Module 4 Assignment 2

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

## Warning: package 'tidyverse' was built under R version 3.6.2

library(caret)

## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(rpart)  
library(rattle)

## Warning: package 'rattle' was built under R version 3.6.2

## Rattle: A free graphical interface for data science with R.  
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

Read in the data and mutate

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()  
## )

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, "Other" = "1", "Kentucky" = "2", "Louisianna" = "3", "Virginia" = "4" ))  
parole = parole %>% mutate(crime = as.factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "Other" = "1", "Larceny" = "2", "Drugs" = "3", "Driving" = "4" ))  
parole = parole %>% mutate(multiple.offenses = as.factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "No" = "0", "Yes" = "1" ))  
parole = parole %>% mutate(violator = as.factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "No" = "0", "Yes" = "1" ))  
  
str(parole)

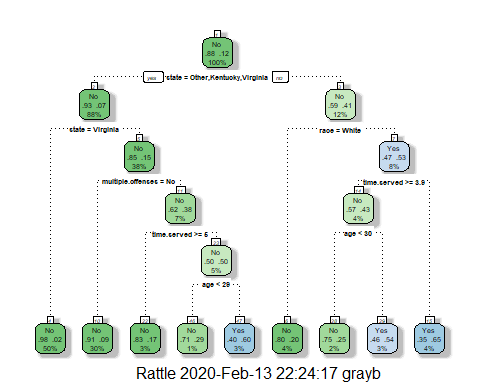
## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "Female","Male": 2 1 2 2 2 2 2 1 1 2 ...  
## $ race : Factor w/ 2 levels "White","Other": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "Other","Kentucky",..: 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 : num 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 "Other","Larceny",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ 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 classiﬁcation tree using all of the predictor variables to predict “violator” in the training set. Plot the 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 classiﬁcation tree to arrive at your answer.

WALK THROUGH: (1) Start at the top. Is the state Other, Kentucky, or Virginia? No. Go to the right. (2) Are they White? We don’t know. We’ll have to go down two paths. White = Yes. Proceed to the left. There is an 80% chance they will NOT be a violator. White = No. Proceed to the right. (3) Have they served more than 3.9 years? Yes. Go to the left. (4) are they less than 30? No. Go to the right. There is a 54 percent chance they will be a violator.

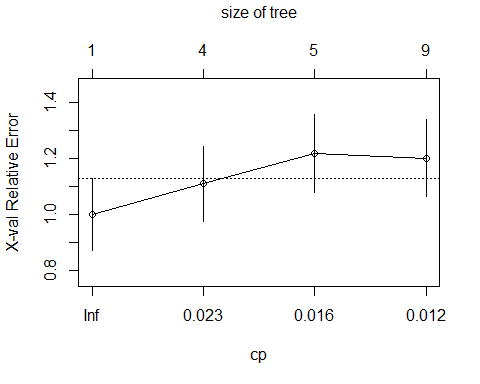
ANSWER: If they are white, there is an 80% chance they will not be a violator. If they are not white, there is a 54% chance of being a violator.

**Task 4** Use the printcp function to evaluate tree performance as a function of the complexity parameter (cp). What cp value should be selected? Note that the printcp table tends to be a more reliable tool than the plot of cp.

printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age multiple.offenses race state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.030303 0 1.00000 1.0000 0.12676  
## 2 0.018182 3 0.90909 1.1091 0.13253  
## 3 0.013636 4 0.89091 1.2182 0.13788  
## 4 0.010000 8 0.83636 1.2000 0.13702

plotcp(tree1)



ANSWER: We want to minimize cross-validation error. From the printcp table we can see that the lowest value for “xerror” is 1.0000, which corresponds to a CP of 0.030303 .

**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. You will ﬁnd that 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)?

tree2 = rpart(violator ~., train, cp=0.030303, method="class")

ANSWER: The non-violators are the majority class.

**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. Note that we would not, in practice, use an unpruned tree as such a tree is very likely to overﬁt on new data.

tree1pred = predict(tree1, train, type = "class")  
head(tree1pred)

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

confusionMatrix(tree1pred,train$violator,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 400 28  
## Yes 18 27  
##   
## Accuracy : 0.9027   
## 95% CI : (0.8724, 0.9279)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.1095   
##   
## Kappa : 0.4862   
##   
## Mcnemar's Test P-Value : 0.1845   
##   
## Sensitivity : 0.49091   
## Specificity : 0.95694   
## Pos Pred Value : 0.60000   
## Neg Pred Value : 0.93458   
## Prevalence : 0.11628   
## Detection Rate : 0.05708   
## Detection Prevalence : 0.09514   
## Balanced Accuracy : 0.72392   
##   
## '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, speciﬁcity, and sensitivty of this tree on the testing data. Comment on the quality of the model.

tree2pred = predict(tree1, test, type = "class")  
head(tree2pred)

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

confusionMatrix(tree2pred,test$violator,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 171 13  
## Yes 8 10  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.3797   
##   
## Kappa : 0.4309   
##   
## Mcnemar's Test P-Value : 0.3827   
##   
## Sensitivity : 0.43478   
## Specificity : 0.95531   
## Pos Pred Value : 0.55556   
## Neg Pred Value : 0.92935   
## Prevalence : 0.11386   
## Detection Rate : 0.04950   
## Detection Prevalence : 0.08911   
## Balanced Accuracy : 0.69504   
##   
## 'Positive' Class : Yes   
##

ANSWER: Accuracy: training - 0.9027; testing - 0.896 . Pretty close No information rate: training - 0.8837; testing - 0.8861. Also pretty close. Conclusion: We have not overfit the data. Yay!

**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”.

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

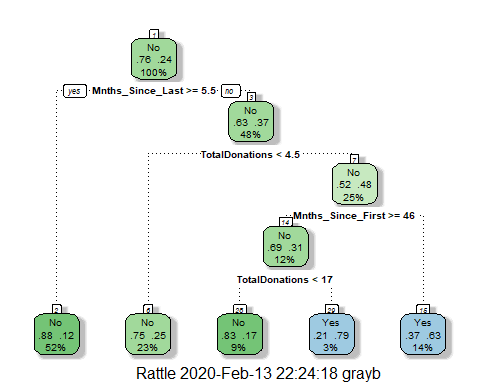
## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

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

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 2 1 2 2 ...

**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” so 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.

set.seed(1234)  
btrain.rows = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
btrain = blood[btrain.rows,]   
btest = blood[-btrain.rows,]  
btree = rpart(DonatedMarch ~., btrain, method="class")  
fancyRpartPlot(btree)

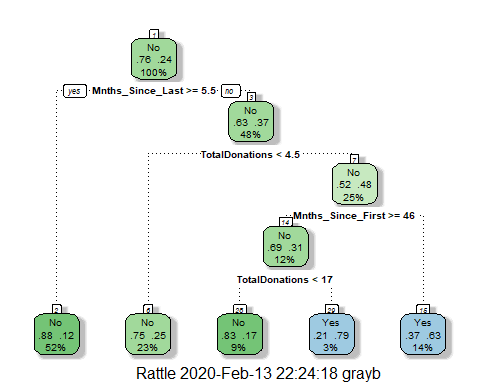


ANSWER: We want to minimize cross-validation error. From the printcp table we can see that the lowest value for “xerror” is 0.848, which corresponds to a CP of 0.010000 .

**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**

btree2 = rpart(DonatedMarch ~., btrain, cp=0.010000, method="class")  
fancyRpartPlot(btree2)



btree2pred = predict(btree2, btrain, type = "class")  
head(tree2pred)

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

confusionMatrix(btree2pred, btrain$DonatedMarch, positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 370 69  
## Yes 29 56  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4216   
##   
## Mcnemar's Test P-Value : 8.162e-05   
##   
## Sensitivity : 0.4480   
## Specificity : 0.9273   
## Pos Pred Value : 0.6588   
## Neg Pred Value : 0.8428   
## Prevalence : 0.2385   
## Detection Rate : 0.1069   
## Detection Prevalence : 0.1622   
## Balanced Accuracy : 0.6877   
##   
## 'Positive' Class : Yes   
##

btree2testpred = predict(btree2, btest, type = "class")  
head(btree2testpred)

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

confusionMatrix(btree2testpred, btest$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 156 35  
## Yes 15 18  
##   
## Accuracy : 0.7768   
## 95% CI : (0.7165, 0.8296)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.35155   
##   
## Kappa : 0.2896   
##   
## Mcnemar's Test P-Value : 0.00721   
##   
## Sensitivity : 0.33962   
## Specificity : 0.91228   
## Pos Pred Value : 0.54545   
## Neg Pred Value : 0.81675   
## Prevalence : 0.23661   
## Detection Rate : 0.08036   
## Detection Prevalence : 0.14732   
## Balanced Accuracy : 0.62595   
##   
## 'Positive' Class : Yes   
##

ANSWER: Accuracy: training - 0.813; testing - 0.7768 . Diff of over 4%. Suspicious. I tried it with other random seeds and got closer numers, so I think this one is just by chance on the low side.  
No information rate: training - 0.7615; testing - 0.7634. Pretty close. Our accuracy rate is higher than the “no information rate”, so we know our model is doing something. Conclusion: We have not overfit the data. Yay!