 6 - Use the tree from Task 2 to develop predictions for the testing data using caret’s confusionMatrix function to calculate accuraracy, specificity, and sensitivity of the tree on the testing data. Comment on the quality of the model.

traintree1pred = predict(traintree1, train, type = "class")  
head(traintree1pred)

## 1 2 3 4 5   
## No Violation No Violation No Violation No Violation No Violation   
## 6   
## No Violation   
## Levels: No Violation Violated

confusionMatrix(traintree1pred,train$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violated  
## No Violation 402 28  
## Violated 16 27  
##   
## Accuracy : 0.907   
## 95% CI : (0.8771, 0.9316)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.06272   
##   
## Kappa : 0.5   
## Mcnemar's Test P-Value : 0.09725   
##   
## Sensitivity : 0.49091   
## Specificity : 0.96172   
## Pos Pred Value : 0.62791   
## Neg Pred Value : 0.93488   
## Prevalence : 0.11628   
## Detection Rate : 0.05708   
## Detection Prevalence : 0.09091   
## Balanced Accuracy : 0.72632   
##   
## 'Positive' Class : Violated   
##

The classification tree for the training data set is 90.7% with a specificity of 96.1% and a sensitivity of 49%.

### Task 7 - Use the unpruned tree from Task 2 to develop predictions for the testing data. Use caret’s confusionMatrix function to calculate accuracy, specificity, and sensitivity of the tree on the testing data. Comment on the quality of the model.

traintree2pred = predict(traintree1, test, type = "class")  
head(traintree2pred)

## 1 2 3 4 5   
## No Violation Violated No Violation No Violation No Violation   
## 6   
## No Violation   
## Levels: No Violation Violated

confusionMatrix(traintree2pred,test$violator, positive = "Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violated  
## No Violation 170 19  
## Violated 9 4  
##   
## Accuracy : 0.8614   
## 95% CI : (0.8059, 0.9059)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.88631   
##   
## Kappa : 0.1525   
## Mcnemar's Test P-Value : 0.08897   
##   
## Sensitivity : 0.17391   
## Specificity : 0.94972   
## Pos Pred Value : 0.30769   
## Neg Pred Value : 0.89947   
## Prevalence : 0.11386   
## Detection Rate : 0.01980   
## Detection Prevalence : 0.06436   
## Balanced Accuracy : 0.56182   
##   
## 'Positive' Class : Violated   
##

The classifcation tree of the testing data has an accuracy of 86.1% but a naive rate of 88.6%. So, the classification tree of the testing set is actually not as good a predictor than simply assuming everyone falls in the majority class of “No Violation”. So, this model is not very good.

### Task 8 - Read in the “Blood.csv” dataset. Convert the DonatedMarch variable to a factor and recode the variable so 0= “No” and 1 = “Yes”.

blood = read\_csv("Blood.csv")

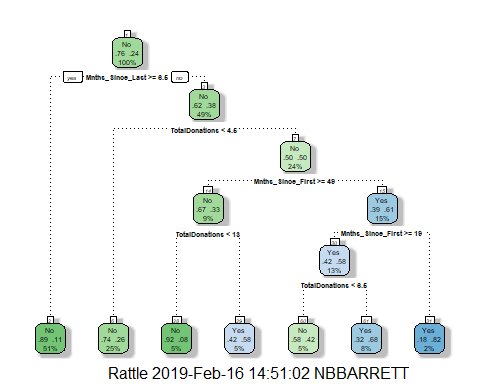
## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_integer(),  
## TotalDonations = col\_integer(),  
## Total\_Donated = col\_integer(),  
## Mnths\_Since\_First = col\_integer(),  
## DonatedMarch = col\_integer()  
## )

blood = blood %>% mutate(DonatedMarch = as.factor(DonatedMarch))%>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "Yes" = "1", "No" = "0"))

### Task 9 - Split the dataset into a training set and testing set. Use set seed 1234. Develop a classification tree on the training set to predict “DonatedMarch”. Evaluate the parameter (cp) selection for the model.

set.seed(1234)  
  
train.rows = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE)   
train2 = blood[train.rows,]   
test2 = blood[-train.rows,]

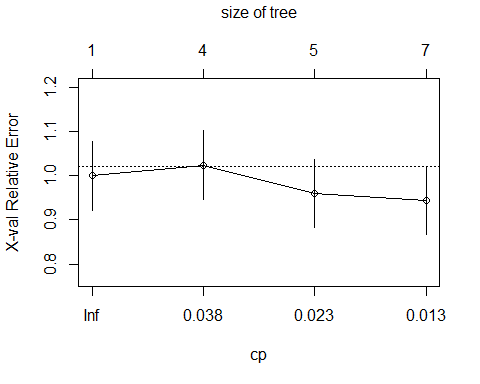
train2tree = rpart(DonatedMarch ~., train2, method = "class")  
fancyRpartPlot(train2tree)



printcp(train2tree)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.045333 0 1.000 1.000 0.078049  
## 2 0.032000 3 0.864 1.024 0.078682  
## 3 0.016000 4 0.832 0.960 0.076949  
## 4 0.010000 6 0.800 0.944 0.076494

plotcp(train2tree)



The cp value that has the lowest cross-validated error is a cp of 0.01 with a 0.944 error.

### Task 10 - Prune the tree back to the optimal cp value, make predictions and use the confusionMatrix function on the training and testing sets. Comment on the quality of the predictions.

train2tree2 = prune(train2tree, cp = train2tree$cptable[which.min(train2tree$cptable[,"xerror"]), "CP"])

train2tree2pred = predict(train2tree2, test2, type = "class")  
head(train2tree2pred)

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

confusionMatrix(train2tree2pred,test2$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 160 32  
## Yes 11 21  
##   
## Accuracy : 0.808   
## 95% CI : (0.7503, 0.8575)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.065180   
##   
## Kappa : 0.3845   
## Mcnemar's Test P-Value : 0.002289   
##   
## Sensitivity : 0.39623   
## Specificity : 0.93567   
## Pos Pred Value : 0.65625   
## Neg Pred Value : 0.83333   
## Prevalence : 0.23661   
## Detection Rate : 0.09375   
## Detection Prevalence : 0.14286   
## Balanced Accuracy : 0.66595   
##   
## 'Positive' Class : Yes   
##

train2tree3pred = predict(train2tree2, train2, type = "class")  
head(train2tree3pred)

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

confusionMatrix(train2tree3pred,train2$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 374 75  
## Yes 25 50  
##   
## Accuracy : 0.8092   
## 95% CI : (0.7729, 0.8419)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.005169   
##   
## Kappa : 0.3911   
## Mcnemar's Test P-Value : 9.584e-07   
##   
## Sensitivity : 0.40000   
## Specificity : 0.93734   
## Pos Pred Value : 0.66667   
## Neg Pred Value : 0.83296   
## Prevalence : 0.23855   
## Detection Rate : 0.09542   
## Detection Prevalence : 0.14313   
## Balanced Accuracy : 0.66867   
##   
## 'Positive' Class : Yes   
##

The classification tree model for the “blood” testing set was 80.8% where the naive model’s accuracy was 76.3%. The P-Value was over 0.06 so some of the variables may not have been significant in the model. The classification tree was, however, more accurate making it a better predictor for DonatedMarch. The classification tree model for the “blood” for the training data set was 80.9% accurate compared to 76.1% accuracy on the naive model. The P-value of the training set was 0.005 showing high significance in the variables used in the model. The training set was a solid predictive model for blood DonatedMarch by donors.