Classification Trees

Chris Collins

February 14, 2019

# Libraries

#install.packages("rattle")  
#install.packages("rpart")  
library(tidyverse)

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

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

## -- 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(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.

# Load Data

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()  
## )

# Structure and Summary

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

# Data Cleaning

parole <- parole %>%   
 mutate(male = recode\_factor(male, '1' = 'Male', '0' = 'Female')) %>%   
 mutate(race = recode\_factor(race, '1' = 'White', '2' = 'Other')) %>%   
 mutate(state = recode\_factor(state, '1' = 'Other', '2' = 'Kentucky', '3' = 'Louisiana', '4' = 'Virginia')) %>%   
 mutate(multiple.offenses = recode\_factor(multiple.offenses, '1' = 'Yes', '0' = 'No')) %>%   
 mutate(crime = recode\_factor(crime, '1' = 'Other', '2' = 'Larceny', '3' = 'Drug-related', '4' = 'Driving-related')) %>%   
 mutate(violator = recode\_factor(violator, '1' = 'Yes', '0' = 'No'))  
  
parole=parole%>%drop\_na()  
str(parole)

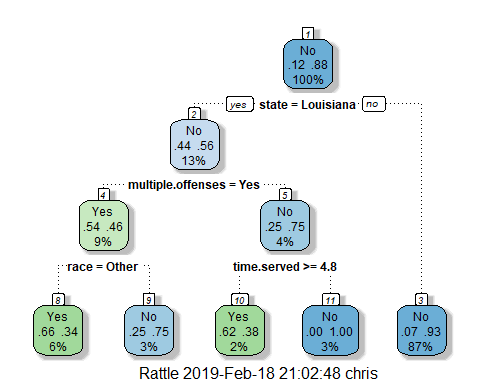
## Classes 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : Factor w/ 2 levels "Male","Female": 1 2 1 1 1 1 1 2 2 1 ...  
## $ race : Factor w/ 2 levels "White","Other": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "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 : num 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 2 ...  
## $ crime : Factor w/ 4 levels "Other","Larceny",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "Yes","No": 2 2 2 2 2 2 2 2 2 2 ...

# Task 1: Split: Train/Test

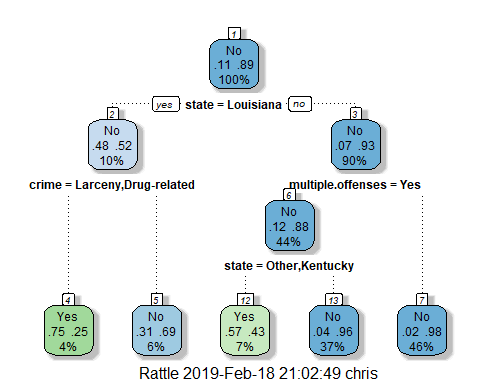
set.seed(12345)  
train.rows=createDataPartition(y=parole$violator,p=0.7,list=FALSE)  
train=parole[train.rows,]  
test=parole[-train.rows,]

# Task 2: Classifaction Tree - Violator

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



tree2 = rpart(violator ~., test, method="class")  
fancyRpartPlot(tree2)

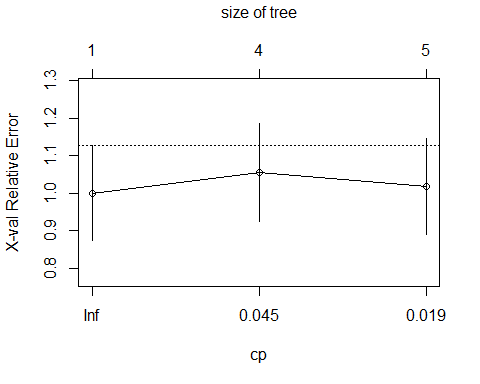
 #Task 3: #40 year old from Louisiana would go down the left side of the Classification Tree because the parolee is from Louisiana. Then file into the greater than 4.8 years time.served column. There is a 2% chance of this type of parolee, 62% of them are violators and 38% of them are non violators.

# Task 4: Plot CP

printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] multiple.offenses race state 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 3 0.83636 1.0545 0.12970  
## 3 0.010000 4 0.80000 1.0182 0.12775

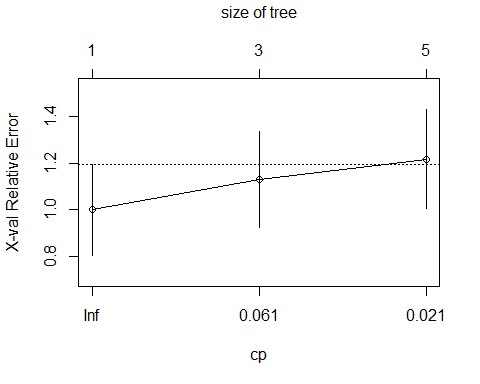
plotcp(tree1)



printcp(tree2)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = test, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] crime multiple.offenses state   
##   
## Root node error: 23/202 = 0.11386  
##   
## n= 202   
##   
## CP nsplit rel error xerror xstd  
## 1 0.086957 0 1.00000 1.0000 0.19628  
## 2 0.043478 2 0.82609 1.1304 0.20694  
## 3 0.010000 4 0.73913 1.2174 0.21353

plotcp(tree2)



# Task 4:

# For the training set, CP Value of 0.05 should be selected because it is the minimum cross-validation value of 1.00 in the graph because as the classification tree splits, the cross-validation value increases.

# Task 5: Prune the Tree

#tree3 = prune(tree1,cp= tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
#fancyRpartPlot(tree3)  
  
#tree4 = prune(tree2,cp= tree2$cptable[which.min(tree2$cptable[,"xerror"]),"CP"])  
#fancyRpartPlot(tree4)

# The class in the training set with the majority class is n/a.

# 6: Tree Predictions

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

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 24 13  
## No 31 405  
##   
## Accuracy : 0.907   
## 95% CI : (0.8771, 0.9316)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.06272   
##   
## Kappa : 0.4724   
## Mcnemar's Test P-Value : 0.01038   
##   
## Sensitivity : 0.43636   
## Specificity : 0.96890   
## Pos Pred Value : 0.64865   
## Neg Pred Value : 0.92890   
## Prevalence : 0.11628   
## Detection Rate : 0.05074   
## Detection Prevalence : 0.07822   
## Balanced Accuracy : 0.70263   
##   
## 'Positive' Class : Yes   
##

treepred\_test = predict(tree2, newdata=test, type = "class")  
head(treepred\_test)

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 14 8  
## No 9 171  
##   
## Accuracy : 0.9158   
## 95% CI : (0.8687, 0.9502)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.1084   
##   
## Kappa : 0.5749   
## Mcnemar's Test P-Value : 1.0000   
##   
## Sensitivity : 0.60870   
## Specificity : 0.95531   
## Pos Pred Value : 0.63636   
## Neg Pred Value : 0.95000   
## Prevalence : 0.11386   
## Detection Rate : 0.06931   
## Detection Prevalence : 0.10891   
## Balanced Accuracy : 0.78200   
##   
## 'Positive' Class : Yes   
##

# Task 6: Training Data Results:

# Accuracy = 0.907, Sensitivity = 0.436, Specificity = 0.969

# The quality of this model is average with a decently high average. The p-value is almost significant at 0.06, which is better than the testing data.

# Task 7: Testing Data Results:

# Accuracy = 0.9158, Sensitivity = 0.6087, Specificity = 0.9553

# The quality of this model is average. There is a higher level of accuracy but the p-value shows that it is not significant at a 0.10.

# Task 8 Read-in Dataset

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()  
## )

# Data Cleaning

blood <- blood %>%   
 mutate(DonatedMarch = recode\_factor(DonatedMarch, '1' = 'Yes', '0' = 'No'))  
  
blood=blood%>%drop\_na()  
str(blood)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "Yes","No": 1 1 1 1 2 2 1 2 1 1 ...

# Structure and Summary

str(blood)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "Yes","No": 1 1 1 1 2 2 1 2 1 1 ...

summary(blood)

## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :34.28   
## 3rd Qu.:14.000 3rd Qu.: 7.000 3rd Qu.: 1750 3rd Qu.:50.00   
## Max. :74.000 Max. :50.000 Max. :12500 Max. :98.00   
## DonatedMarch  
## Yes:178   
## No :570   
##   
##   
##   
##

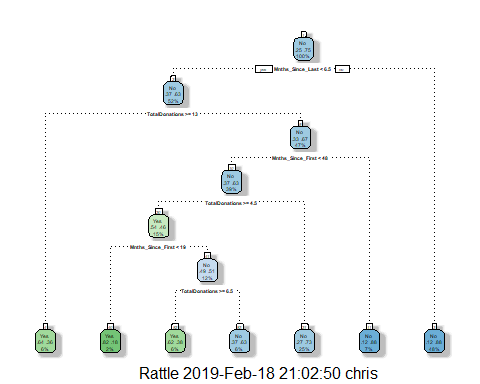
# Task 9:

# Split into Training/Testing

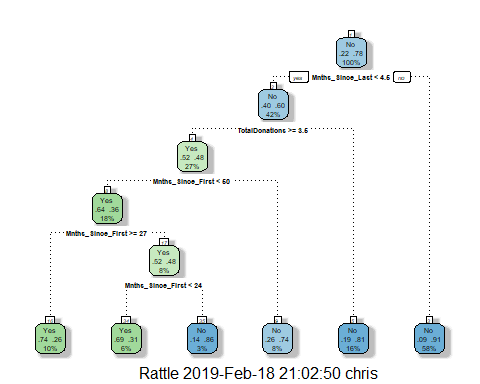
set.seed(1234)  
train.rows2=createDataPartition(y=blood$DonatedMarch,p=0.7,list=FALSE)  
train2=blood[train.rows,]  
test2=blood[-train.rows,]

# Classification Tree

tree5 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree5)



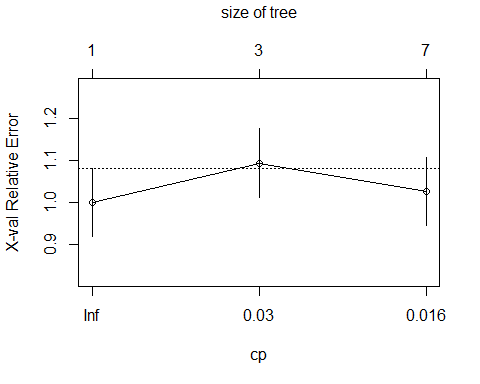
tree6 = rpart(DonatedMarch ~., test2, method="class")  
fancyRpartPlot(tree6)



printcp(tree5)

##   
## 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: 117/473 = 0.24736  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.034188 0 1.00000 1.0000 0.080205  
## 2 0.025641 2 0.93162 1.0940 0.082584  
## 3 0.010000 6 0.81197 1.0256 0.080884

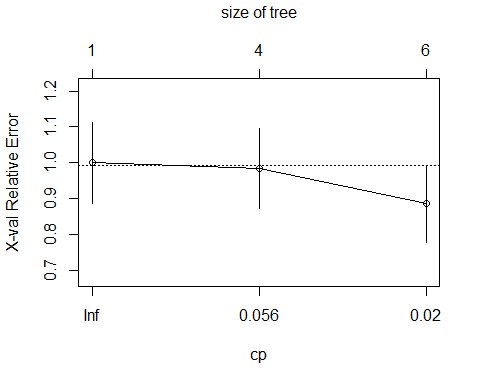
plotcp(tree5)



printcp(tree6)

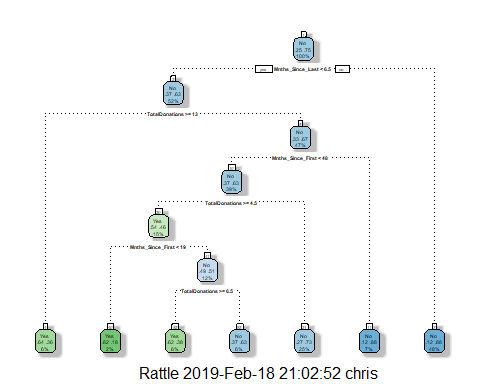
##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = test2, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 61/275 = 0.22182  
##   
## n= 275   
##   
## CP nsplit rel error xerror xstd  
## 1 0.076503 0 1.00000 1.00000 0.11295  
## 2 0.040984 3 0.77049 0.98361 0.11228  
## 3 0.010000 5 0.68852 0.88525 0.10799

plotcp(tree6)

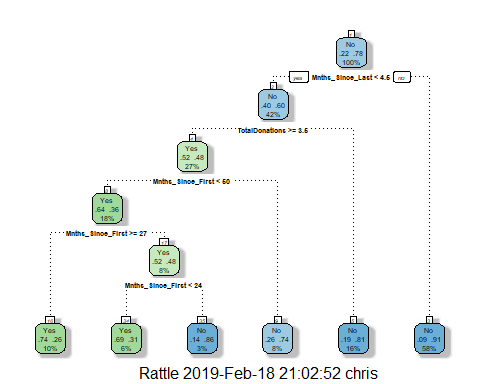
 #For the training set, CP Value of 0.034 should be selected because it is the minimum cross-validation value of 1.00 in the graph because as the classification tree splits, the cross-validation value increases, then slightly decreases, but remains higher than 1.00. #For the testing set, CP Value of 0.01 should be selected because it is the minimum cross-validation value of 0.885 in the graph with an nsplit of 5.

# Task 10: Prune

tree7 = prune(tree5,cp= tree5$cptable[which.min(tree5$cptable[,"xerror"]),"CP"])  
fancyRpartPlot(tree5)



tree8 = prune(tree6,cp= tree6$cptable[which.min(tree6$cptable[,"xerror"]),"CP"])  
fancyRpartPlot(tree6)



#treepred2 = predict(tree5, train, type = "class")  
#head(treepred2)  
  
#confusionMatrix(treepred\_2,train$DonatedMarch,positive="Yes")  
  
#treepred\_test2 = predict(tree6, newdata=test, type = "class")  
#head(treepredtest2)  
  
#confusionMatrix(treepredtest\_2,test2$DonatedMarch,positive="Yes")

# Quality of Predictions is n/a