M4 Assignment 1

# Module 4: Assignment 1 – Classification Trees

## Kellie McLiverty

### Assignment Needs & Data Importation

Libraries needed for Assignemnt

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 1.2.1 v forcats 0.3.0

## -- Conflicts ----------------------------------------------------------------------------------------- tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(rpart)

## Warning: package 'rpart' was built under R version 3.5.2

library(caret)

## Warning: package 'caret' was built under R version 3.5.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following object is masked from 'package:purrr':  
##   
## lift

library(rattle)

## Warning: package 'rattle' was built under R version 3.5.2

## 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)

## Warning: package 'RColorBrewer' was built under R version 3.5.2

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()  
## )

### Converting data to factors & setting training/testing data

#converting data into male or female  
parole = parole %>% mutate(male = as\_factor(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"Female" = "0",  
"Male" = "1"  
))  
  
#converting race into White or otherwise  
parole = parole %>% mutate(race = as\_factor(as.character(race))) %>%  
mutate(race = fct\_recode(race,  
"White" = "1",  
"Otherwise" = "2"  
))  
  
#converting states  
parole = parole %>% mutate(state = as\_factor(as.character(state))) %>%  
mutate(state = fct\_recode(state,  
"Any Other State" = "1",  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4"  
))  
  
#converting Crimes  
parole = parole %>% mutate(crime = as\_factor(as.character(crime))) %>%  
mutate(crime = fct\_recode(crime,  
"Any Other Crime" = "1",  
"Larceny" = "2",  
"Drug-related crime" = "3",  
"Driving-related crime" = "4"  
))  
  
#converting Multiple Offenses  
parole = parole %>% mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"Otherwise" = "0",  
"Incarcerated for multiple offenses" = "1"  
))  
  
#converting parole  
parole = parole %>% mutate(violator = as\_factor(as.character(violator))) %>%  
mutate(violator = fct\_recode(violator,  
"Completed the parole without violation" = "0",  
"Violated the parole" = "1"  
))

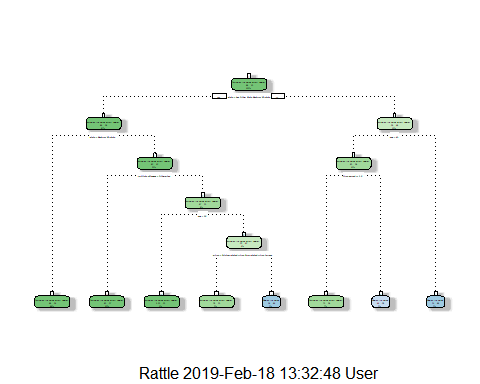
For this assignment, we’ll start by splitting the data into training and testing, using a random number seed of 12345.

set.seed(12345) #sets random number seed for cross validation  
train.rows = createDataPartition(y = parole$violator, p=0.7, list= FALSE)  
train = parole[train.rows,]  
test = parole[-train.rows,]

### Creating Classification Trees

First we will build a classification tree based on parole violator.

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



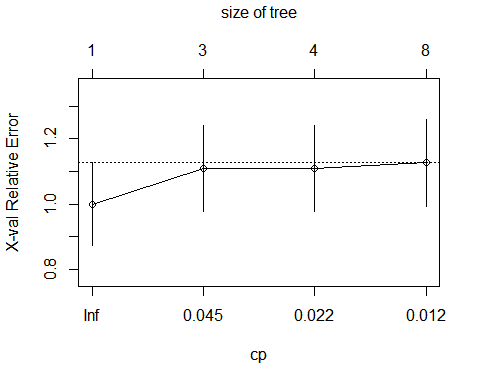
Now that we’ve built the tree, we will use it to classify a 40 year-old parolee from Louisiana who served a 5 year prison sentence. To do this we begin at the top of the tree. Click the Zoom button to get a better look. First we look at the State variable, as this parolee is from Louisiana we go to the right of the first category. The second step shows us Completed Parolee Without Violation and look at the age. As this indidivual is younger than 43 we go to the left to the next variable to look at time served. As their sentence is greater than or equal to 2.5 years, we can classify this individual as “Completed Parole Without Violation.”

Next, we’ll evaluate tree performance as a function of the complexity parameter (cp)

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)



As we can see from our complexity parameters, using a cp of 0.036364 would be accpetable as it does not appear to overfit the data in the same way that a cp of 0.010000 would. We would then use this value to prune the tree back.

Next, we’ll Prune the tree (at minimum cross-validated error of 0.036364) to create a “root”; However, this time we will not plot the tree.

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

Based on this model, we can see that Completed the parole without violation is our majority class.

### Predictions from the Tree

For this next section, we will be creating predictions on training set and taking a look at it’s confusion matrix.

treepred = predict(tree2, train, type = "class")  
head(treepred)

## [1] Completed the parole without violation  
## [2] Completed the parole without violation  
## [3] Completed the parole without violation  
## [4] Completed the parole without violation  
## [5] Completed the parole without violation  
## [6] Completed the parole without violation  
## Levels: Completed the parole without violation Violated the parole

confusionMatrix(treepred,train$violator, positive = "Completed the parole without violation") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed the parole without violation  
## Completed the parole without violation 418  
## Violated the parole 0  
## Reference  
## Prediction Violated the parole  
## Completed the parole without violation 55  
## Violated the parole 0  
##   
## Accuracy : 0.8837   
## 95% CI : (0.8513, 0.9112)   
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.5358   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : 3.305e-13   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8837   
## Neg Pred Value : NaN   
## Prevalence : 0.8837   
## Detection Rate : 0.8837   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Completed the parole without violation  
##

Accoringing to this matrix we are 88.37% accurate with the unpruned tree. It is also important to note that there is no significant difference in the improvement in accuracy. Our naive accuracy is roughly 88% meaning that everyone in the dataset falls within the majority class.

The next step is to to create predictions on testing set and taking a look at it’s confusion matrix.

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

## [1] Completed the parole without violation  
## [2] Completed the parole without violation  
## [3] Completed the parole without violation  
## [4] Completed the parole without violation  
## [5] Completed the parole without violation  
## [6] Completed the parole without violation  
## Levels: Completed the parole without violation Violated the parole

confusionMatrix(treepred\_test,test$violator, positive = "Completed the parole without violation") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed the parole without violation  
## Completed the parole without violation 179  
## Violated the parole 0  
## Reference  
## Prediction Violated the parole  
## Completed the parole without violation 23  
## Violated the parole 0  
##   
## Accuracy : 0.8861   
## 95% CI : (0.8341, 0.9264)   
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.5553   
##   
## Kappa : 0   
## Mcnemar's Test P-Value : 4.49e-06   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8861   
## Neg Pred Value : NaN   
## Prevalence : 0.8861   
## Detection Rate : 0.8861   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Completed the parole without violation  
##

Accoringing to this matrix we are 88.61% accurate with the unpruned tree. It is also important to note that there is no significant difference in the improvement in accuracy. This matrix also reports a sensitivity of 1.0000 which indicates to me that this is model is fairly good at correctly classifying the parolees; However, it is also reporting a specificity of 0.0000, meaninging there are no true negatives in the data. This might not be the best model then, especially since incorrectly classifying this data can have negative consequences.

### Import in second dataset & Converting data

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()  
## )

#converting data  
blood = blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%  
mutate(DonatedMarch = fct\_recode(DonatedMarch,  
"No" = "0",  
"Yes" = "1"  
))

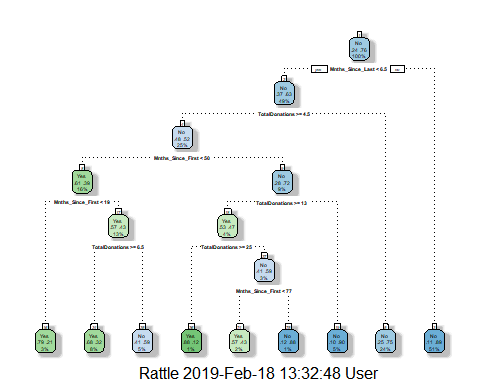
We’ll start by splitting the data into training and testing, using a random number seed of 1234.

set.seed(1234) #sets random number seed for cross validation  
train.rows = createDataPartition(y = blood$DonatedMarch, p=0.7, list= FALSE)  
train2 = blood[train.rows,]  
test2 = blood[-train.rows,]

### Creating Classification Trees

Next, we will build a classification tree based on parole violator.

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

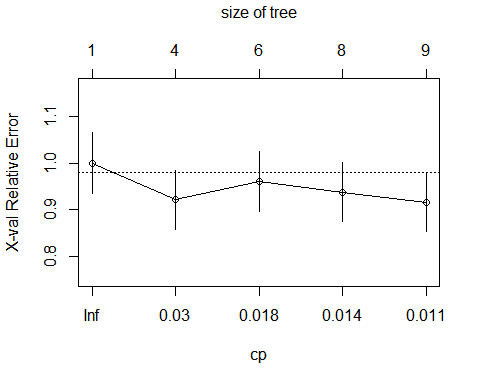


Next, we’ll evaluate tree performance as a function of the complexity parameter (cp)

printcp(tree3)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = blood, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] Mnths\_Since\_First Mnths\_Since\_Last TotalDonations   
##   
## Root node error: 178/748 = 0.23797  
##   
## n= 748   
##   
## CP nsplit rel error xerror xstd  
## 1 0.046816 0 1.00000 1.00000 0.065430  
## 2 0.019663 3 0.85955 0.92135 0.063571  
## 3 0.016854 5 0.82022 0.96067 0.064523  
## 4 0.011236 7 0.78652 0.93820 0.063985  
## 5 0.010000 8 0.77528 0.91573 0.063431

plotcp(tree3)



As we can see from our complexity parameters, using a cp of 0.019663 would be accpetable as it does not appear to overfit the data.

Prune the tree (at minimum cross-validated error)

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

### Predictions from the Tree

Predictions on training set

bloodpred = predict(tree3, train2, type = "class")  
head(bloodpred)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(bloodpred,train2$DonatedMarch, positive = "Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 53 19  
## No 72 380  
##   
## Accuracy : 0.8263   
## 95% CI : (0.7911, 0.8578)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.0001954   
##   
## Kappa : 0.4405   
## Mcnemar's Test P-Value : 5.006e-08   
##   
## Sensitivity : 0.4240   
## Specificity : 0.9524   
## Pos Pred Value : 0.7361   
## Neg Pred Value : 0.8407   
## Prevalence : 0.2385   
## Detection Rate : 0.1011   
## Detection Prevalence : 0.1374   
## Balanced Accuracy : 0.6882   
##   
## 'Positive' Class : Yes   
##

Accoringing to this matrix we are 83% accurate with the unpruned tree. It is also important to note that there is is a significant difference in the improvement in accuracy [P-Value [Acc > NIR] : 0.0001954]. We also have a naive accuracy of 76%, predicting that most donators will fall into the majority class.

Predictions on testing set

bloodpred\_test = predict(tree3, newdata=test2, type = "class")  
head(bloodpred\_test)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(bloodpred\_test,test2$DonatedMarch, positive = "Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 17 11  
## No 36 160  
##   
## Accuracy : 0.7902   
## 95% CI : (0.7309, 0.8416)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.1945593   
##   
## Kappa : 0.3063   
## Mcnemar's Test P-Value : 0.0004639   
##   
## Sensitivity : 0.32075   
## Specificity : 0.93567   
## Pos Pred Value : 0.60714   
## Neg Pred Value : 0.81633   
## Prevalence : 0.23661   
## Detection Rate : 0.07589   
## Detection Prevalence : 0.12500   
## Balanced Accuracy : 0.62821   
##   
## 'Positive' Class : Yes   
##

Accoringing to this matrix we are 79% accurate with the unpruned tree. It is also important to note that there is no significant difference in the improvement in accuracy [P-Value [Acc > NIR] : 0.1945593]. Our naive accuracy is 76%, predicting that most donators will fall into the majority class. This model has a sensitivity of 0.32075 so there are more false postives than true positives, but high specificity of 0.93567, showing there are fewer false negatives than true negatives. This might not be the best quality model to use to predict who will donate blood.