# Classification Trees

## Brian Fortier

### Module 4 Assignment 1

Loading Packages:

#install.packages("tidyverse")  
#install.packages("caret")  
#install.packages("rpart")  
#install.packages("rattle")  
#install.packages("RColorBrewer")  
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 0.7.7  
## v tidyr 0.8.2 v stringr 1.3.1  
## v readr 1.1.1 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(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(e1071)

Loading Parole and Conversions of Variables:

parole <- read.csv("parole.csv")  
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,  
 "Kentucky" = "2",  
 "Louisiana" = "3",  
 "Virginia" = "4",  
 "Other" = "1"))  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime,  
 "larceny" = "2",  
 "drug-related" = "3",  
 "driving-related" = "4",  
 "other" = "1"))  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
 "multiple" = "1",  
 "other" = "0"))  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator,  
 "violator" = "1",  
 "other" = "0"))  
parole = parole %>% drop\_na()  
glimpse(parole)

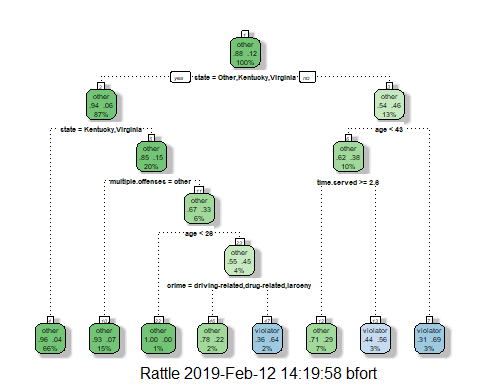
## Observations: 675  
## Variables: 9  
## $ male <fct> male, female, male, male, male, male, male, ...  
## $ race <fct> white, white, otherwise, white, otherwise, o...  
## $ age <dbl> 33.2, 39.7, 29.5, 22.4, 21.6, 46.7, 31.0, 24...  
## $ state <fct> Other, Other, Other, Other, Other, Other, Ot...  
## $ time.served <dbl> 5.5, 5.4, 5.6, 5.7, 5.4, 6.0, 6.0, 4.8, 4.5,...  
## $ max.sentence <int> 18, 12, 12, 18, 12, 18, 18, 12, 13, 12, 12, ...  
## $ multiple.offenses <fct> other, other, other, other, other, other, ot...  
## $ crime <fct> driving-related, drug-related, drug-related,...  
## $ violator <fct> other, other, other, other, other, other, ot...

Splitting Parole into Training/Testing Sets:

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

Building Classification Tree:

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



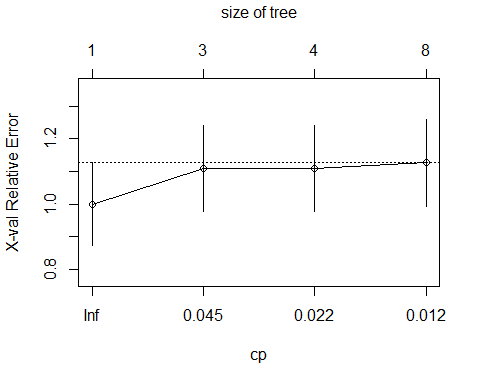
With our example of a 40 year-old parolee from Louisana who served a 5 year prison sentence here is where we would end up: Starting at the top, we are not from “other, Kentucky, or Virginia”. Moving to the right, we are less than 43. Moving to the left, we have served greater than or equal to 2.6 years. So we would move down to the left and be left with a non-violator.

Plot and Print our Tree to Evaluate Performance:

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)



Looking at our complexity parameter, we see 7 splits as we reach the lower bound of the parameter. However, this graph is different as we see the xerror increase as we lower our parameter, just like the std. This happens when we have the majority of the data fitting to the characteristic of a parolee not being a violator. We would observe our best CP to be near the initial point but I would put an estimation at 0.05 just to see the resulting tree.

Pruning the Tree (CP = 0.05):

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

State seems to have the most observations as it sits on the top of the tree and has the biggest impact on the resulting violator or other.

Predictions on Training Set:

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

## 1 3 4 5 6 7   
## other other other other other other   
## Levels: other violator

Our initial prediction on the first classification tree gives us all 7 resulting “other” predictions.

Confusion Matrix for Accuracy, Sensitivity, and Specificity on Training Set:

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction other violator  
## other 402 28  
## violator 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 : violator   
##

We have an accuracy of 90.7% with a no information rate of 88.37%. We also see the sensitivity to be faily high at 0.49091 with a specificity of 0.96172.

Predictions on Testing Set:

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

## 2 11 13 14 17 18   
## other violator other other other other   
## Levels: other violator

The testing set prediction gives us 1/7 as “violator”.

Confusion Matrix for Accuracy, Sensitivity, and Specificity on Testing Set:

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction other violator  
## other 170 19  
## violator 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 : violator   
##

Our accuracy on the testing set is a little lower at 86.14% which is expected with a smaller sample size. We also have a lower sensitivity at 0.17391 and specificity at 0.94972.

Loading Blood and Converting Variables:

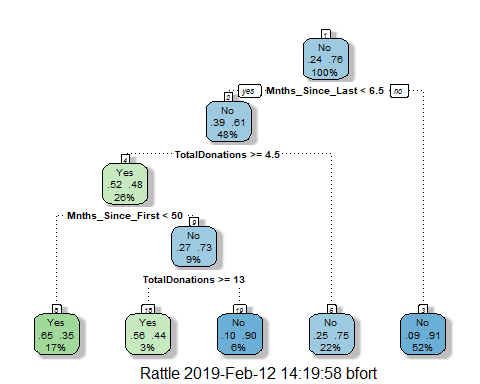
blood <- read.csv("Blood.csv")  
blood = blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%   
 mutate(DonatedMarch = fct\_recode(DonatedMarch,  
 "No" = "0",  
 "Yes" = "1"))

Splitting Blood into Training/Testing Sets:

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

Building Classification Tree for Blood:

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

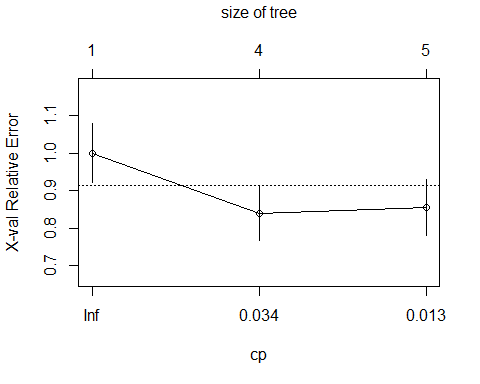


Plot and Print our Tree to Evaluate Performance:

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 0.072 0 1.000 1.000 0.078049  
## 2 0.016 3 0.784 0.840 0.073304  
## 3 0.010 4 0.768 0.856 0.073822

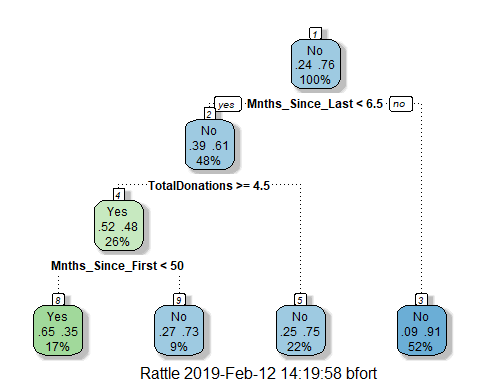
plotcp(tree3)



After looking at our CP graph, we would see the lowest point to be about at our 4th split which would be at 0.034. The print screen shows the true value of this split to be at 0.016. We will use this value for our pruning of the tree.

Pruning back tree (CP = 0.016):

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



This resulting tree has 3 splits between the variables: “Mnths\_Since\_Last”, “TotalDonations”, and “Mnths\_Since\_First”.

Prediction on Training Set:

treepred3 = predict(tree4, train2, type="class")  
head(treepred3)

## 2 3 5 6 7 9   
## Yes Yes No No Yes Yes   
## Levels: Yes No

we see from our prediction on the training set of our new tree, we have 4/6 predicting “Yes” for a Donation in March.

Confusion Matrix for Training Set:

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 58 31  
## No 67 368  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4287   
## Mcnemar's Test P-Value : 0.000407   
##   
## Sensitivity : 0.4640   
## Specificity : 0.9223   
## Pos Pred Value : 0.6517   
## Neg Pred Value : 0.8460   
## Prevalence : 0.2385   
## Detection Rate : 0.1107   
## Detection Prevalence : 0.1698   
## Balanced Accuracy : 0.6932   
##   
## 'Positive' Class : Yes   
##

Our confusion matrix for the training set gives us only 81.3% accuracy with sensitivity and specificity levels of 0.4640 and 0.9223, respectively. We also see our adjusted P-value is < 0.05 which helps us show that our new pruned model is an improvement.

Prediction on Testing Set:

treepred4 = predict(tree4, test2, type="class")  
head(treepred4)

## 1 4 8 11 12 17   
## No Yes Yes No No Yes   
## Levels: Yes No

Our testing set prediction shows us to have 3/6 showing a “Yes” for Donation in March.

Confusion Matrix for Testing set:

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 14 16  
## No 39 155  
##   
## Accuracy : 0.7545   
## 95% CI : (0.6927, 0.8094)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.657104   
##   
## Kappa : 0.2006   
## Mcnemar's Test P-Value : 0.003012   
##   
## Sensitivity : 0.2642   
## Specificity : 0.9064   
## Pos Pred Value : 0.4667   
## Neg Pred Value : 0.7990   
## Prevalence : 0.2366   
## Detection Rate : 0.0625   
## Detection Prevalence : 0.1339   
## Balanced Accuracy : 0.5853   
##   
## 'Positive' Class : Yes   
##

As we expected, our confusion matrix for the lower data set in the Testing set shows a lower accuracy percentage of 75.45% and sensitivity and specificity values of 0.2642 and 0.9064.

However, we do see in this pruned tree prediction our P-value is > 0.05 significantly which may indicate this isn’t the best model for our predictions.