### Module 4, Assignment 2 - Classification Trees

### Hollye Moss

### Due 08 June 2020

**Task 1** 675 observations in the original data set split into 473 observations in the training set and 202 in the testing set.

**Task 2** The classification tree and plot are below.

**Task 3** A 40 year old from Louisiana who served 5 years in prison would be classified as a non-violator if he is white or violator if he is not white.  
He is not from “state=other, Kentucky, Virginia” so we go to the right. If I assume is he is not white, I stay to the right. His time served is >=3.9. Staying to the right, I see that he will be classified as a violator. If he is white, he is classified as a non-violator.

**Task 4** At Cp=.030303 we see the lowest cross-validated error (1.0000). This is actually without a split.

**Task 5** At the root, we would simply classify people as non-violators. This is a naive model that puts everyone into the majority class.

**Task 6** Using the unpruned tree on the training data set:

Accuracy = 0.9027 (compared to a naive model accuracy of 0.8837)

Sensitivity = 0.9569  
Specificity = 0.4909

**Task 7** Using the unpruned tree on the testing data set we see similar numbers:

Accuracy = 0.896 (compared to a naive model accuracy of 0.8861)

Sensitivity = 0.9553  
Specificity = 0.4348

Accuracy is a measure of the number correctly predicted. In this model, we’re just barely improving accuracy (versus the naive model) by using the classification Tree. The p-values tell us the model is not significant.

Sensitivity is a measure of how good the model is at detecting positives. In this example, positives are defined as “non-violators”.

Specificity is a measure of how good the model is at avoiding false positives–in this case, saying someone will be a non-violator when they actually violated parole. In this model, we’re incorrectly detecting/predicting those who will violate just under 50% of the time.

**Task 8** See code below.

**Task 9** The 748 observations in the blood data set were split into 524 for the training set and 224 for the testing set.

The software suggests a classification tree with 8 splits before hitting the Cp threshold of 0.01. The Cp is actually minimized at 3 splits (Cp = 0.016 and cross-validation error = 0.88).

**Task 10** On the training data, the model has the following: Accuracy = 0.813 (compared to a naive model accuracy of 0.7615) Sensitivity = 0.4240  
Specificity = 0.9348

On the testing data, the model has the following: Accuracy = 0.7545 (compared to a naive model accuracy of 0.7634) Sensitivity = 0.33962  
Specificity = 0.88304

That naive models with the two data sets had the same accuracy tells me that I got an even split on the *DonatedMarch* variable. The training set had better accuracy. The model on the testing set was not as accurate as the naive model–the model actually hurt the accuracy.

Sensitivity and specificity were lower with the testing set, but in the same ball park.

Overall, this model does not do a good job of predicting who will donate, but it does a better job of avoiding false positives. If you are seeking donors, it might not tell you who will donate, but it could suggest where you should not spend your resources asking.

## PAROLE DATA SET

library(tidyverse)

library(caret)

library(rpart)  
library(rattle)

library(RColorBrewer)  
parole <- read\_csv("parole.csv")

View(parole)

### Convert the Data

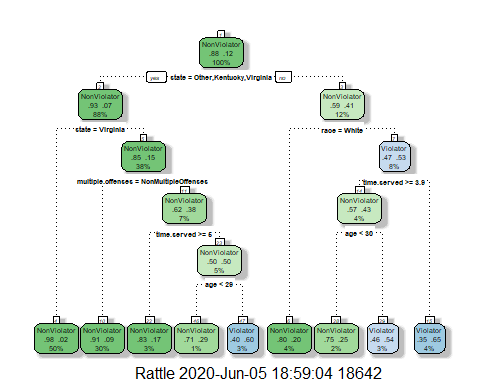
parole <- parole %>%  
# convert the male variable from a number to a character then to a factor  
 mutate(male = as\_factor(as.character(male))) %>%  
# rename the male codes currently 0, 1   
 mutate(male = fct\_recode(male,  
 "Female" = "0",  
 "Male" = "1")) %>%  
   
# convert the race variable from a number to a character then to a factor  
 mutate(race = as\_factor(as.character(race))) %>%  
# rename the race codes currently 1, 2   
 mutate(race = fct\_recode(race,  
 "White" = "1",  
 "NonWhite" = "2")) %>%  
   
# convert the state variable from a number to a character then to a factor  
 mutate(state = as\_factor(as.character(state))) %>%  
# rename the state codes currently varies by state. These three states were chosen because they have high representation in the data set.  
 mutate(state = fct\_recode(state,  
 "Other" = "1",  
 "Kentucky" = "2",  
 "Louisiana" = "3",  
 "Virginia" = "4")) %>%  
  
# convert the multiple.offenses variable from a number to a character then to a factor  
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
# rename the multiple.offenses codes currently 0, 1   
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
 "MultipleOffenses" = "1",  
 "NonMultipleOffenses" = "0")) %>%  
  
# convert the crime variable from a number to a character then to a factor  
 mutate(crime = as\_factor(as.character(crime))) %>%  
# rename the crime codes currently 1, 2, 3, 4.   
 mutate(crime = fct\_recode(crime,  
 "OtherCrime" = "1",  
 "Larceny" = "2",  
 "DrugRelated" = "3",  
 "DrivingRelated" = "4")) %>%  
   
 # convert the violator variable from a number to a character then to a factor  
 mutate(violator = as\_factor(as.character(violator))) %>%  
# rename the multiple.offenses codes currently 0, 1   
 mutate(violator = fct\_recode(violator,  
 "Violator" = "1",  
 "NonViolator" = "0"))   
  
# View(parole)

### Split the Data

# seed the random number generator for repeatability  
set.seed(12345)  
# split the data 70% to training set  
train.rows <- createDataPartition(y = parole$violator, p=0.7, list = FALSE)  
train <- slice(parole, train.rows)  
test <- slice(parole, -train.rows)

### Create a Classification Tree

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

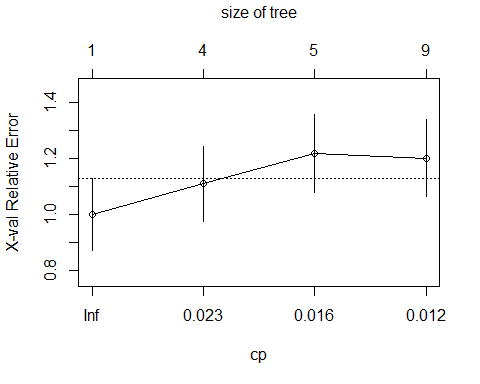


### Complexity Paramater, Cp, and Prune to Min Cp

# output Cp and cross validated error values  
printcp(tree1)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, 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

# visualize the tree  
plotcp(tree1)



# prune the first tree to the minimum Cp branch  
tree2 = prune(tree1,cp=tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
printcp(tree2)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = train, method = "class")  
##   
## Variables actually used in tree construction:  
## character(0)  
##   
## Root node error: 55/473 = 0.11628  
##   
## n= 473   
##   
## CP nsplit rel error xerror xstd  
## 1 0.030303 0 1 1 0.12676

plotcp(tree2)

### Predictions Based on Training Set & Confusion Matrix

# predictions based on training data set and UNpruned tree  
treepred <- predict(tree1, train, type = "class")  
# look at first six lines and comment out before printing  
# head(treepred)  
# confusion matrix  
confusionMatrix(treepred,train$violator,positive="NonViolator")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolator Violator  
## NonViolator 400 28  
## Violator 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 : 0.9569   
## Specificity : 0.4909   
## Pos Pred Value : 0.9346   
## Neg Pred Value : 0.6000   
## Prevalence : 0.8837   
## Detection Rate : 0.8457   
## Detection Prevalence : 0.9049   
## Balanced Accuracy : 0.7239   
##   
## 'Positive' Class : NonViolator   
##

### Predictions Based on Testing Set & Confusion Matrix

# predictions based on testing data set and UNpruned tree  
treepred <- predict(tree1, test, type = "class")  
# look at first six lines and comment out before printing  
# head(treepred)  
# confusion matrix  
confusionMatrix(treepred,test$violator,positive="NonViolator")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NonViolator Violator  
## NonViolator 171 13  
## Violator 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 : 0.9553   
## Specificity : 0.4348   
## Pos Pred Value : 0.9293   
## Neg Pred Value : 0.5556   
## Prevalence : 0.8861   
## Detection Rate : 0.8465   
## Detection Prevalence : 0.9109   
## Balanced Accuracy : 0.6950   
##   
## 'Positive' Class : NonViolator   
##

## PAROLE DATA SET

### Read and Convert to Factors

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()  
## )

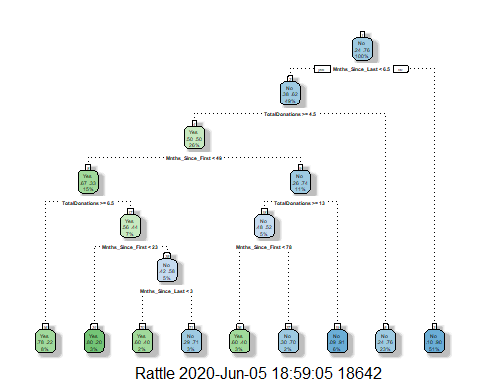
View(blood)  
blood <- blood %>%  
# convert the DonatedMarch variable from a number to a character then to a factor  
 mutate(DonatedMarch = as\_factor(as.character(DonatedMarch))) %>%  
# rename the DonatedMarch codes currently 0, 1   
 mutate(DonatedMarch = fct\_recode(DonatedMarch,  
 "No" = "0",  
 "Yes" = "1"))

### Split the Blood Data

# seed the random number generator for repeatability  
set.seed(1234)  
# split the data 70% to training set  
train.rows <- createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE)  
train <- slice(blood, train.rows)  
test <- slice(blood, -train.rows)

### Build Classification Tree and View Cp

# build the tree  
tree1 <- rpart(DonatedMarch ~., train, method="class")  
# view the tree   
fancyRpartPlot(tree1)



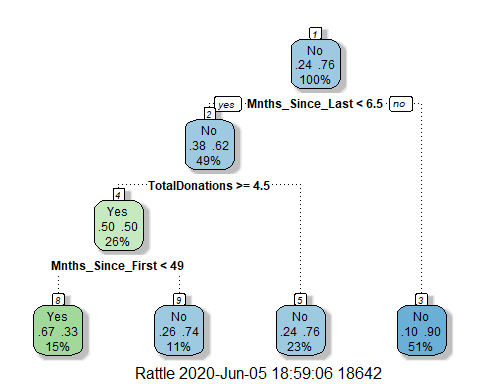
# output Cp and cross validated error values  
printcp(tree1)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = train, 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

# visualize the Cp  
# plotcp(tree1)

### Prune blood Classifiation Tree

# prune the tree at the min Cp value above.  
tree2 <- prune(tree1,cp=tree1$cptable[which.min(tree1$cptable[,"xerror"]),"CP"])  
# view the tree   
fancyRpartPlot(tree2)



### Use Pruned Tree to Predict with Train and Test Sets

# Predictions on training set  
treepred <- predict(tree2, train, type = "class")  
head(treepred)

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

# confustion matrix on training set  
confusionMatrix(treepred,train$DonatedMarch,positive="Yes")

## 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   
##

# Predictions on testing set  
treepred <- predict(tree2, test, type = "class")  
head(treepred)

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

# confustion matrix on training set  
confusionMatrix(treepred,test$DonatedMarch,positive="Yes")

## 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   
##