# Peter Rieth

## BAN 502, Module 4

### Assignment: Classification Tree Assignment

**Load Libraries**

Will use the following libraries: tidyverse, caret, rpart, rattle, and RColorBrewer.

library("tidyverse")  
library("caret")  
library("rpart")  
library("rattle")  
library("RColorBrewer")

Before beginning the assignment tasks, you should read-in the data for the assignment into a data frame called parole. Carefully convert the male, race, state, crime, multiple.offenses, and violator variables to factors. Recode (rename) the factor levels of each of these variables according to the description of the variables provided in the ParoleData.txt file (located with the assignment on Canvas). Note: You did this in a previous assignment. I would encourage you to re-use your code.

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()  
## )

parole = parole %>%   
 mutate(male = as\_factor(as.character(male))) %>% #convert male to factor  
 mutate(male = fct\_recode(male,  
 "male" = "1",  
 "female" = "0")) %>% #rename male factor levels   
 mutate(race = as\_factor(as.character(race))) %>% #convert race to factor  
 mutate(race = fct\_recode(race,  
 "white" = "1",  
 "NOTwhite" = "2")) %>% #rename race factor levels   
 mutate(state = as\_factor(as.character(state))) %>% #convert state to factor  
 mutate(state = fct\_recode(state,  
 "OTHERstate" = "1",  
 "Kentucky" = "2",  
 "Louisiana" = "3",  
 "Virginia" = "4")) %>% #rename state factor levels   
 mutate(crime = as\_factor(as.character(crime))) %>% #convert crime to factor  
 mutate(crime = fct\_recode(crime,  
 "OTHERcrime" = "1",  
 "larceny" = "2",  
 "drug-related" = "3",  
 "driving-related" = "4")) %>% #rename crime factor levels   
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>% #convert multiple.offenses to factor  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
 "incarcerated" = "1",  
 "NOTincarcerated" = "0")) %>% #rename multiple.offenses factor levels   
 mutate(violator = as\_factor(as.character(violator))) %>% #convert violator to factor  
 mutate(violator = fct\_recode(violator,  
 "ViolatedParole" = "1",  
 "CompletedParole" = "0")) #rename violator factor levels   
str(parole)

## 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","NOTwhite": 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 "OTHERstate","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 "NOTincarcerated",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "driving-related",..: 1 2 2 3 3 1 2 3 2 4 ...  
## $ violator : Factor w/ 2 levels "CompletedParole",..: 1 1 1 1 1 1 1 1 1 1 ...

### Task 1:

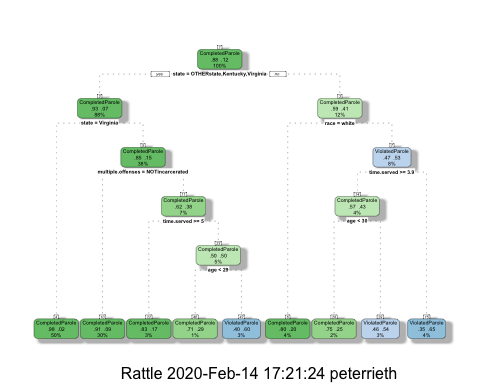
Split the data into training (70%) and testing (30%) sets. Use a random number (set.seed) of 12345.

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

### Task 2:

Create a classification tree using all of the predictor variables to predict “violator” in the training set. Plot the tree.

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



### Task 3:

For the tree created in Task 2, 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.

**40 year-old parolee from Louisiana who served a 5 year prison sentence classification:** Well if they were white I’d classify them as CompletedParole and if they were NOTwhite I’d classify them as ViolatedParole.

**Walk through the tree:**

* Node 1: Answer no and go to right branch to node 2 since Louisiana is not in OTHERState, Kentucky or Virginia
* Node 2: Now I need more information - because the next question is race.
  + If the parolee was white: Take the left branch and be done indicating CompletedParole
  + If the parolee was not white: take the right branch to node 3.
    - Node 3: Answer yes and go down the left branch to node 4 since time.served of 5 years is greater than or equal to 3.9
    - Node 4: Answer no and go down right branch and be done indicating ViolatedParole since 40 years old is not <30.

### Task 4:

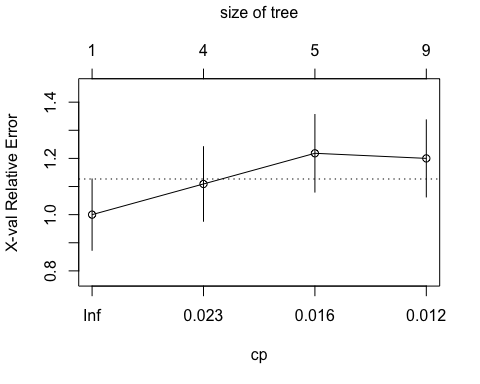
Use the printcp function to evaluate tree performance as a function of the complexity parameter (cp). What cp value should be selected? Note that the printcp table tends to be a more reliable tool than the plot of cp.

**What value of CP should be selected?** 0.030303 has the lowest cross-validated error.

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

plotcp(tree1)



### Task 5:

Prune the tree from Task 2 back to the cp value that you selected in Task 4. **Do not attempt to plot the tree.** You will find that the resulting tree is known as a “root”. A tree that takes the form of a root is essentially a naive model that assumes that the prediction for all observations is the majority class.

**Which class (category) in the training set is the majority class (i.e., has the most observations)?** CompletedParole

Prune the tree (at minimum cross-validated error)

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

### Task 6:

Use the unpruned tree from Task 2 to develop predictions for the training data. Use caret’s confusionMatrix function to calculate the accuracy, specificity, and sensitivity of this tree on the training data. Note that we would not, in practice, use an unpruned tree as such a tree is very likely to overfit on new data.

Predictions on training set

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

## 1 2 3 4   
## CompletedParole CompletedParole CompletedParole CompletedParole   
## 5 6   
## CompletedParole CompletedParole   
## Levels: CompletedParole ViolatedParole

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred,train$violator,positive="ViolatedParole") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 400 28  
## ViolatedParole 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 : ViolatedParole   
##

### Task 7:

Use the unpruned tree from Task 2 to develop predictions for the testing data. Use caret’s confusionMatrix function to calculate the accuracy, specificity, and sensitivity of this tree on the testing data. Comment on the quality of the model.

**Comments on quality of of the model:** the accuracy level of the predictions on the training data is 0.9027 which is slightly better than the accuracy of the naive model with no information rate (i.e., if we had just predicted CompletedParole for everything). The accuracy level of the predictions for the testing data is slighty worse at 0.896, but still better than the naive model. Neither the testing or training predictions improvement in accuracy are statistically significant with both values above .05, but this is a relatively small dataset so perhaps this doesn’t mean the model isn’t usable.

Predictions on testing set

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

## 1 2 3 4   
## CompletedParole CompletedParole CompletedParole CompletedParole   
## 5 6   
## CompletedParole CompletedParole   
## Levels: CompletedParole ViolatedParole

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_test,test$violator,positive="ViolatedParole") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction CompletedParole ViolatedParole  
## CompletedParole 171 13  
## ViolatedParole 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 : ViolatedParole   
##

### Task 8:

Read in the “Blood.csv” dataset. The dataset contains five variables:  
Mnths\_Since\_Last: Months since last donation  
TotalDonations: Total number of donation  
Total\_Donated: Total amount of blood donated  
Mnths\_Since\_First: Months since first donation  
DonatedMarch: Binary variable representing whether he/she donated blood in March (1 = Yes, 0 = No)

Convert the DonatedMarch variable to a factor and recode the variable so 0 = “No” and 1 = “Yes”.

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))) %>% #convert DoneateMarch to factor  
 mutate(DonatedMarch = fct\_recode(DonatedMarch,  
 "Yes" = "1",  
 "No" = "0")) #rename DonatedMarch factor levels

### Task 9:

Split the dataset into training (70%) and testing (30%) sets. **You may wish to name your training and testing sets “train2” and “test2” so as to not confuse them with the parole datsets.** Use set.seed of 1234. Then develop a classification tree on the training set to predict “DonatedMarch”. Evaluate the complexity parameter (cp) selection for this model.

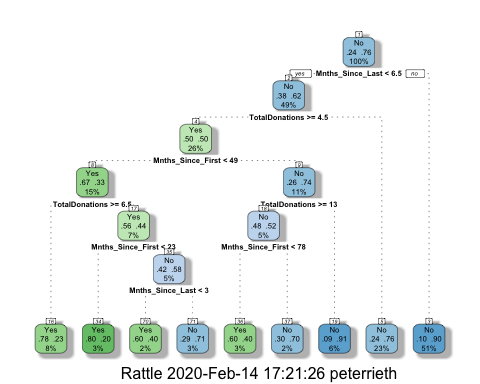
**Evaluate cp:** cp value of 0.016 has the lowest cross-validated error.

Split the data into training (70%) and testing (30%) sets. Use a random number (set.seed) of 1234.

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

Create a classification tree using all of the predictor variables to predict “DonatedMarch” in the training set. Plot the tree.

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

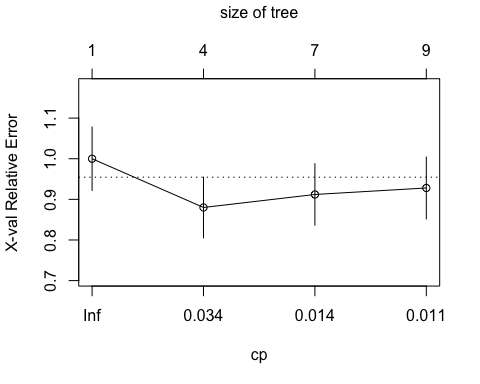


Evaluate the complexity parameter

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 0.880 0.074580  
## 3 0.012 6 0.736 0.912 0.075556  
## 4 0.010 8 0.712 0.928 0.076030

plotcp(tree3)



### Task 10:

Prune the tree back to the optimal cp value, make predictions, and use the confusionMatrix function on the both training and testing sets. Comment on the quality of the predictions.

**Comments on quality of predictions:** the accuracy level of the predictions on the training data is 0.813 which is better than the accuracy of the naive model with no information rate (i.e., if we had just predicted no for everything) of 0.7615 and the p-value of the improvement in accuracy is statistically significant. However, the accuracy level of the predictions on the testing data was 0.7543, which is actually worse than the naive model, indicating this is not a very good predictive model beyond the training data.

Prune the tree (at minimum cross-validated error)

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

Predictions on training set

treepred2 = predict(tree4, train2, type = "class")  
head(treepred2)

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

Caret confusion matrix and accuracy, etc. calcs for training data

confusionMatrix(treepred2,train2$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   
##

Predictions on testing set

treepred2\_test = predict(tree4, newdata=test2, type = "class")  
head(treepred2\_test)

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

Caret confusion matrix and accuracy, etc. calcs for training data

confusionMatrix(treepred2\_test,test2$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   
##