## Classification Trees - MOd 4 Assignment 2

Load Libraries

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

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(rpart)  
library(rattle)

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

Reading Data

parole <- read.csv("parole.csv")  
parole = parole %>% drop\_na() #delete any row with an NA value

Structure and summary

str(parole)

## 'data.frame': 675 obs. of 9 variables:  
## $ male : int 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : int 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 : int 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: int 0 0 0 0 0 0 0 0 0 0 ...  
## $ crime : int 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : int 0 0 0 0 0 0 0 0 0 0 ...

summary(parole)

## male race age state   
## Min. :0.0000 Min. :1.000 Min. :18.40 Min. :1.000   
## 1st Qu.:1.0000 1st Qu.:1.000 1st Qu.:25.35 1st Qu.:2.000   
## Median :1.0000 Median :1.000 Median :33.70 Median :3.000   
## Mean :0.8074 Mean :1.424 Mean :34.51 Mean :2.887   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.:42.55 3rd Qu.:4.000   
## Max. :1.0000 Max. :2.000 Max. :67.00 Max. :4.000   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 Min. :0.0000 Min. :1.000   
## 1st Qu.:3.250 1st Qu.:12.00 1st Qu.:0.0000 1st Qu.:1.000   
## Median :4.400 Median :12.00 Median :1.0000 Median :2.000   
## Mean :4.198 Mean :13.06 Mean :0.5363 Mean :2.059   
## 3rd Qu.:5.200 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:3.000   
## Max. :6.000 Max. :18.00 Max. :1.0000 Max. :4.000   
## violator   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.1156   
## 3rd Qu.:0.0000   
## Max. :1.0000

parole = parole %>% mutate(male = as\_factor(male)) %>%  
mutate(male = fct\_recode(male,  
"male" = "1",  
"female" = "0"))  
  
parole = parole %>% mutate(race = as\_factor(race)) %>%  
mutate(race = fct\_recode(race,  
"white" = "1",  
"otherwise" = "2"))  
  
parole = parole %>% mutate(state = as\_factor(state)) %>%  
mutate(state = fct\_recode(state,  
"anyotherstate" = "1",  
"Kentucky" = "2",  
"Louisian" = "3",  
"Virginia" = "4"))  
  
parole = parole %>% filter(time.served < 7)  
parole = parole %>% filter(max.sentence < 19)  
  
parole = parole %>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"multiple" = "1",  
"otherwise" = "0"))  
  
parole = parole %>% mutate(crime = as\_factor(crime)) %>%  
mutate(crime = fct\_recode(crime,  
"other" = "1",  
"larceny" = "2",  
"drug\_related" = "3",  
"driving\_realted" = "4"))  
  
parole = parole %>% mutate(violator = as\_factor(violator)) %>%  
mutate(violator = fct\_recode(violator,  
"violated" = "1",  
"completed" = "0"))

Structure and summary

str(parole)

## '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","otherwise": 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 "anyotherstate",..: 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 "otherwise","multiple": 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 "completed","violated": 1 1 1 1 1 1 1 1 1 1 ...

summary(parole)

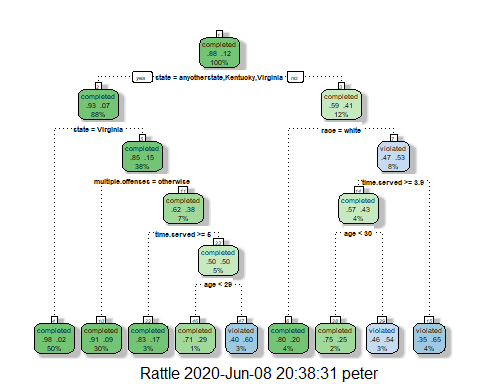
## male race age state   
## female:130 white :389 Min. :18.40 anyotherstate:143   
## male :545 otherwise:286 1st Qu.:25.35 Kentucky :120   
## Median :33.70 Louisian : 82   
## Mean :34.51 Virginia :330   
## 3rd Qu.:42.55   
## Max. :67.00   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 otherwise:313 other :315   
## 1st Qu.:3.250 1st Qu.:12.00 multiple :362 larceny :106   
## Median :4.400 Median :12.00 drug\_related :153   
## Mean :4.198 Mean :13.06 driving\_realted:101   
## 3rd Qu.:5.200 3rd Qu.:15.00   
## Max. :6.000 Max. :18.00   
## violator   
## completed:597   
## violated : 78   
##   
##   
##   
##

Split the data (training and testing)

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

Let’s build a classification tree.

tree1P = rpart(violator ~., trainP, method="class")  
fancyRpartPlot(tree1P)

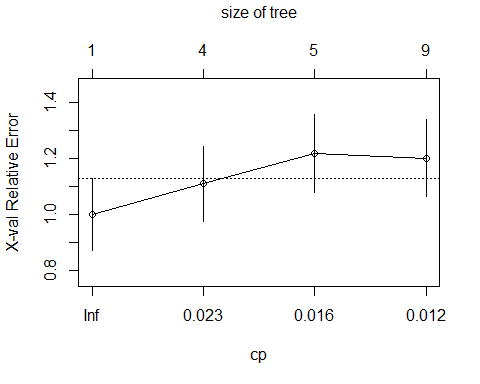


Classifying a 40 yr old from louisiana who served a 5 year prison sentence.  
Starting at the Top; state is louisiana, so we go right. We do not know the race, hence if the parolee is white he/she completed parole, else time served is greater than 3.5, move left and finally age is not less than 30, so move right indicating the parolee violated the parole.

printcp(tree1P)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = trainP, 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(tree1P)

 **Based on the table above cp should be .0303.**

Prune the tree (at minimum cross-validated error)

tree2P = prune(tree1P,cp= tree1P$cptable[which.min(tree1P$cptable[,"xerror"]),"CP"])

**Parole completed is the majority class.**

Predictions on training set

treepredP = predict(tree1P, trainP, type = "class")  
head(treepredP)

## 1 2 3 4 5 6   
## completed completed completed completed completed completed   
## Levels: completed violated

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepredP,trainP$violator,positive="completed") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed violated  
## completed 400 28  
## violated 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 : completed   
##

Predictions on testing set

treepred\_testP = predict(tree1P, newdata=testP, type = "class")  
head(treepred\_testP)

## 1 2 3 4 5 6   
## completed completed completed completed completed completed   
## Levels: completed violated

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_testP,testP$violator,positive="completed") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction completed violated  
## completed 171 13  
## violated 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 : completed   
##

*The model has 90% accuracy, slightly higher than the no information rate.*

Reading Data

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

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

Structure and summary

str(Blood)

## 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : int 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : int 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : int 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: int 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 ...

summary(Blood)

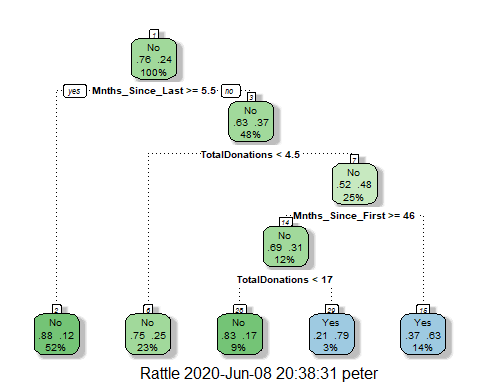
## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :34.28   
## 3rd Qu.:14.000 3rd Qu.: 7.000 3rd Qu.: 1750 3rd Qu.:50.00   
## Max. :74.000 Max. :50.000 Max. :12500 Max. :98.00   
## DonatedMarch  
## No :570   
## Yes:178   
##   
##   
##   
##

Split the data (training and testing)

set.seed(1234)  
train.rows = createDataPartition(y = Blood$DonatedMarch , p=0.7, list = FALSE) #70% in training  
trainB = slice(Blood,train.rows)   
testB = slice(Blood,-train.rows)

Let’s build a classification tree.

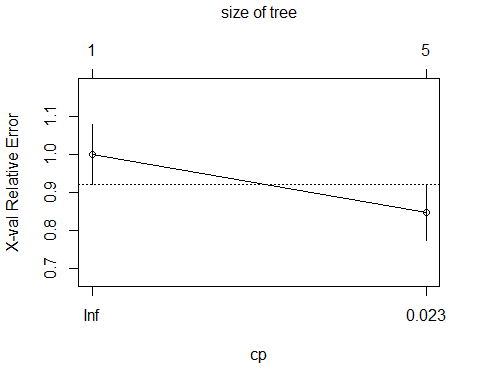
treeB = rpart(DonatedMarch ~., trainB, method="class")  
fancyRpartPlot(treeB)



printcp(treeB)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = trainB, 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 0.050667 0 1.000 1.000 0.078049  
## 2 0.010000 4 0.784 0.848 0.073564

plotcp(treeB)



*cp of .01 gives the minimum cross-validated error.*

Prune the tree (at minimum cross-validated error)

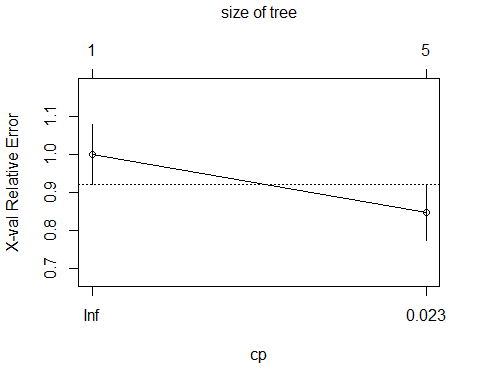
tree2B = prune(treeB,cp= treeB$cptable[which.min(treeB$cptable[,"xerror"]),"CP"])

*Pruning might not have been necessary.*

printcp(tree2B)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = trainB, 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 0.050667 0 1.000 1.000 0.078049  
## 2 0.010000 4 0.784 0.848 0.073564

plotcp(tree2B)



Predictions on training set

treepredB = predict(tree2B, trainB, type = "class")  
head(treepredB)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepredB,trainB$DonatedMarch,positive="Yes") #predictions first then actual

## 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   
##

Predictions on testing set

treepred\_testB = predict(tree2B, newdata=testB, type = "class")  
head(treepred\_testB)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_testB,testB$DonatedMarch,positive="Yes") #predictions first then actual

## 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   
##

The accuracy of training and testing set is low at 81% and 78% respectively. I expected more “No” in the predictions, training did not predict any “No”. More analysis on the data is needed, totaldonations and totaldonated may be highly correlated.