# BAN 502

## Kyle Capponcelli

### Module\_4 Assignment 2 Classification Trees

Library Load

library("tidyverse", quietly = TRUE)

## -- Attaching packages ---------------------- tidyverse 1.3.0 --

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 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("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")

Data Load

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()  
## )

Data Conversion

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", "other" = "2"))   
   
parole = parole %>% mutate(state = as.factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "OtherState" = "1"))  
   
parole = parole %>% mutate(crime = as.factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related" = "3", "driving-related" = "4", "OtherCrime" = "1"))   
  
parole = parole %>% mutate(multiple.offenses = as.factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "MultipleOffense" = "1", "Other" = "0"))  
  
parole = parole %>% mutate(violator = as.factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator, "violated" = "1", "completed" = "0"))  
  
str(parole)

## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and '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","other": 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 : num 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "Other","MultipleOffense": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "OtherCrime","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 ...

#### Task 1

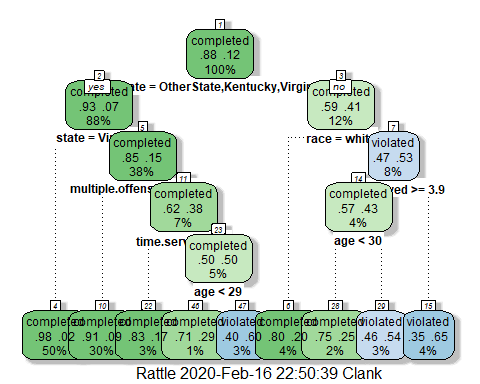
Data Split

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

#### Task 2

Classification Tree

tree1 = rpart(violator~., train, method = "class")  
fancyRpartPlot(tree1, cex = .75)



#### Task 3

I would classify a 40 year-old parolee from Louisiana with a 5 year prison sentence as a potential parole violator. Going through the classification tree we went to the right (No, the parolee is not from OtherState, Kentucky or Virginia). We do not know the persons race but if we go to the left the other factors don’t matter so I went to the right. Time served was greater than 3.9 years so I went left (Yes) to age. Age is greater than 30 so I went to the right (No) and ended up with a result of violating parole (3% of the data population fits in this category).

#### Task 4

Complexity Parameter Value

printcp(tree1)

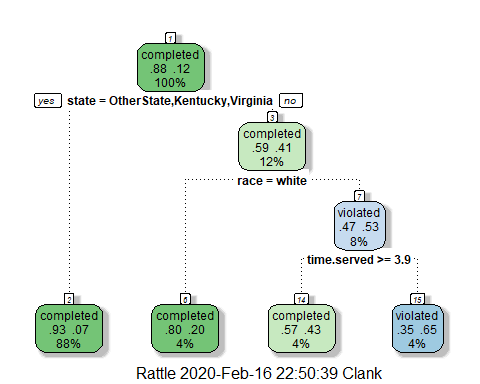
##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age multiple.offenses race state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit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

0.018182 should be used as this gives us the lowest xerror for our classification tree.

#### Task 5

Tree Pruning

PrunedTree = rpart(violator~., train, cp = 0.018182, method = "class")  
fancyRpartPlot(PrunedTree, cex = .75)



State = OtherState, Kentucky, Virgina (Yes) has the most data as it contains 88% of the data, making State a key indicator for parole violation prediction.

#### Task 6

Training Data Predictions

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

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

Confusion Matrix

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

## 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 : 0.49091   
## 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   
##

The accuracy of the training model from this data is 90.27% which means this is a fairly accurate model. The sensitivity is 49.091% and the specificity is 95.694%. Since sensitivity is relatively low, we may want to consider a modified model. This is essentially means that half of those we predict will violate parole, do not violate parole. With a high specificity score, we can see that this model is still viable when the majority of the data (in this case, predicted to complete parole and having those individuals complete parole). No values are significant (<0.05).

#### Task 7

Testing Data Predictions

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

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

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

## 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 : 0.43478   
## 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   
##

When testing the training model against the test data we find that we have slightly less accuracy (89.6%). We also get a sensitivity of 43.478% (less than half of the violator predictions are innaccurate) and a specfificity of 95.531% (the majority of those predicted to complete parole, do). Since this model is being constructed to predict those that do violate parole, I would say that this is a poor model since we misidentify violators over half the time. Detection rate was the only significant value at 0.04950.

#### Task 8

Blood read-in

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)

DonatedMarch Conversion

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

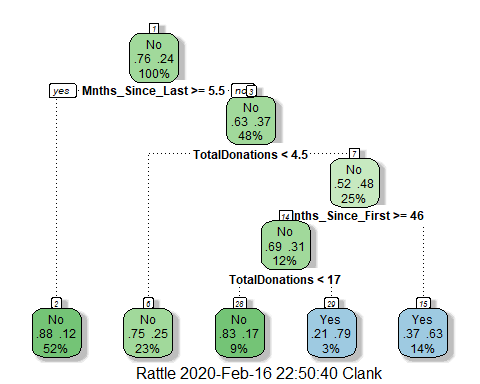
#### Task 9

Data Split

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

Training Classification Tree

tree2 = rpart(DonatedMarch ~., train2, method = "class")  
fancyRpartPlot(tree2, cex = .75)



Complexity Parameter

printcp(tree2)

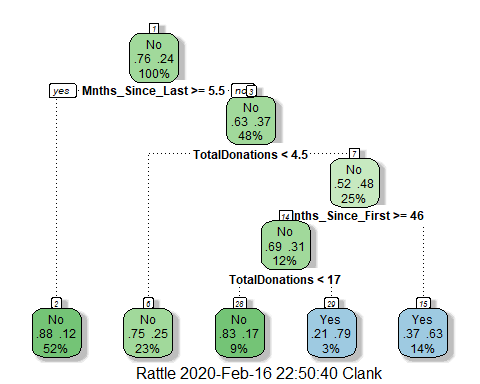
##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.050667 0 1.000 1.000 0.078049  
## 2 0.010000 4 0.784 0.848 0.073564

The complexity parameter should be set to 0.01000 as this shows the lowest xerror for the data. Ultimately, this means that the model cannot be modified further by removing categories as this would damage the model.

#### Task 10

Tree Pruning

PrunedTree2 = rpart(DonatedMarch ~., train2, cp = 0.01000, method = "class")  
fancyRpartPlot(PrunedTree2, cex = .75)



Training Predictions and Confustion Matrix

treepred3 = predict(PrunedTree2, train2, type = "class")  
head(treepred3)

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

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

## 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   
##

For the training data set, there is an accuracy measure of 81.3%. This is not the most desirable measure of accuracy as it leaves a little less than 20% of the data to be incorrectly placed. Sensitivity is low (44.8%) and specificity is decent at 92.73% (meaning that although this isn’t the most accurate model we are still correctly placing the majority of the data that we say will not be donating blood. No values were significant (<0.05).

Testing Predictions and Confusion Matrix

treepred4 = predict(PrunedTree2, test2, type = "class")  
head(treepred4)

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

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

## 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   
##

For the testing data, our accuracy dropped even further when using the pruned classification tree. We now have an accuracy of 77.68% (I would not say that this was great), a sensitivity of 33.962% and a specificity of 91.228%. This new data when tested with the model built by the training data performed even worse. I would consider other model building methods to predict whether or not somone will donate blood. No values were siginifcant (<0.05).