## BAN 502

## Mandy Hawkins

## Classification Tree Assignment

Load tidyverse, caret, rpart, rattle, and RColorBrewer.

library(tidyverse)

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

## v ggplot2 3.1.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.7  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

## -- 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)

library(readr)  
parole <- read\_csv("~/BAN502/Module 4/Classification Tree Assignment/Classification Tree Project/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()  
## )

View(parole)

Factoring (renaming) the data Convert the male, race, state, crime, multiple.o???enses, and violator variables to factors.

parole=parole %>% mutate(male =as.factor(as.character(male))) %>% mutate(male = fct\_recode(male, "Female" = "0" , "Male" = "1"))

parole = parole %>% mutate(race = as\_factor(as.character(race))) %>% mutate(race = fct\_recode(race, "NotWhite" = "2", "White" = "1"))

parole = parole %>% mutate(state = as\_factor(as.character(state))) %>% mutate(state = fct\_recode(state, "Kentucky"= "2", "Louisiana"="3", "Virginia"= "4", "Not\_KY\_LA\_or\_VA" ="1"))

parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>% mutate(crime = fct\_recode(crime, "Larceny" ="2", "Drug" = "3", "Driving" ="4", "Other" = "1"))

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

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

Drop empty rows

parole = parole %>% drop\_na() #delete any row with an NA value   
str(parole) #check structure after the drop

## Classes 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "Female","Male": 2 1 2 2 2 2 2 1 1 2 ...  
## $ race : Factor w/ 2 levels "White","NotWhite": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "Not\_KY\_LA\_or\_VA",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : int 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "Driving","Drug",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...

## 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.

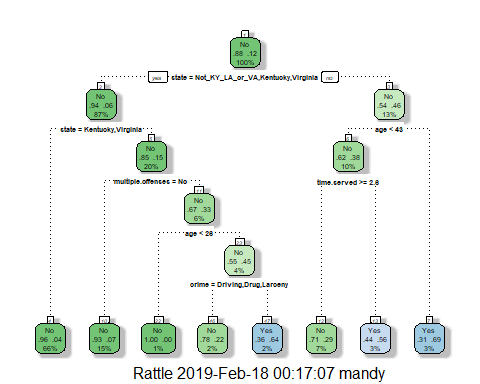
set.seed(12345)  
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE) #70% in training  
train = parole[train.rows,]   
test = parole[-train.rows,]

## Task 2

Create a classi???cation tree to predict “violator” in the training set. Plot the tree

Let’s build a classification tree.

tree1 = rpart(violator ~., train, method="class")  
fancyRpartPlot(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 classi???cation tree to arrive at your answer.

Starting at the top since the parolee is from Louisiana you would move right to ‘no’ then since he is 40 years old, he is under age 43 so you would move ‘yes’ to the left and time served is 5 years which is greater than 2.6 so you would move ‘yes’ to the left which brings us to No- he wouldn’t violate parole.

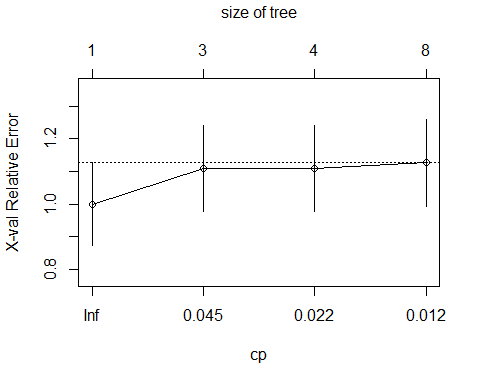
## Task 4

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 di???erent than others we have seen. What cp value should be selected? Probably should choose .036 b/c after that even though the cp decreases it seems to start to overfit.

printcp(tree1)

##   
## 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(tree1)

 ##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)?Prune the tree (at minimum cross-validated error)

tree2 = prune(tree1,cp= tree1$cptable[which.min(tree1$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)

## 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, speci???city, and sensitivty of this tree on the training data.

treepred = predict(tree2, train, type = "class")  
head(treepred)

## [1] No No No No No No  
## Levels: No Yes

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred,train$violator,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 55  
## Yes 0 0  
##   
## Accuracy : 0.8837   
## 95% CI : (0.8513, 0.9112)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.5358   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : 3.305e-13   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8837   
## Prevalence : 0.1163   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

## 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, speci???city, and sensitivty of this tree on the testing data. Comment on the quality of the model. The accuracy is about 88% versus 83% so there is a very slight improvement. The ‘no information rate’ and the ‘P-value’ are both greater than 0.05.

treepred\_test = predict(tree2, newdata=test, type = "class")  
head(treepred\_test)

## [1] No No No No No No  
## Levels: No Yes

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_test,test$violator,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 179 23  
## Yes 0 0  
##   
## Accuracy : 0.8861   
## 95% CI : (0.8341, 0.9264)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.5553   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : 4.49e-06   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8861   
## Prevalence : 0.1139   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

## Task 8

Read in the “Blood.csv” dataset. The dataset contains ???ve 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 ???rst 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”.

Blood <- read\_csv("~/BAN502/Module 4/Classification Tree Assignment/Classification Tree Project/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()  
## )

View(Blood)

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

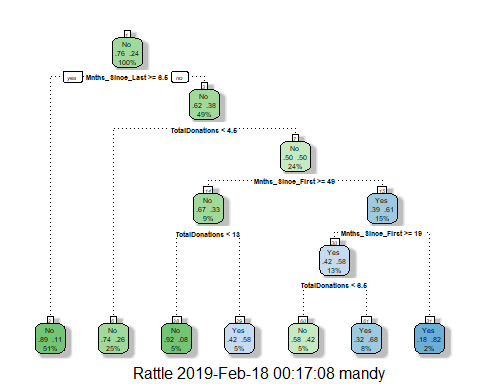
## Task 9

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 classi???cation tree on the training set to predict “DonatedMarch”. Evaluate the complexity parameter (cp) selection for this model.

set.seed(1234)  
train.rows = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train2 = Blood[train.rows,]   
test2 = Blood[-train.rows,]

Create a classi???cation tree to predict “DonatedMarch” in the training set. Plot the tree

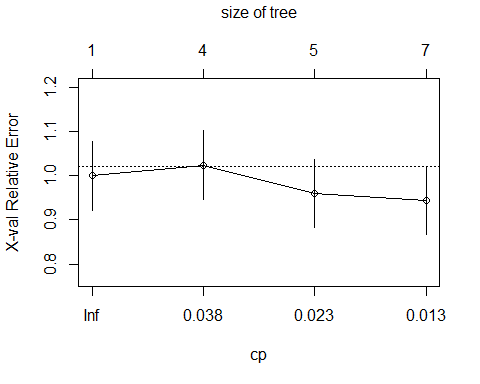
treeone = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(treeone)



printcp(treeone)

##   
## 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

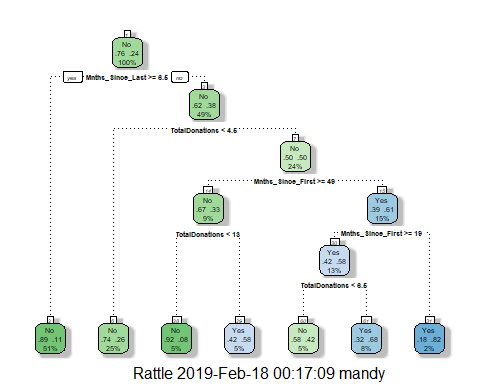
plotcp(treeone)



## 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. The accuracy is about 80% which is higher than the CI of 77%.

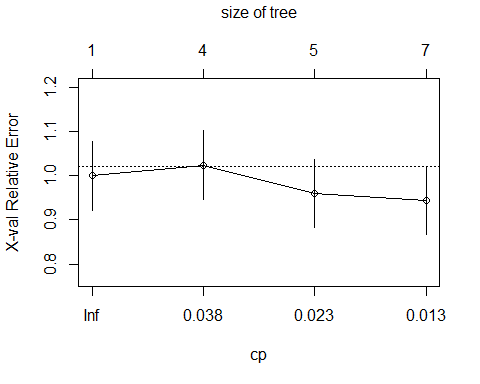
treetwo = prune(treeone,cp= treeone$cptable[which.min(treeone$cptable[,"xerror"]),"CP"])  
#most of the code in the line above can be left untouched.   
fancyRpartPlot(treetwo)



printcp(treetwo)

##   
## 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(treetwo)



treepred2 = predict(treetwo, train2, type = "class")  
head(treepred2)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred2,train2$DonatedMarch,positive="Yes") #predictions first then actual

## 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   
##