## Week4 Assignment 2

### DEY, Sankha

#### Classification Trees

# message=FALSE done before last knit  
library(tidyverse)  
library(caret)  
library(rpart)  
library(rattle)  
library(RColorBrewer)

parole2 <- 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()  
## )

parole2 = parole2 %>% mutate(male = as\_factor(as.character(male))) %>% #Convert Male  
mutate(male = fct\_recode(male,  
"Male" = "1",  
"Female" = "0")) %>%  
 mutate(race = as\_factor(as.character(race))) %>% #Convert Race  
mutate(race = fct\_recode(race,  
"White" = "1",  
"Others" = "2")) %>%  
 mutate(state = as\_factor(as.character(state))) %>% #Convert State  
mutate(state = fct\_recode(state,  
"Kentucky" = "2",  
"Louisiana" = "3",  
"Virginia" = "4",  
"Others" = "1")) %>%  
 mutate(crime = as\_factor(as.character(crime))) %>% #Convert Crime  
mutate(crime = fct\_recode(crime,  
"Larceny" = "2",  
"Drug" = "3",  
"Driving" = "4",  
"Others" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>% #Convert Offences  
mutate(multiple.offenses = fct\_recode(multiple.offenses,  
"Multiple" = "1",  
"Others" = "0")) %>%  
 mutate(violator = as\_factor(as.character(violator))) %>% #Convert Violators  
mutate(violator = fct\_recode(violator,  
"Violation" = "1",  
"No Violation" = "0"))  
str(parole2)

## Classes 'spec\_tbl\_df', '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","Others": 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 "Others","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 "Others","Multiple": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "Driving","Drug",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "No Violation",..: 1 1 1 1 1 1 1 1 1 1 ...

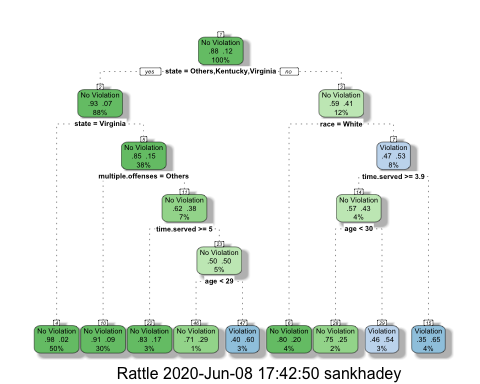
#### Task 1

set.seed(12345)   
trainrows42 = createDataPartition(y = parole2$violator, p=0.7, list = FALSE) #70% in training  
train42 = slice(parole2,trainrows42)  
test42 = slice(parole2,-trainrows42)

Train set has 473 observations and Test set has 202 observations.

#### Task 2

tree1 = rpart(violator ~., train42, method="class")  
fancyRpartPlot(tree1)



#### Task 3

How would you classify a 40 year-old parolee from Louisiana who served a 5 year prison sentence? Describe how you “walk through” the classification tree to arrive at your answer.  
**Answer**: In the classification tree, we will start with the root, which checks if the state = Others or Kentucky or Virginia. Since the state of the parole taken in the example is Louisiana, we will take the right branch (no, in the condition). Then the next node is to check if the race is white. In our example there is no race mentioned. If we assume that the parolee is white, then we can conclude that there is no violation (left side of race=white node). If the parolee is non-white, then we will go to the right hand side and check the next node (Time Served >=3.9 years). Our parolee served 5 years, so will go to left (yes condition) and hit the next node age (<30). Our parolee is 40 years old, so we will go to right (no condition). We will end up having violation.

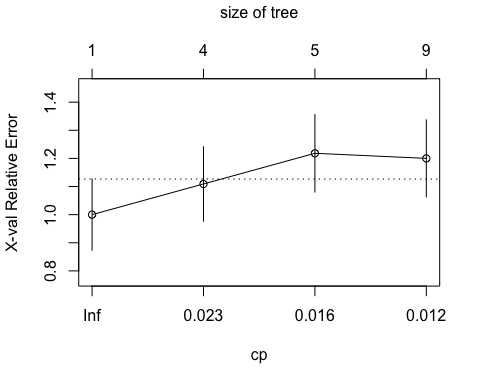
So, if the said parolee is white, then we can classify his/her as Non-Violators. If the parolee is non-white then we can classify him/her as Violator.

#### Task 4

printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train42, method = "class")  
##   
## Variables actually used in tree construction:  
## [1] age multiple.offenses race state   
## [5] time.served   
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.030303 0 1.00000 1.0000 0.12676  
## 2 0.018182 3 0.90909 1.1091 0.13253  
## 3 0.013636 4 0.89091 1.2182 0.13788  
## 4 0.010000 8 0.83636 1.2000 0.13702

plotcp(tree1)



Lowest Xerror value is 1.00 and the corresponding CP value is 0.0303. So, the optimal value of CP is 0.0303.

#### Task 5

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

treepred = predict(tree2, train42, type = "class")  
head(treepred)

## [1] No Violation No Violation No Violation No Violation No Violation  
## [6] No Violation  
## Levels: No Violation Violation

confusionMatrix(treepred,train42$violator,positive="Violation") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violation  
## No Violation 418 55  
## Violation 0 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 : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8837   
## Prevalence : 0.1163   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Violation   
##

NIR and Accuracy are having same value. This happened because per this model there are no predictions of violations. Hence the majority class predictions are same with the model accuracy.  
Majority of the observations are “Non Violation”s in training set.

#### Task 6

treepred1 = predict(tree1, train42, type = "class")  
head(treepred1)

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

confusionMatrix(treepred1,train42$violator,positive="Violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violation  
## No Violation 400 28  
## Violation 18 27  
##   
## Accuracy : 0.9027   
## 95% CI : (0.8724, 0.9279)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.1095   
##   
## Kappa : 0.4862   
##   
## Mcnemar's Test P-Value : 0.1845   
##   
## Sensitivity : 0.49091   
## Specificity : 0.95694   
## Pos Pred Value : 0.60000   
## Neg Pred Value : 0.93458   
## Prevalence : 0.11628   
## Detection Rate : 0.05708   
## Detection Prevalence : 0.09514   
## Balanced Accuracy : 0.72392   
##   
## 'Positive' Class : Violation   
##

Accuracy : 0.9027 is better than the No Information Rate 0.8837. This model looks better than the Naive model.

Sensitivity: 0.49091 Specificity: 0.95694

#### Task 7

treepred\_test = predict(tree1, newdata=test42, type = "class")  
head(treepred\_test)

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

confusionMatrix(treepred\_test,test42$violator,positive="Violation")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Violation Violation  
## No Violation 171 13  
## Violation 8 10  
##   
## Accuracy : 0.896   
## 95% CI : (0.8455, 0.9345)  
## No Information Rate : 0.8861   
## P-Value [Acc > NIR] : 0.3797   
##   
## Kappa : 0.4309   
##   
## Mcnemar's Test P-Value : 0.3827   
##   
## Sensitivity : 0.43478   
## Specificity : 0.95531   
## Pos Pred Value : 0.55556   
## Neg Pred Value : 0.92935   
## Prevalence : 0.11386   
## Detection Rate : 0.04950   
## Detection Prevalence : 0.08911   
## Balanced Accuracy : 0.69504   
##   
## 'Positive' Class : Violation   
##

Accuracy : 0.896,Sensitivity : 0.43478, Specificity : 0.95531  
Accuracy in Test data is very close to the accuracy of Train data though there is a slight drop at test. But still the Test accuracy (0.896) is better than the Navive (NIR) model which is 0.8861. There is no overfitting of data. Model looks good on both Training and Test data set.

#### Task 8

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()  
## )

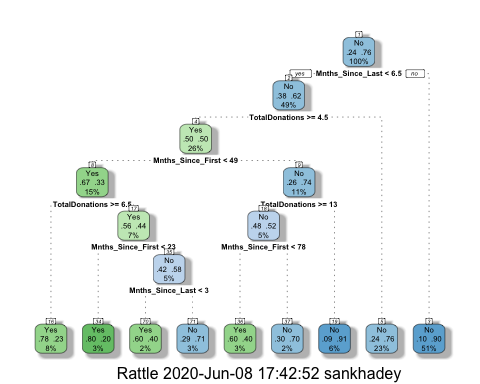
Blood = Blood %>% mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%   
mutate(DonatedMarch = fct\_recode(DonatedMarch,  
"No" = "0",  
"Yes" = "1"))  
str(Blood)

## 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 : Factor w/ 2 levels "Yes","No": 1 1 1 1 2 2 1 2 1 1 ...

#### Task 9

set.seed(1234)   
trainrows42b = createDataPartition(y = Blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train42b = slice(Blood,trainrows42b)  
test42b = slice(Blood,-trainrows42b)

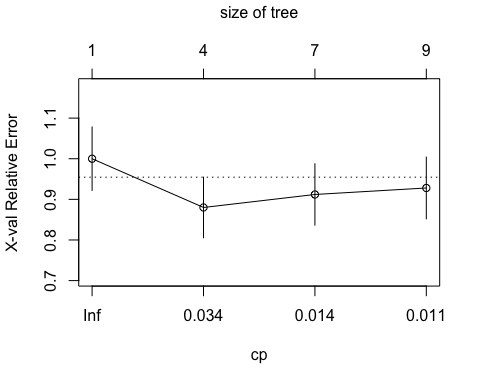
tree42b = rpart(DonatedMarch ~., train42b, method="class")  
fancyRpartPlot(tree42b)



printcp(tree42b)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train42b, 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 0.880 0.074580  
## 3 0.012 6 0.736 0.912 0.075556  
## 4 0.010 8 0.712 0.928 0.076030

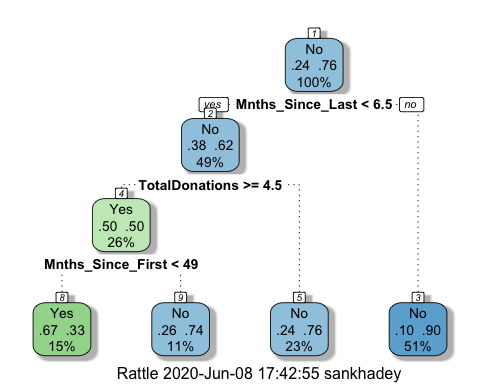
plotcp(tree42b)



Lowest Xerror value is 0.88 and the corresponding CP value is 0.016. So, the optimal value of CP is 0.016.

#### Task 10

tree42c = prune(tree42b,cp= tree42b$cptable[which.min(tree42b$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(tree42c)



Predictions on training set

treepred42b = predict(tree42c, train42b, type = "class")  
head(treepred42b)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred42b,train42b$DonatedMarch,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 53 26  
## No 72 373  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4107   
##   
## Mcnemar's Test P-Value : 5.476e-06   
##   
## Sensitivity : 0.4240   
## Specificity : 0.9348   
## Pos Pred Value : 0.6709   
## Neg Pred Value : 0.8382   
## Prevalence : 0.2385   
## Detection Rate : 0.1011   
## Detection Prevalence : 0.1508   
## Balanced Accuracy : 0.6794   
##   
## 'Positive' Class : Yes   
##

On Training set, Accuracy is 0.813, Sensitivity is 0.4240 and Specificity : 0.9348  
Naive Model value is 0.7615. So accuracy of our model is better and p-value is small enough. Overal the model is acceptable on training data.

Predictions on testing set

treepred\_test42b = predict(tree42c, newdata=test42b, type = "class")  
head(treepred\_test42b)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_test42b,test42b$DonatedMarch,positive="Yes") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 18 20  
## No 35 151  
##   
## Accuracy : 0.7545   
## 95% CI : (0.6927, 0.8094)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.65710   
##   
## Kappa : 0.2468   
##   
## Mcnemar's Test P-Value : 0.05906   
##   
## Sensitivity : 0.33962   
## Specificity : 0.88304   
## Pos Pred Value : 0.47368   
## Neg Pred Value : 0.81183   
## Prevalence : 0.23661   
## Detection Rate : 0.08036   
## Detection Prevalence : 0.16964   
## Balanced Accuracy : 0.61133   
##   
## 'Positive' Class : Yes   
##

On Testing set, Accuracy is 0.7545, Sensitivity is 0.33962 and Specificity : 0.88304  
Naive Model value is 0.7634. So accuracy of our model is not better than Naive model. Hence, the model doesn’t perform well on the test data. A higher p-value (0.657) supports our conclusion. So, this model doesn’t have good accuracy on test data set or even real world data. A Naive can do better. It also seems that the data have been overfitted at training set.