## Classification Tree Assignment 1

Katherine Mobley

library(tidyverse)

##

## v ggplot2 3.1.0 v purrr 0.2.5  
## v tibble 1.4.2 v dplyr 0.7.7  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.1.1 v forcats 0.3.0

##   
## 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 5.2.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

Parole = read\_csv("parole (2).csv")

## Parsed with column specification:  
## cols(  
## male = col\_integer(),  
## race = col\_integer(),  
## age = col\_double(),  
## state = col\_integer(),  
## time.served = col\_double(),  
## max.sentence = col\_integer(),  
## multiple.offenses = col\_integer(),  
## crime = col\_integer(),  
## violator = col\_integer()  
## )

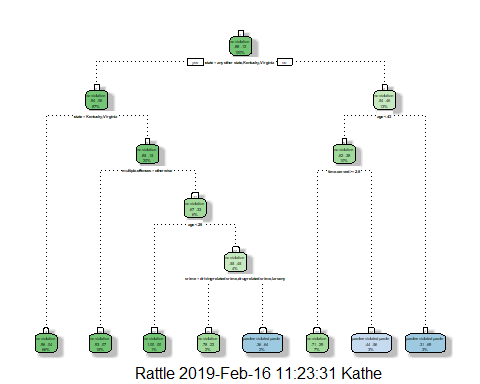
Parole = Parole %>% mutate(male = as\_factor(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"male" = "1",  
"female" = "0"))  
  
Parole = Parole %>% mutate(race = as\_factor(as.character(race))) %>%  
mutate(race = fct\_recode(race,  
"white" = "1",  
"otherwise" = "2"))  
  
Parole = Parole %>% mutate(state = as\_factor(as.character(state))) %>%  
mutate(state = fct\_recode(state,  
"any other state" = "1",  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4"))  
  
Parole = Parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"any other crime" = "1",  
"larceny" = "2",  
"drug-related crime" = "3",  
"driving-related crime" = "4"))  
  
Parole = Parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"incarcerated" = "1",  
"otherwise" = "0"))  
  
Parole = Parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"parolee violated parole" = "1",  
"no violation" = "0"))

# Task 1 Split training and testing set

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

# Task 2 Create Classification tree to predict violator

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

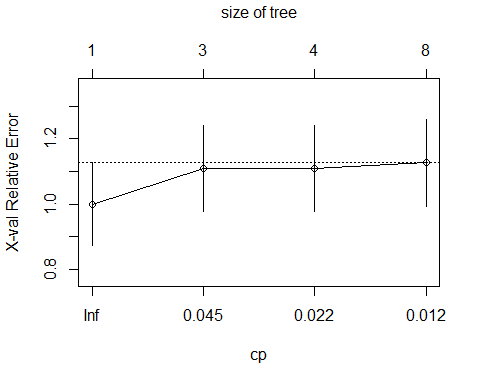


# Task 3

First we arrive at the state clasification, it ask if the person id from states kentucky, Virginia, or any other state. We know Louisiana is a category so it does not appear at this question, we would say no and follow that trail. Next question is multiple ofense. This person has multiple sentence because his time served was 5 years.He then goes to box 28, where he has had no violations and is apart of 4% of the no violation parolees.

# Task 4

plotcp(tree1)



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

The CP value selected should be 0.0100 this gives the most validation. In the video you referred to this CP value as the lowest plotcp() would go, but gives us the most validation.

# Task 5

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

In the dataset the class of violator has the majority or the most observations.

# Task 6

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

## 1 2 3 4 5   
## no violation no violation no violation no violation no violation   
## 6   
## no violation   
## Levels: no violation parolee violated parole

confusionMatrix(treepred\_test,train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation parolee violated parole  
## no violation 402 28  
## parolee violated parole 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 : 0.9617   
## Specificity : 0.4909   
## Pos Pred Value : 0.9349   
## Neg Pred Value : 0.6279   
## Prevalence : 0.8837   
## Detection Rate : 0.8499   
## Detection Prevalence : 0.9091   
## Balanced Accuracy : 0.7263   
##   
## 'Positive' Class : no violation   
##

The accuracy of this data set is 90.7%. The sensitivity of data set is 96.17%. The specificity is 49.09%.

# Task 7

treepred\_test1 = predict(tree1, test, type = "class")  
head(treepred\_test1)

## 1 2 3   
## no violation parolee violated parole no violation   
## 4 5 6   
## no violation no violation no violation   
## Levels: no violation parolee violated parole

confusionMatrix(treepred\_test1,test$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation parolee violated parole  
## no violation 170 19  
## parolee violated parole 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 : 0.9497   
## Specificity : 0.1739   
## Pos Pred Value : 0.8995   
## Neg Pred Value : 0.3077   
## Prevalence : 0.8861   
## Detection Rate : 0.8416   
## Detection Prevalence : 0.9356   
## Balanced Accuracy : 0.5618   
##   
## 'Positive' Class : no violation   
##

The accuracy of this data set is 86.14%. The specifcity of the data set is 17.39%. The sensitivity of the dataset is 94.97%,

# Task 8 Read in Blood Data set

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

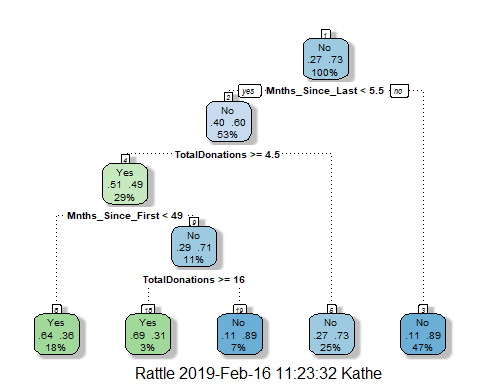
## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_integer(),  
## TotalDonations = col\_integer(),  
## Total\_Donated = col\_integer(),  
## Mnths\_Since\_First = col\_integer(),  
## DonatedMarch = col\_integer()  
## )

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

# Task 9

set.seed(1234)  
train.row2 = createDataPartition(y= Blood$DonatedMarch, p=0.7, list = FALSE)   
train2 = Blood[train.row,]  
test2 = Blood[-train.row,]

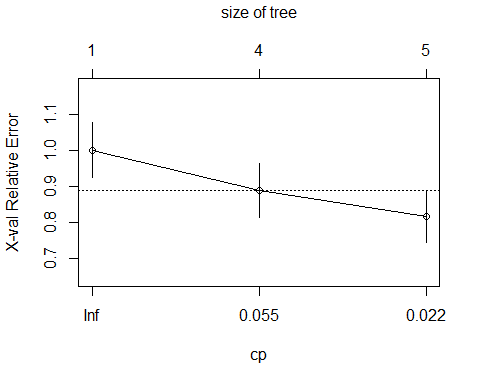
tree1B = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree1B)



printcp(tree1B)

##   
## 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: 126/473 = 0.26638  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.063492 0 1.00000 1.00000 0.076304  
## 2 0.047619 3 0.80952 0.88889 0.073377  
## 3 0.010000 4 0.76190 0.81746 0.071239

plotcp(tree1B)



This model shows us the best CP is actuall 0.055 because on the graph it is right on point with the relative error.

# Task 10

tree4B = prune(tree1B,cp= tree1B$cptable[which.min(tree1B$cptable[,"xerror"]),"CP"])

Prediction on Test Data Set

treepred\_test4B = predict(tree1B, test2, type = "class")  
head(treepred\_test4B)

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

Predictions on Train Data Set

treepred\_train4B = predict(tree1B, train2, type = "class")  
head(treepred\_train4B)

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

Confusion Matrix

confusionMatrix(treepred\_test4B,test2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 18 19  
## No 34 204  
##   
## Accuracy : 0.8073   
## 95% CI : (0.7556, 0.8522)  
## No Information Rate : 0.8109   
## P-Value [Acc > NIR] : 0.59717   
##   
## Kappa : 0.2934   
## Mcnemar's Test P-Value : 0.05447   
##   
## Sensitivity : 0.34615   
## Specificity : 0.91480   
## Pos Pred Value : 0.48649   
## Neg Pred Value : 0.85714   
## Prevalence : 0.18909   
## Detection Rate : 0.06545   
## Detection Prevalence : 0.13455   
## Balanced Accuracy : 0.63048   
##   
## 'Positive' Class : Yes   
##

confusionMatrix(treepred\_train4B,train2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 65 35  
## No 61 312  
##   
## Accuracy : 0.797   
## 95% CI : (0.7579, 0.8324)  
## No Information Rate : 0.7336   
## P-Value [Acc > NIR] : 0.0008251   
##   
## Kappa : 0.4442   
## Mcnemar's Test P-Value : 0.0107244   
##   
## Sensitivity : 0.5159   
## Specificity : 0.8991   
## Pos Pred Value : 0.6500   
## Neg Pred Value : 0.8365   
## Prevalence : 0.2664   
## Detection Rate : 0.1374   
## Detection Prevalence : 0.2114   
## Balanced Accuracy : 0.7075   
##   
## 'Positive' Class : Yes   
##

In the testing data set there is an accuracy of 80.73%. There is a sensitivity of 34.61% for testing data. For specficity there is a 91.48% for the testing data.

In the training data set there is 79.7% accuracy. There is a sensititivty of 51.59% for training data. Specificty is a 89.91% for training dataset.