## Module 4 Assignment 1: Classification Trees

library(tidyverse)

## -- Attaching packages ------------------------------------------------ tidyverse 1.2.1 --

## v ggplot2 3.1.0 v purrr 0.3.2   
## v tibble 2.1.1 v dplyr 0.8.0.1  
## v tidyr 0.8.3 v stringr 1.4.0   
## v readr 1.3.1 v forcats 0.4.0

## -- Conflicts --------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(RColorBrewer)  
library(rattle)

## Rattle: A free graphical interface for data science with R.  
## Version 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(rpart)  
library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(readr)  
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()  
## )

View(parole)

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",  
"Other" = "2"))

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

parole = parole%>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"Other" = "1",  
"Larceny" = "2",  
"Drug-related" = "3",  
"Driving-related" = "4" ))

parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"No" = "0",  
"Yes" = "1"))

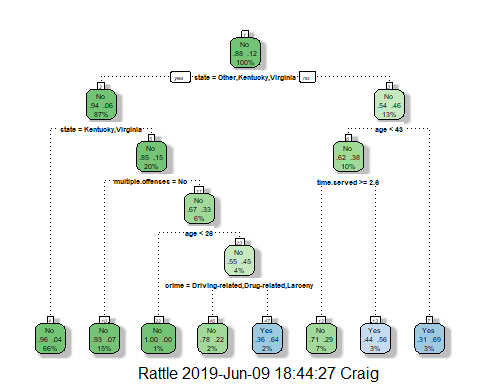
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"No" = "0",  
"Yes" = "1"))

#### Task 1: Split the data into training and testing sets

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 to predict “violator” in the training set and 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?

I would clasify the parolee as a “No” as a likey parole violator.

Describe how you “walk through” the classification tree to arrive at your answer.

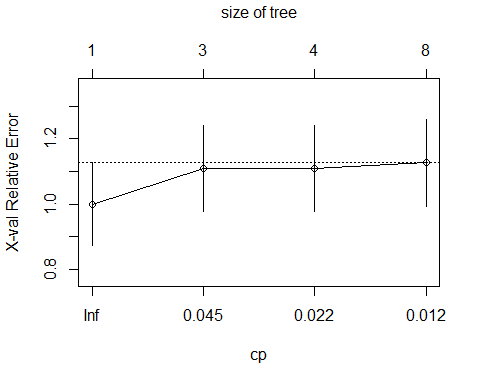
I started at the top of the tree and chose “No” for state = Other, Kentucky, Virginia. The Working down the right side of the tree, I chose “Yes” for age < 43, then on the left of that branch chose “yes” for time served >= 2.6 which ended with the “No” leaf.

#### Task 4: Use the plotcp and printcp functions to evaluate tree performance as a function of the complexity parameter (cp)

printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age crime multiple.offenses state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.054545 0 1.00000 1.0000 0.12676  
## 2 0.036364 2 0.89091 1.1091 0.13253  
## 3 0.013636 3 0.85455 1.1091 0.13253  
## 4 0.010000 7 0.80000 1.1273 0.13345

plotcp(tree1)



What cp value should be selected? .04 should be selected.

#### Task 5: Prune the tree from Task 2 back to the cp value that you selected in Task 4

tree2 = prune(tree1,cp= tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])

Which class (category) in the training set is the majority class (i.e., has the most observations)? Violator = No is the majority class.

#### Task 6: Use the unpruned tree from Task 2 to develop predictions for the training data

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")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 402 28  
## Yes 16 27  
##   
## Accuracy : 0.907   
## 95% CI : (0.8771, 0.9316)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.06272   
##   
## Kappa : 0.5   
##   
## Mcnemar's Test P-Value : 0.09725   
##   
## Sensitivity : 0.49091   
## Specificity : 0.96172   
## Pos Pred Value : 0.62791   
## Neg Pred Value : 0.93488   
## Prevalence : 0.11628   
## Detection Rate : 0.05708   
## Detection Prevalence : 0.09091   
## Balanced Accuracy : 0.72632   
##   
## 'Positive' Class : Yes   
##

Accuracy is .91, Sensitivity is .49 and Specicifity is .96.

#### Task 7: Use the unpruned tree from Task 2 to develop predictions for the testing data

treepred2 = predict(tree1, newdata = test, type = "class")  
head(treepred2)

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

confusionMatrix(treepred2,test$violator,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 170 19  
## Yes 9 4  
##   
## Accuracy : 0.8614   
## 95% CI : (0.8059, 0.9059)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.88631   
##   
## Kappa : 0.1525   
##   
## Mcnemar's Test P-Value : 0.08897   
##   
## Sensitivity : 0.17391   
## Specificity : 0.94972   
## Pos Pred Value : 0.30769   
## Neg Pred Value : 0.89947   
## Prevalence : 0.11386   
## Detection Rate : 0.01980   
## Detection Prevalence : 0.06436   
## Balanced Accuracy : 0.56182   
##   
## 'Positive' Class : Yes   
##

The Accuracy rate is .86, Sensitivity is .17 and Specificity is .94. The accuracy rate is lower than on the Train data and it is a lower rate than the .88 of the naive model. This could indicate that the model is overfitted.

#### Task 8: Read in the “Blood.csv” dataset

library(readr)  
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()  
## )

View(Blood)

Convert the DonatedMarch variable to a factor and recode the variable so 0 = “No” and 1 = “Yes”.

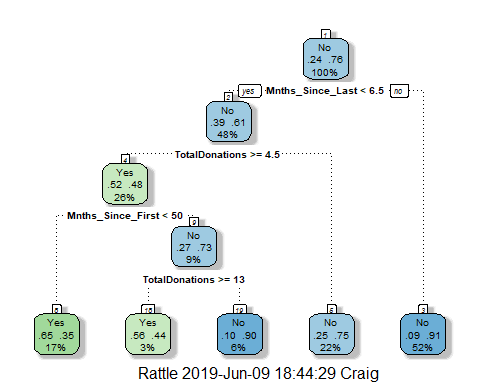
Blood = Blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%  
mutate(DonatedMarch = fct\_recode(DonatedMarch,  
"No" = "0",  
"Yes" = "1"))

#### Task 9: Split the dataset into training (70%) and testing (30%) sets

set.seed(1234)  
train.rows = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE)  
train2 = Blood[train.rows,]   
test2 = Blood[-train.rows,]

Then develop a classification tree on the training set to predict “DonatedMarch”.

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

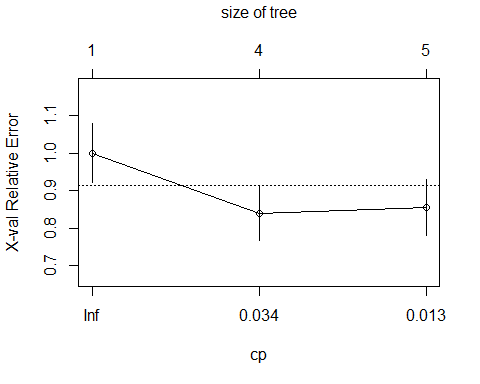


Evaluate the complexity parameter (cp) selection for this model.

printcp(tree3)

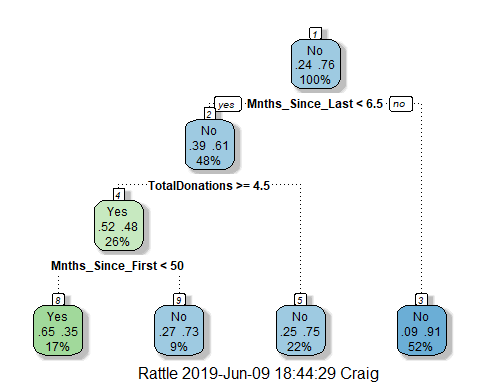
##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.072 0 1.000 1.000 0.078049  
## 2 0.016 3 0.784 0.840 0.073304  
## 3 0.010 4 0.768 0.856 0.073822

plotcp(tree3)

 The optimal CP value would be 0.16 rather than .01 based on the xerror score being lower.

#### 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

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



treepred3 = predict(tree4, train2, type = "class")  
head(treepred3)

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

confusionMatrix(treepred3,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 58 31  
## No 67 368  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4287   
##   
## Mcnemar's Test P-Value : 0.000407   
##   
## Sensitivity : 0.4640   
## Specificity : 0.9223   
## Pos Pred Value : 0.6517   
## Neg Pred Value : 0.8460   
## Prevalence : 0.2385   
## Detection Rate : 0.1107   
## Detection Prevalence : 0.1698   
## Balanced Accuracy : 0.6932   
##   
## 'Positive' Class : Yes   
##

treepred4 = predict(tree4, test2, type = "class")  
head(treepred4)

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

confusionMatrix(treepred4,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 14 16  
## No 39 155  
##   
## Accuracy : 0.7545   
## 95% CI : (0.6927, 0.8094)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.657104   
##   
## Kappa : 0.2006   
##   
## Mcnemar's Test P-Value : 0.003012   
##   
## Sensitivity : 0.2642   
## Specificity : 0.9064   
## Pos Pred Value : 0.4667   
## Neg Pred Value : 0.7990   
## Prevalence : 0.2366   
## Detection Rate : 0.0625   
## Detection Prevalence : 0.1339   
## Balanced Accuracy : 0.5853   
##   
## 'Positive' Class : Yes   
##

Comment on the quality of the predictions.

The accuracy of .75 is lower than the naive model .76 which could indicate the model is overfitted. In addition, the p-value of the training data set is much less than .05 and this would indicate that there is not a gain in accuracy.