## Classification Trees

### Anthony Sumter

**Libraries**

#install.packages("tidyverse","caret", "rpart", "rattle", "RColorBrewer" )  
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)

**Dataset**

parole = read.csv("parole.csv")  
parole = parole %>% mutate(male = as\_factor(as.character(male)), race = as\_factor (as.character (race)), state = as\_factor(as.character (state)), crime = as\_factor(as.character (crime)), multiple.offenses = as\_factor(as.character(multiple.offenses)), violator = as\_factor(as.character(violator))) %>%  
mutate(male = fct\_recode(male,"female" = "0","male" = "1")) %>% mutate(race = fct\_recode(race,"White" = "1","Otherwise" = "2"))%>% mutate(state = fct\_recode(state,"OtherState" = "1","Kentucky" = "2", "Louisiana"= "3", "Virgina" = "4"))%>% mutate(crime = fct\_recode(crime,"OtherCrime" = "1", "Larceny" = "2", "DrugRelated" = "3", "DrivingRelated" = "4")) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses,"Otherwise" = "0", "MultipleOffenses" = "1")) %>% mutate(violator = fct\_recode(violator,"Violated" = "1", "NonViolated" = "0"))  
str(parole)

## 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "male","female": 1 2 1 1 1 1 1 2 2 1 ...  
## $ 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 "OtherState","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 : int 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "Otherwise","MultipleOffenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "DrivingRelated",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "NonViolated",..: 1 1 1 1 1 1 1 1 1 1 ...

summary(parole)

## male race age state   
## male :545 White :389 Min. :18.40 OtherState:143   
## female:130 Otherwise:286 1st Qu.:25.35 Kentucky :120   
## Median :33.70 Louisiana : 82   
## Mean :34.51 Virgina :330   
## 3rd Qu.:42.55   
## Max. :67.00   
## time.served max.sentence multiple.offenses  
## Min. :0.000 Min. : 1.00 Otherwise :313   
## 1st Qu.:3.250 1st Qu.:12.00 MultipleOffenses:362   
## Median :4.400 Median :12.00   
## Mean :4.198 Mean :13.06   
## 3rd Qu.:5.200 3rd Qu.:15.00   
## Max. :6.000 Max. :18.00   
## crime violator   
## DrivingRelated:101 NonViolated:597   
## DrugRelated :153 Violated : 78   
## OtherCrime :315   
## Larceny :106   
##   
##

#### 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.

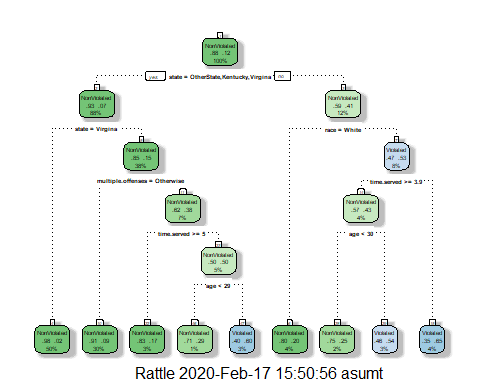
parole = parole %>% drop\_na()  
str(parole)

## 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "male","female": 1 2 1 1 1 1 1 2 2 1 ...  
## $ 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 "OtherState","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 : int 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "Otherwise","MultipleOffenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "DrivingRelated",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "NonViolated",..: 1 1 1 1 1 1 1 1 1 1 ...

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

Classifying a 40 year-old parolee using the classification tree goes as followed. Since they are from the state of Louisiana you would move on to the Right handed branch. If the parolee was white then the age wouldn’t matter as they wouldn’t have violated parole. If the parolee wasn’t white the you would move on to the next branch evaluating time served and since in this case the parolee served greater than 3.9 years you would move on to the next branch looking at age. With this being the lasst decision branch for this scenario this parolee being older than 30 years would have violated his parolee based on the classification tree.

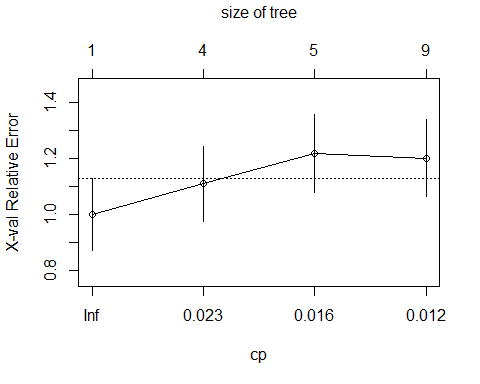
#### Task 4: Use the printcp function to evaluate tree performance as a function of the complexity parameter (cp). What cp value should be selected? Note that the printcp table tends to be a more reliable tool than the plot of cp.

Based on the printcp function the cp value of .023 should be selected to evaluate tree performance.

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

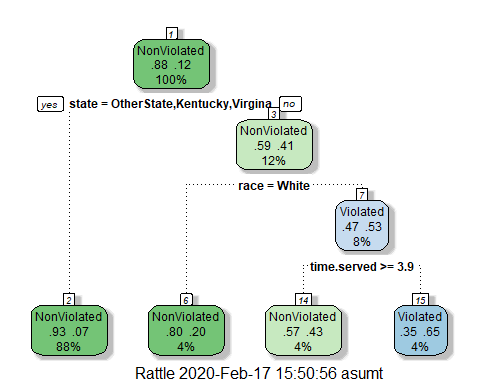
plotcp(tree1)



#### Task 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)?

It appears bassed on the tree the majority class would be the state variable.

tree2 = prune(tree1,cp= tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
tree2 = rpart(violator ~., train, cp=0.023, method="class")  
fancyRpartPlot(tree2)



#### 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.

Accuracy for this tree on the training data is 90% , Sensitivity is 49%, and Specificity is 96%

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

## 1 3 6 7 8 10   
## NonViolated NonViolated NonViolated NonViolated NonViolated NonViolated   
## Levels: NonViolated Violated

confusionMatrix(treepred, train$violator ,positive="Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolated Violated  
## NonViolated 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 : 0.49091   
## 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 : Violated   
##

#### 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.

The predictions would be an marginal gain of 1% compared to the naive rate of 88.6% increasing the accuracy of the predictions to 89.6%. This could indicate that the data is not overfitted.

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

## 2 4 5 9 11 19   
## NonViolated NonViolated NonViolated NonViolated NonViolated NonViolated   
## Levels: NonViolated Violated

confusionMatrix(treepred\_test, test$violator ,positive="Violated")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolated Violated  
## NonViolated 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 : 0.43478   
## 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 : Violated   
##

#### 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")  
blood = blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%  
mutate(DonatedMarch = fct\_recode(DonatedMarch,"No" = "0","Yes" = "1"))  
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 "Yes","No": 1 1 1 1 2 2 1 2 1 1 ...

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  
## Yes:178   
## No :570   
##   
##   
##   
##

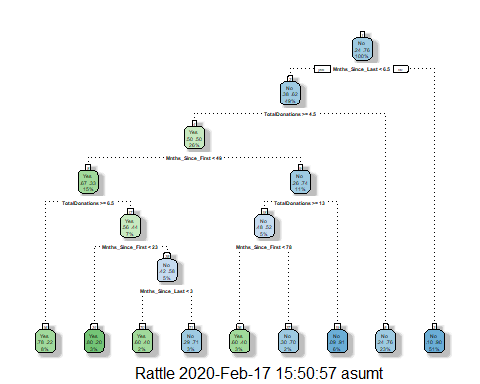
#### Task 9: Split the dataset into training (70%) and testing (30%) sets. You may wish to name your training and testing sets “train2” and “test2” so as to not confuse them with the parole datsets Use set.seed of 1234. Then develop a classification tree on the training set to predict “DonatedMarch”. Evaluate the complexity parameter (cp) selection for this model.

It appears that the complextity parameter selection for this model is at .01 for the best classification tree.

blood = blood %>% drop\_na()  
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 "Yes","No": 1 1 1 1 2 2 1 2 1 1 ...

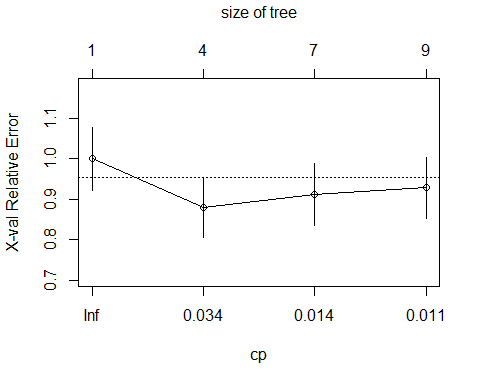
set.seed(1234)  
train.rows1 = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE)  
train2 = blood[train.rows1,]   
test2 = blood[-train.rows1,]  
tree3 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree3)



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: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.072 0 1.000 1.000 0.078049  
## 2 0.016 3 0.784 0.880 0.074580  
## 3 0.012 6 0.736 0.912 0.075556  
## 4 0.010 8 0.712 0.928 0.076030

plotcp(tree3)



#### 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 predictions would be an significant gain of 5.1% compared to the naive rate of 76.2% increasing the accuracy of the predictions to 81.3%. This could indicate that the data is overfitted.

tree4 = prune(tree3,cp= tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
treepred2 = predict(tree4, train2, type = "class")  
head(treepred2)

## 2 3 4 5 6 9   
## Yes Yes Yes No No Yes   
## Levels: Yes No

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 53 26  
## No 72 373  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4107   
##   
## Mcnemar's Test P-Value : 5.476e-06   
##   
## Sensitivity : 0.4240   
## Specificity : 0.9348   
## Pos Pred Value : 0.6709   
## Neg Pred Value : 0.8382   
## Prevalence : 0.2385   
## Detection Rate : 0.1011   
## Detection Prevalence : 0.1508   
## Balanced Accuracy : 0.6794   
##   
## 'Positive' Class : Yes   
##

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

## 1 7 8 10 12 14   
## No Yes Yes No No Yes   
## Levels: Yes No

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 18 20  
## No 35 151  
##   
## Accuracy : 0.7545   
## 95% CI : (0.6927, 0.8094)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.65710   
##   
## Kappa : 0.2468   
##   
## Mcnemar's Test P-Value : 0.05906   
##   
## Sensitivity : 0.33962   
## Specificity : 0.88304   
## Pos Pred Value : 0.47368   
## Neg Pred Value : 0.81183   
## Prevalence : 0.23661   
## Detection Rate : 0.08036   
## Detection Prevalence : 0.16964   
## Balanced Accuracy : 0.61133   
##   
## 'Positive' Class : Yes   
##