## Module 4 – Classification Trees Assignment

# Lea Shipley

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

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()  
## )

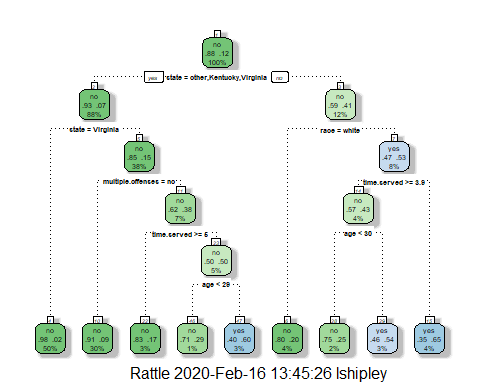
parole = parole %>% mutate(male = as\_factor(as.numeric(male))) %>%  
mutate(male = fct\_recode(male,"male" = "1","female" = "0")) %>%  
 mutate(race = as\_factor(as.numeric(race))) %>%  
 mutate(race = fct\_recode(race, "white" = "1", "other" = "2")) %>%  
 mutate(state = as\_factor(as.numeric(state))) %>%  
 mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "other" = "1")) %>%  
 mutate(crime = as\_factor(as.numeric(crime))) %>%  
 mutate(crime = fct\_recode(crime, "larceny" = "2", "drugs" = "3", "driving" = "4", "other" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(as.numeric(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "yes" = "1", "no" = "0")) %>%  
 mutate(violator = as.factor(as.numeric(violator))) %>%   
 mutate(violator = fct\_recode(violator, "yes" = "1", "no" = "0"))

**Task 1**

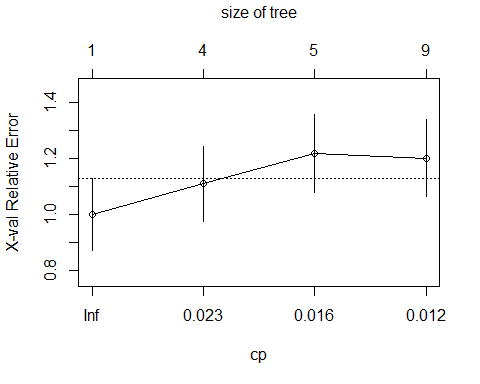
set.seed(12345)   
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE) #70% in training  
train = parole[train.rows,]   
test = parole[-train.rows,]

**Task 2**

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



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?

If the parolee is white, they would be classified as a non-violator. If the parolee is not white, the parolee would be classified as a violator when taking into consideration their age and time served. These answers are arrived upon by beginning at the top of the tree and answering “yes” or “no” at each node, and going to the left for “yes” and right if the answer is “no.”

**Task 4**

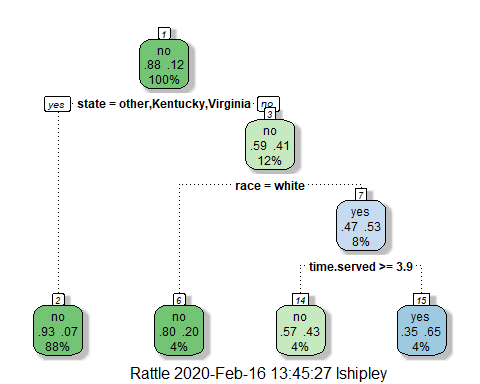
printcp(tree1)

##   
## 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

The cp value 0.030303 should be selected to yield the best cross-validated error.

**Task 5**

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



The majority class in the training set is predicted to not violate parole.

**Task 6**

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

## 1 2 3 4 5 6   
## no no no no no no   
## Levels: no yes

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

## 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**

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

## 1 2 3 4 5 6   
## no no no no no no   
## Levels: no yes

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

## 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   
##

The level of accuracy is very close for both the training set (0.9027) and the testing set (0.896), which would imply that the data was likely not overfit. There is only a marginal gain in accuracy over the No Information Rate.

**Task 8**

blood <- read\_csv("Blood.csv")

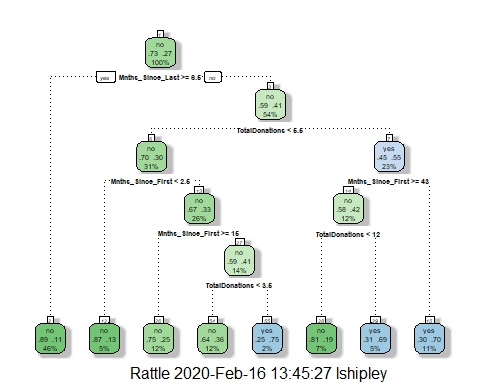
## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

blood = blood %>% mutate(DonatedMarch = as\_factor(as.numeric(DonatedMarch))) %>%  
mutate(DonatedMarch = fct\_recode(DonatedMarch,"yes" = "1","no" = "0"))

**Task 9**

set.seed(1234)   
train.rows2 = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train2 = blood[train.rows,]   
test2 = blood[-train.rows,]

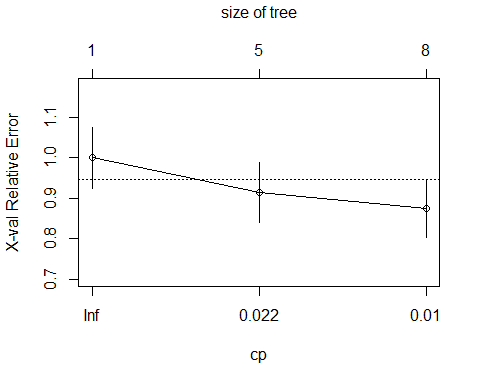
tree3 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree3)



printcp(tree3)

##   
## 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: 128/473 = 0.27061  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.046875 0 1.00000 1.00000 0.075487  
## 2 0.010417 4 0.75781 0.91406 0.073312  
## 3 0.010000 7 0.72656 0.87500 0.072231

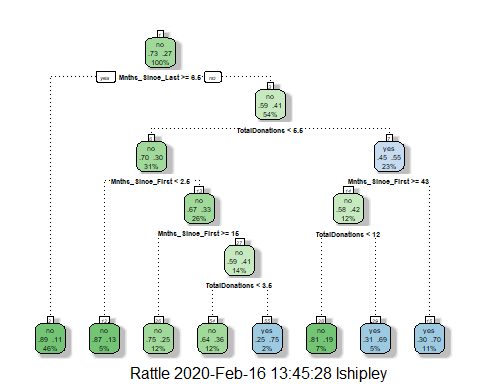
plotcp(tree3)



The best cp value here would be 0.010000 to yield the lowest cross-validated error.

**Task 10**

tree4 = prune(tree3,cp=tree3$cptable[which.min(tree3$cptable[,"xerror"]),"CP"])  
  
fancyRpartPlot(tree4)



treepred2 = predict(tree4, train2, type = "class")  
head(treepred2)

## 1 2 3 4 5 6   
## yes yes yes yes yes yes   
## Levels: no yes

confusionMatrix(treepred2,train2$DonatedMarch,positive="yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 319 67  
## yes 26 61  
##   
## Accuracy : 0.8034   
## 95% CI : (0.7647, 0.8383)  
## No Information Rate : 0.7294   
## P-Value [Acc > NIR] : 0.0001165   
##   
## Kappa : 0.4461   
##   
## Mcnemar's Test P-Value : 3.357e-05   
##   
## Sensitivity : 0.4766   
## Specificity : 0.9246   
## Pos Pred Value : 0.7011   
## Neg Pred Value : 0.8264   
## Prevalence : 0.2706   
## Detection Rate : 0.1290   
## Detection Prevalence : 0.1839   
## Balanced Accuracy : 0.7006   
##   
## 'Positive' Class : yes   
##

treepred\_test2 = predict(tree4, newdata=test2, type = "class")  
head(treepred\_test2)

## 1 2 3 4 5 6   
## yes yes yes yes yes yes   
## Levels: no yes

confusionMatrix(treepred\_test2,test2$DonatedMarch,positive="yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no yes  
## no 202 34  
## yes 23 16  
##   
## Accuracy : 0.7927   
## 95% CI : (0.74, 0.8391)  
## No Information Rate : 0.8182   
## P-Value [Acc > NIR] : 0.8784   
##   
## Kappa : 0.2382   
##   
## Mcnemar's Test P-Value : 0.1853   
##   
## Sensitivity : 0.32000   
## Specificity : 0.89778   
## Pos Pred Value : 0.41026   
## Neg Pred Value : 0.85593   
## Prevalence : 0.18182   
## Detection Rate : 0.05818   
## Detection Prevalence : 0.14182   
## Balanced Accuracy : 0.60889   
##   
## 'Positive' Class : yes

Accuracy on the training set is 0.8034, while accuracy on the testing set is 0.7927, so the data is likely not over fit. However, the No Information Rate is higher than the model’s accuracy on the testing set.