library("tidyverse")

## ── Attaching packages ─────────────────────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.2.1 ✓ purrr 0.3.3  
## ✓ tibble 2.1.3 ✓ dplyr 0.8.4  
## ✓ tidyr 1.0.2 ✓ stringr 1.4.0  
## ✓ readr 1.3.1 ✓ 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 5.3.0 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.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, "white" = "1", "nonwhite" = "2"))

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

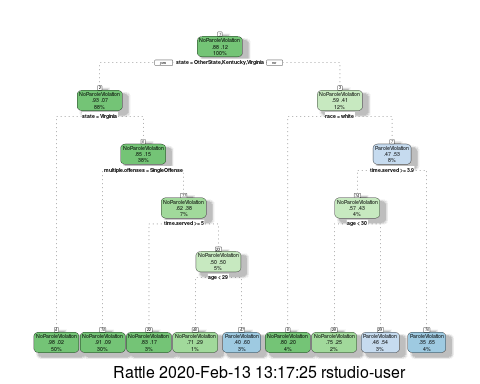
parole = parole %>% mutate(crime = as.factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime, "OtherCrime" = "1", "larceny" = "2", "drugs" = "3", "driving" = "4"))

parole = parole %>% mutate(multiple.offenses = as.factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "SingleOffense" = "0", "MultiOffenses" = "1"))

parole = parole %>% mutate(violator = as.factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator, "NoParoleViolation" = "0", "ParoleViolation" = "1"))

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

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

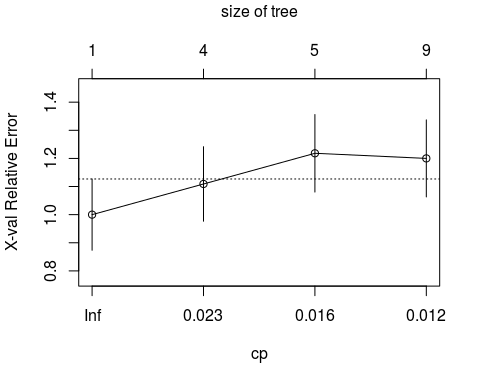


Task 3 - So you go from the top of the tree and so the person is not from any of the states listed so we go down the right side, next is race which we weren’t given so i moved down the no column to the right, next to time served greater than 3.9 years which is a yes so i move to the left next to age less than 30 which is a no so i go down the left to my answer on the bottom which says they will violate parole.

printcp(tree1)

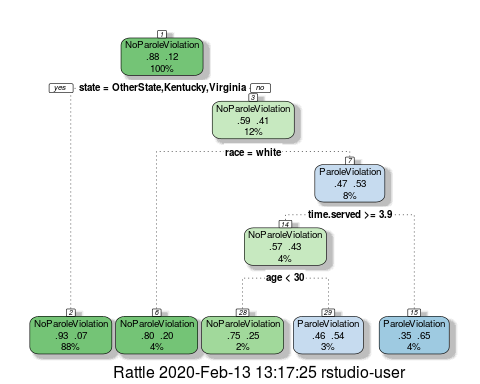
##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age multiple.offenses race state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit 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)



Task 4 - .018 based on looking at the graph above and the r console piece.

tree2 = prune(tree1, cp = 0.018)  
fancyRpartPlot(tree2)



Task 5 - state has the most observations.

library("e1071")  
treepred = predict(tree1, train, type = "class")  
confusionMatrix(treepred,train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 400 28  
## ParoleViolation 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 : 0.9569   
## Specificity : 0.4909   
## Pos Pred Value : 0.9346   
## Neg Pred Value : 0.6000   
## Prevalence : 0.8837   
## Detection Rate : 0.8457   
## Detection Prevalence : 0.9049   
## Balanced Accuracy : 0.7239   
##   
## 'Positive' Class : NoParoleViolation  
##

treepred2 = predict(tree1, test, type = "class")  
confusionMatrix(treepred2, test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoParoleViolation ParoleViolation  
## NoParoleViolation 171 13  
## ParoleViolation 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 : 0.9553   
## Specificity : 0.4348   
## Pos Pred Value : 0.9293   
## Neg Pred Value : 0.5556   
## Prevalence : 0.8861   
## Detection Rate : 0.8465   
## Detection Prevalence : 0.9109   
## Balanced Accuracy : 0.6950   
##   
## 'Positive' Class : NoParoleViolation  
##

Task 7 = The test data has a lower accuracy rating as the train data had a higher accuracy rating and so the train data seems to be the better model.

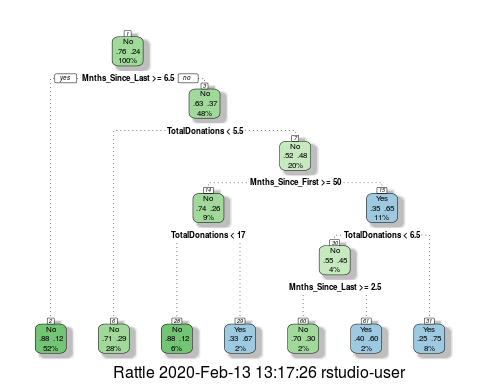
blood = read\_csv("Blood.csv")

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

set.seed(12345)  
train.rows = createDataPartition(y=blood$DonatedMarch, p=0.7, list=FALSE)  
train2 = blood[train.rows,]  
test2 = blood[-train.rows,]

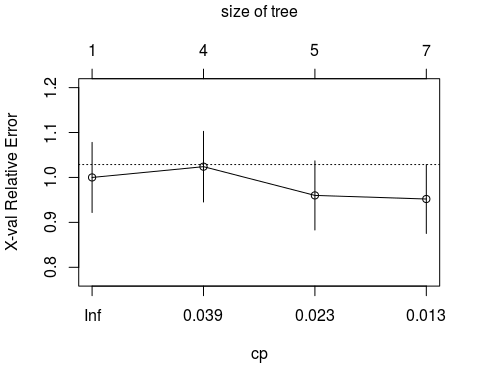
treeblood = rpart(DonatedMarch ~., train2, method = "class")  
fancyRpartPlot(treeblood)



printcp(treeblood)

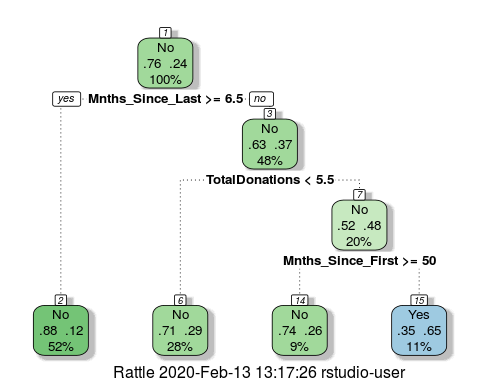
##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.048 0 1.000 1.000 0.078049  
## 2 0.032 3 0.856 1.024 0.078682  
## 3 0.016 4 0.824 0.960 0.076949  
## 4 0.010 6 0.792 0.952 0.076723

plotcp(treeblood)



Task 9 = i would say 0.032 would be the best CP value juding from the above plt and the r console data.

treeblood2 = prune(treeblood, cp = 0.032)  
fancyRpartPlot(treeblood2)



treepred = predict(treeblood2, train2, type = "class")  
confusionMatrix(treepred,train2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 378 86  
## Yes 21 39  
##   
## Accuracy : 0.7958   
## 95% CI : (0.7587, 0.8295)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.03474   
##   
## Kappa : 0.3157   
##   
## Mcnemar's Test P-Value : 6.128e-10   
##   
## Sensitivity : 0.9474   
## Specificity : 0.3120   
## Pos Pred Value : 0.8147   
## Neg Pred Value : 0.6500   
## Prevalence : 0.7615   
## Detection Rate : 0.7214   
## Detection Prevalence : 0.8855   
## Balanced Accuracy : 0.6297   
##   
## 'Positive' Class : No   
##

treepred2 = predict(treeblood2, test2, type = "class")  
confusionMatrix(treepred2,test2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 157 35  
## Yes 14 18  
##   
## Accuracy : 0.7812   
## 95% CI : (0.7213, 0.8336)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.294458   
##   
## Kappa : 0.2986   
##   
## Mcnemar's Test P-Value : 0.004275   
##   
## Sensitivity : 0.9181   
## Specificity : 0.3396   
## Pos Pred Value : 0.8177   
## Neg Pred Value : 0.5625   
## Prevalence : 0.7634   
## Detection Rate : 0.7009   
## Detection Prevalence : 0.8571   
## Balanced Accuracy : 0.6289   
##   
## 'Positive' Class : No   
##

Task 10 - Both of these predictions on both train and test have low accuracy ratings. The train population has a better accuracy rating than the testing population.