## Module 4 Assignment 1

#install.packages("rpart")  
#install.packages("rattle")  
#install.packages("RColorBrewer")

library(tidyverse)

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

## v ggplot2 3.1.0 v purrr 0.3.0  
## v tibble 2.0.1 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(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)

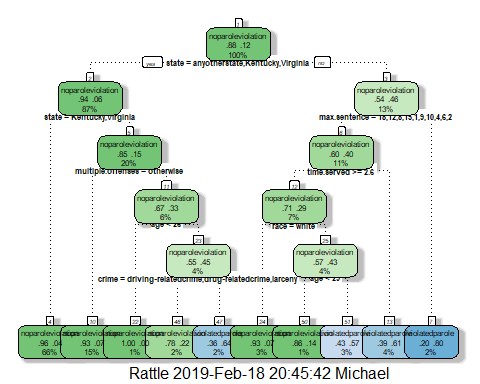
## Classes 'spec\_tbl\_df', 'tbl\_df', 'tbl' and 'data.frame': 675 obs. of 9 variables:  
## $ male : num 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : num 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : num 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served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

parole = parole %>% mutate(male = as\_factor(as.character(male))) %>%  
mutate(male = fct\_recode(male,  
"female" = "0",  
"male" = "1"))%>%  
   
 mutate(age = as.numeric(as.character(age)))%>%  
   
mutate(time.served = as.numeric(as.character(time.served)))%>%   
   
 mutate(max.sentence = as\_factor(as.character(max.sentence)))%>%   
   
 mutate(race = as\_factor(as.character(race)))%>%  
   
mutate(race = fct\_recode(race,  
"white" = "1",  
"otherwise" = "2"))%>%   
   
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses)))%>%  
   
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"otherwise" = "0",  
"multipleoffenses" = "1"))%>%  
   
 mutate(state = as\_factor(as.character(state))) %>%  
   
mutate(state = fct\_recode(state,  
"anyotherstate" = "1",  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4"))%>%  
   
 mutate(crime = as\_factor(as.character(crime))) %>%  
   
   
mutate(crime = fct\_recode(crime,  
"othercrime" = "1",  
"larceny" = "2",  
"drug-relatedcrime" = "3",  
"driving-relatedcrime" = "4"))%>%   
   
 mutate(violator = as\_factor(as.character(violator)))%>%  
   
mutate(violator = fct\_recode(violator,  
"violatedparole" = "1",  
"noparoleviolation" = "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 "anyotherstate",..: 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 : Factor w/ 17 levels "18","12","13",..: 1 2 2 1 2 1 1 2 3 2 ...  
## $ multiple.offenses: Factor w/ 2 levels "otherwise","multipleoffenses": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "driving-relatedcrime",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "noparoleviolation",..: 1 1 1 1 1 1 1 1 1 1 ...

set.seed(12345)  
train.rows = createDataPartition(y = parole$violator, p=0.7, list = FALSE) #70% in training  
train = parole[train.rows,]   
test = parole[-train.rows,]

tree1 = rpart(violator ~., train, method="class")  
fancyRpartPlot(tree1, cex=.5)

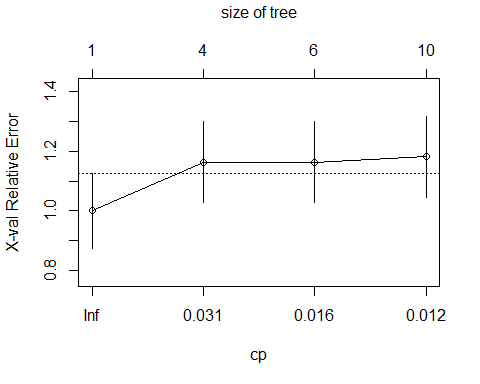


Based on the classification tree above we see that if you are from Louisiana you do not even have to go to the level of information of the age of the parolee. You see that with Louisiana you go to the right on the tree. Then when it asks the max sentence 5 years is not equal to any of the listed numbers so we go down the right side (“<> not equal to” side) and it states that yes that particular individual described did violate parole. We land at the bottom segment that is all the way to the right.

printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age crime max.sentence multiple.offenses  
## [5] 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 0.018182 3 0.81818 1.1636 0.13526  
## 3 0.013636 5 0.78182 1.1636 0.13526  
## 4 0.010000 9 0.72727 1.1818 0.13614

plotcp(tree1)



Based on the lecture what we are looking for is a CP value of .01. But we see with this classification tree that the better CP value is .0545. By using this CP value we see the lowest relative error.

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

The majority class for the training data set is “no parole violation”.

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

## 1 2 3 4   
## noparoleviolation noparoleviolation noparoleviolation noparoleviolation   
## 5 6   
## noparoleviolation noparoleviolation   
## Levels: noparoleviolation violatedparole

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction noparoleviolation violatedparole  
## noparoleviolation 399 21  
## violatedparole 19 34  
##   
## Accuracy : 0.9154   
## 95% CI : (0.8866, 0.9389)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.01576   
##   
## Kappa : 0.5819   
## Mcnemar's Test P-Value : 0.87437   
##   
## Sensitivity : 0.61818   
## Specificity : 0.95455   
## Pos Pred Value : 0.64151   
## Neg Pred Value : 0.95000   
## Prevalence : 0.11628   
## Detection Rate : 0.07188   
## Detection Prevalence : 0.11205   
## Balanced Accuracy : 0.78636   
##   
## 'Positive' Class : violatedparole   
##

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

## 1 2 3 4   
## noparoleviolation violatedparole noparoleviolation noparoleviolation   
## 5 6   
## noparoleviolation noparoleviolation   
## Levels: noparoleviolation violatedparole

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction noparoleviolation violatedparole  
## noparoleviolation 169 17  
## violatedparole 10 6  
##   
## Accuracy : 0.8663   
## 95% CI : (0.8115, 0.91)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.8409   
##   
## Kappa : 0.2363   
## Mcnemar's Test P-Value : 0.2482   
##   
## Sensitivity : 0.26087   
## Specificity : 0.94413   
## Pos Pred Value : 0.37500   
## Neg Pred Value : 0.90860   
## Prevalence : 0.11386   
## Detection Rate : 0.02970   
## Detection Prevalence : 0.07921   
## Balanced Accuracy : 0.60250   
##   
## 'Positive' Class : violatedparole  
##

This is not a very good model. One could argue that the accuracy is high but the fact that it is lower than the naive model suggests you are better off just assuming the variable of the majority class. Also, the p-value is very high so there is not a lot of significance in relation to the 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)  
str(blood)

## Classes 'spec\_tbl\_df', '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 : num 1 1 1 1 0 0 1 0 1 1 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Mnths\_Since\_Last = col\_double(),  
## .. TotalDonations = col\_double(),  
## .. Total\_Donated = col\_double(),  
## .. Mnths\_Since\_First = col\_double(),  
## .. DonatedMarch = col\_double()  
## .. )

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

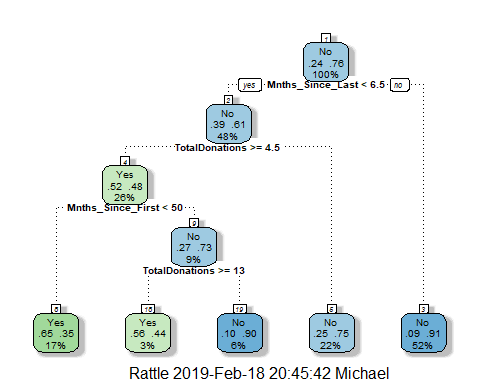
## Classes 'tbl\_df', 'tbl' and 'data.frame': 748 obs. of 5 variables:  
## $ Mnths\_Since\_Last : num 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num 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   
##   
##   
##   
##

set.seed(1234)  
train.rows2 = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train2 = blood[train.rows2,]   
test2 = blood[-train.rows2,]

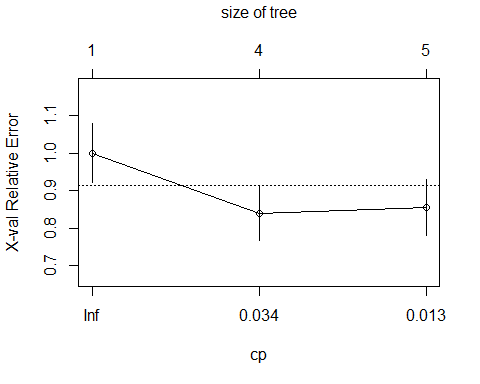
tree.blood = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree.blood, cex=.6)



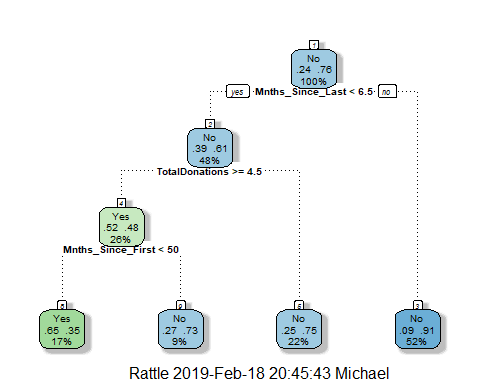
printcp(tree.blood)

##   
## 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(tree.blood)



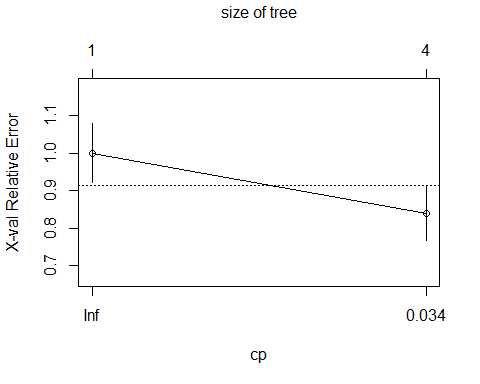
tree.blood2 = prune(tree.blood,cp= tree.blood$cptable[which.min(tree.blood$cptable[,"xerror"]),"CP"])  
fancyRpartPlot(tree.blood2, cex=.6)



printcp(tree.blood2)

##   
## 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 1.00 0.078049  
## 2 0.016 3 0.784 0.84 0.073304

plotcp(tree.blood2)



treepred.bloodtrain = predict(tree.blood2, train2, type = "class")  
head(treepred.bloodtrain)

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

confusionMatrix(treepred.bloodtrain,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   
##

treepred.bloodtest = predict(tree.blood2, test2, type = "class")  
head(treepred.bloodtest)

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

confusionMatrix(treepred.bloodtest,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   
##

We see that with the training data our classification tree has greater accuracy than if we were to just go with the majority class. It is actually the opposite for the test data set though. Of course this may have something to do with the randomize data that was grouped into each data set. Of course the P-value on each Confusion Matrix is wildly different. The training data set implies there is a significant difference while the testing data set implies there isn’t a whole lot of difference.