# John R. Cobb

# Ban 502

# 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(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 (1).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)

str(parole)

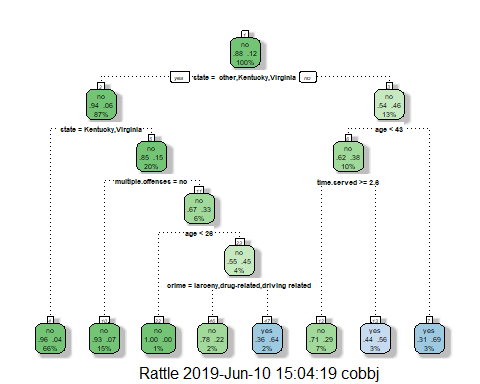
## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : num 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : num 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : 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: num 0 0 0 0 0 0 0 0 0 0 ...  
## $ crime : num 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : num 0 0 0 0 0 0 0 0 0 0 ...  
## - attr(\*, "spec")=  
## .. 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()  
## .. )

summary(parole)

## male race age state   
## Min. :0.0000 Min. :1.000 Min. :18.40 Min. :1.000   
## 1st Qu.:1.0000 1st Qu.:1.000 1st Qu.:25.35 1st Qu.:2.000   
## Median :1.0000 Median :1.000 Median :33.70 Median :3.000   
## Mean :0.8074 Mean :1.424 Mean :34.51 Mean :2.887   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.:42.55 3rd Qu.:4.000   
## Max. :1.0000 Max. :2.000 Max. :67.00 Max. :4.000   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 Min. :0.0000 Min. :1.000   
## 1st Qu.:3.250 1st Qu.:12.00 1st Qu.:0.0000 1st Qu.:1.000   
## Median :4.400 Median :12.00 Median :1.0000 Median :2.000   
## Mean :4.198 Mean :13.06 Mean :0.5363 Mean :2.059   
## 3rd Qu.:5.200 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:3.000   
## Max. :6.000 Max. :18.00 Max. :1.0000 Max. :4.000   
## violator   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.1156   
## 3rd Qu.:0.0000   
## Max. :1.0000

parole = parole %>% mutate(male = as.factor(male)) %>%   
 mutate(male = fct\_recode(male, "no" = "0", "yes"= "1"))   
parole = parole %>% mutate(race = as.factor(race)) %>%   
 mutate(race = fct\_recode(race, "white" = "1", "not white" = "2" ))  
parole = parole %>% mutate(state = as.factor(state)) %>%   
 mutate(state = fct\_recode(state, " other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4" ))  
parole = parole %>% mutate(crime = as.factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "other"= "1", "larceny" = "2", "drug-related" = "3", "driving related" = "4" ))  
parole = parole %>% mutate(multiple.offenses = as.factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "no" = "0", "yes" = "1" ))  
parole = parole %>% mutate(violator = as.factor(violator)) %>%   
 mutate(violator = fct\_recode(violator,"no" = "0", "yes" = "1" ))  
set.seed(12345)   
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE)   
train = parole[train.rows,]   
test = parole[-train.rows,]

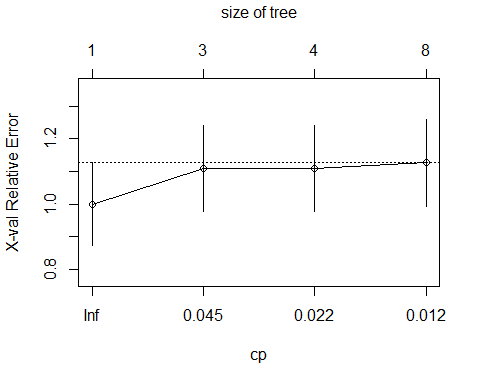
tree1 = rpart(violator ~., train, method="class")  
fancyRpartPlot(tree1)

 I would classify a 40 year old from Louisiana who served a 5 year sentence as a non-violator. As far as this classification tree goes, I chose no to other, vitginia, or kentucky, thn I chose yes to less than 43 years old, then I chose yes to greter than or equal to 5 years in prison. This process led me to a decision of non-violator.

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)

 it appears that an infinite cp value should be used

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

the majority category falls into the non-violator category

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

## 1 2 3 4 5 6   
## no no no no no no   
## Levels: no yes

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred,train$violator)

## 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 : 0.9617   
## 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   
##

the model using the training data had an accuracy of .907, whereas the naive model had an accuracy of .8837 which is a marginal improvement. the sensitivity is .9617 and the specificity is .4909

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

## 1 2 3 4 5 6   
## no yes no no no no   
## Levels: no yes

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_test,test$violator) #predictions first then actual

## 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 : 0.9497   
## 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   
##

compared to the traing model, the testing model fell below the threshhold of the naive model. In this case the accuracy of the test model was .86 and the naive model was .89. also in both cases the pvalue was not signficantly significant, this of course could be from a relatively small data set. I still intuitively don’t think it would be best to run with the naive model assuming that most of the population would be non-violators. Im not sure if the 4 percent difference in testing and training accuracy matters all that much, I still like the idea of using and visualizing adhoc predictions instead of using a blanket naive prediction. I think it is a good model

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

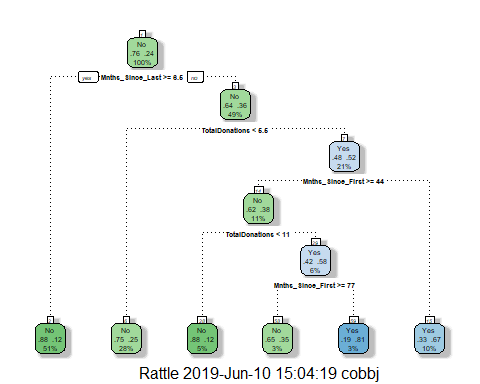
## Parsed with column specification:  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

View(Blood)

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

set.seed(12345)   
train.rows = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE)   
train2 = Blood[train.rows,]   
test2 = Blood[-train.rows,]

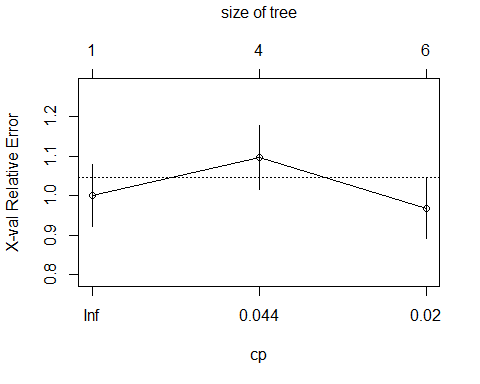
tree3 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree3)



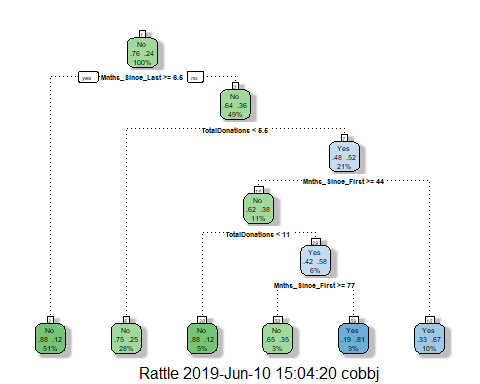
printcp(tree3)

##   
## 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 0.048 0 1.000 1.000 0.078049  
## 2 0.040 3 0.856 1.096 0.080471  
## 3 0.010 5 0.776 0.968 0.077174

plotcp(tree3)



tree4 = prune(tree3,cp= tree3$cptable[which.min(tree3$cptable[,"xerror"]),"CP"])  
#most of the code in the line above can be left untouched. Just change tree1 to the name of your tree model (if it's not called tree1)  
fancyRpartPlot(tree3)



Predictions on training set

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

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred2,train2$DonatedMarch) #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 379 77  
## Yes 20 48  
##   
## Accuracy : 0.8149   
## 95% CI : (0.7789, 0.8472)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.001931   
##   
## Kappa : 0.3959   
##   
## Mcnemar's Test P-Value : 1.301e-08   
##   
## Sensitivity : 0.9499   
## Specificity : 0.3840   
## Pos Pred Value : 0.8311   
## Neg Pred Value : 0.7059   
## Prevalence : 0.7615   
## Detection Rate : 0.7233   
## Detection Prevalence : 0.8702   
## Balanced Accuracy : 0.6669   
##   
## 'Positive' Class : No   
##

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

## 1 2 3 4 5 6   
## no yes no no no no   
## Levels: no yes

confusionMatrix(treepred\_test2,test2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 153 36  
## Yes 18 17  
##   
## Accuracy : 0.7589   
## 95% CI : (0.6975, 0.8134)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.5982   
##   
## Kappa : 0.2441   
##   
## Mcnemar's Test P-Value : 0.0207   
##   
## Sensitivity : 0.8947   
## Specificity : 0.3208   
## Pos Pred Value : 0.8095   
## Neg Pred Value : 0.4857   
## Prevalence : 0.7634   
## Detection Rate : 0.6830   
## Detection Prevalence : 0.8438   
## Balanced Accuracy : 0.6077   
##   
## 'Positive' Class : No   
##

From the training and testing set, it appears that compared to there is differene in accuracy, the training set accuracy was .81 and the naive was .76 which is better. the testing model accuracy was .7589 and the naive model was .76. it can be seen that the model difference in the testing model was not statistically significant, while the trainig model was, again, this is probably due to such a small data set. I think the models are a better fit for the data than the naive