# Module 4 Assignment 1 Classification Trees

## BAN 502

### Jennifer Brill

library(tidyverse)

## ── Attaching packages ──────────────────────────────────────────────────────── tidyverse 1.2.1 ──

## ✔ ggplot2 3.1.0 ✔ purrr 0.2.5  
## ✔ tibble 1.4.2 ✔ dplyr 0.7.8  
## ✔ tidyr 0.8.2 ✔ stringr 1.3.1  
## ✔ readr 1.1.1 ✔ forcats 0.3.0

## ── Conflicts ─────────────────────────────────────────────────────────── tidyverse\_conflicts() ──  
## ✖ dplyr::filter() masks stats::filter()  
## ✖ 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)

library(readr)  
parole <- read\_csv("parole (1).csv")

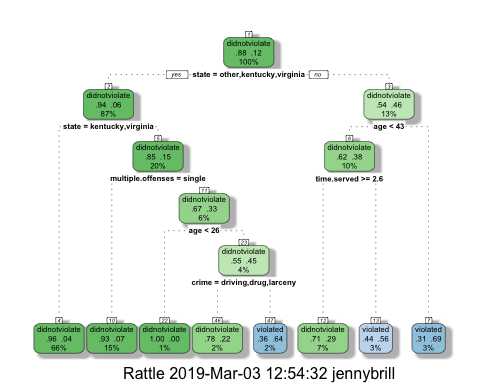
## Parsed with column specification:  
## cols(  
## male = col\_integer(),  
## race = col\_integer(),  
## age = col\_double(),  
## state = col\_integer(),  
## time.served = col\_double(),  
## max.sentence = col\_integer(),  
## multiple.offenses = col\_integer(),  
## crime = col\_integer(),  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

parole = parole %>%  
 mutate(male = as\_factor(as.character(male))) %>%  
 mutate(male = fct\_recode(male, "male" = "1", "female" = "0")) %>%  
 mutate(race = as\_factor(as.character(race))) %>%  
 mutate(race = fct\_recode(race, "white" = "1", "otherwise" = "2")) %>%  
 mutate(state = as\_factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state, "kentucky" = "2", "louisiana" = "3", "virginia" = "4", "other" = "1")) %>%  
 mutate(crime = as\_factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime, "larceny" = "2", "drug" = "3", "driving" = "4", "other" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "multiple" = "1", "single" = "0")) %>%  
 mutate(violator = as\_factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator, "violated" = "1", "didnotviolate" = "0"))  
str(parole)

## Classes 'tbl\_df', 'tbl' and '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 "other","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 "single","multiple": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "driving","drug",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "didnotviolate",..: 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,]

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

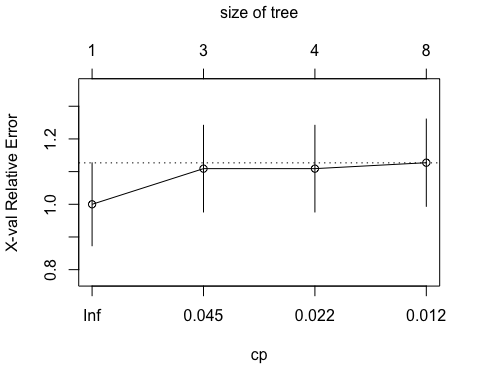


A 40 year-old parolee from Lousiana who served a 5 year prison sentence would not be classified as someone who would be likely to violate parole. In order to come to this decision, you must start at the top of the classification tree. The first box asks if the person is from Kentucky, Virginia, or a state classified as “other”. This person is not so you move to the right. The next box asks if ther person is under the age of 43 which this person is, so you move down and to the left. The person did serve more than 2.6 years in prison, so you move down and to the left, which puts the person in the did not violate classifaction.

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)



The CP value that should be selected is 0.03.

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

The majority class in the training set is did not violate.

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

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction didnotviolate violated  
## didnotviolate 402 28  
## violated 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 : violated   
##

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

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction didnotviolate violated  
## didnotviolate 170 19  
## violated 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 : violated   
##

The model seems to be very accurate. The accuracy and no information rate between the training and testing set are very similar. It does not appear that we overfit the data with the model.

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

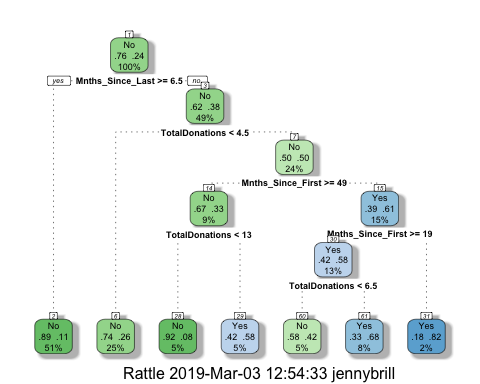
## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_integer(),  
## TotalDonations = col\_integer(),  
## Total\_Donated = col\_integer(),  
## Mnths\_Since\_First = col\_integer(),  
## DonatedMarch = col\_integer()  
## )

blood <- blood %>%  
 mutate(DonatedMarch = as.factor(DonatedMarch)) %>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1"))  
str(blood)

## Classes 'tbl\_df', 'tbl' and '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 ...

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

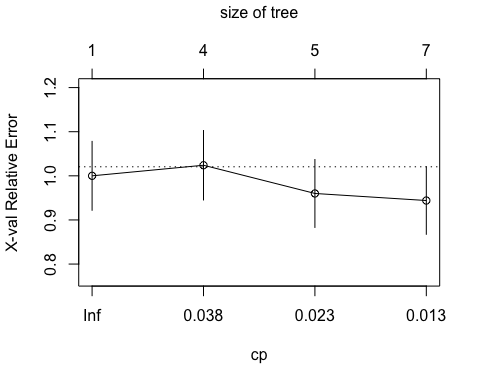
tree3 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree3)



printcp(tree3)

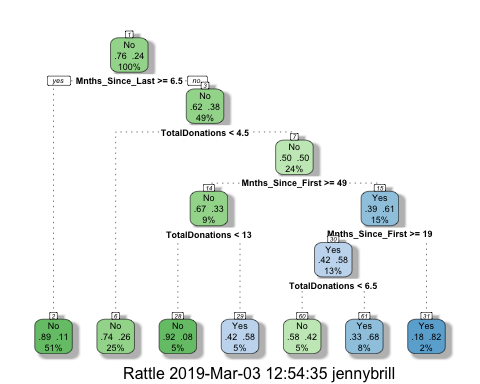
##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 125/524 = 0.23855  
##   
## n= 524   
##   
## CP nsplit rel error xerror xstd  
## 1 0.045333 0 1.000 1.000 0.078049  
## 2 0.032000 3 0.864 1.024 0.078682  
## 3 0.016000 4 0.832 0.960 0.076949  
## 4 0.010000 6 0.800 0.944 0.076494

plotcp(tree3)



The best CP for this model is the default, 0.01. This gives us the lowest cross-validated error.

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



treepred2 = predict(tree4, train2, type = "class")  
head(treepred2)

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 374 75  
## Yes 25 50  
##   
## Accuracy : 0.8092   
## 95% CI : (0.7729, 0.8419)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.005169   
##   
## Kappa : 0.3911   
## Mcnemar's Test P-Value : 9.584e-07   
##   
## Sensitivity : 0.40000   
## Specificity : 0.93734   
## Pos Pred Value : 0.66667   
## Neg Pred Value : 0.83296   
## Prevalence : 0.23855   
## Detection Rate : 0.09542   
## Detection Prevalence : 0.14313   
## Balanced Accuracy : 0.66867   
##   
## 'Positive' Class : Yes   
##

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

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 160 32  
## Yes 11 21  
##   
## Accuracy : 0.808   
## 95% CI : (0.7503, 0.8575)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.065180   
##   
## Kappa : 0.3845   
## Mcnemar's Test P-Value : 0.002289   
##   
## Sensitivity : 0.39623   
## Specificity : 0.93567   
## Pos Pred Value : 0.65625   
## Neg Pred Value : 0.83333   
## Prevalence : 0.23661   
## Detection Rate : 0.09375   
## Detection Prevalence : 0.14286   
## Balanced Accuracy : 0.66595   
##   
## 'Positive' Class : Yes   
##

The model used to predit if a person do nated blood in March, appears to be an accurate model. The Accuracy percentage and No Information Rate between the training and testing model are very close together and the accuracy percentage is high at 80%.