### Module 4 Classification Trees Assignment

## Madison West

Load in libraries

library(tidyverse)

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

## v ggplot2 3.2.1 v purrr 0.3.3  
## v tibble 2.1.3 v dplyr 0.8.3  
## v tidyr 1.0.0 v stringr 1.4.0  
## v readr 1.3.1 v forcats 0.4.0

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

library(caret)

## Warning: package 'caret' was built under R version 3.6.2

## Loading required package: lattice

##   
## Attaching package: 'caret'

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

library(rpart)  
library(rattle)

## Warning: package 'rattle' was built under R version 3.6.2

## Rattle: A free graphical interface for data science with R.  
## Version 5.3.0 Copyright (c) 2006-2018 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

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()  
## )

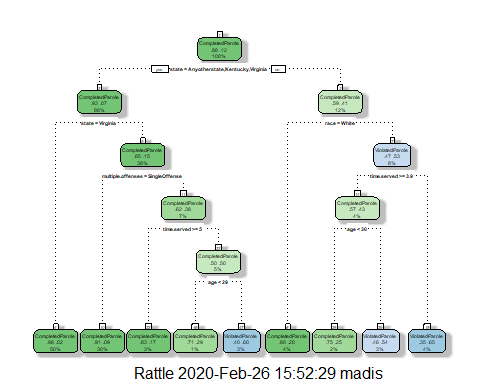
parole <- parole %>% mutate(male = as\_factor(as.character(male))) %>%  
 mutate(male = fct\_recode(male,  
 "Female" = "0",  
 "Male" = "1")) %>%   
 mutate(race = as\_factor(as.character(race))) %>%  
 mutate(race = fct\_recode(race,  
 "White" = "1",  
 "NotWhite" = "2")) %>%  
 mutate(state = as\_factor(as.character(state))) %>%  
 mutate(state = fct\_recode(state,  
 "Kentucky" = "2",  
 "Louisiana" = "3",  
 "Virginia" = "4",  
 "Anyotherstate" = "1")) %>%  
 mutate(crime = as\_factor(as.character(crime))) %>%  
 mutate(crime = fct\_recode(crime,  
 "larceny" = "2",  
 "drugrelated" = "3",  
 "drivingrelated" = "4",  
 "anyothercrime" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(as.character(multiple.offenses))) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses,  
 "SingleOffense" = "0",  
 "MultipleOffense" = "1")) %>%  
 mutate(violator = as\_factor(as.character(violator))) %>%  
 mutate(violator = fct\_recode(violator,  
 "CompletedParole" = "0",  
 "ViolatedParole" = "1"))

Task 1: Split the data into training and testing sets. Your training set should have 70% of the data. Use a random number (set.seed) of 12345.

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,]

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.  
**If the inmate described is White, then he is expected to complete parole, but if the inmate is non-white he is expected to violate parole. This was determined by the tree by following each node to either the left for yes, and right for no, until reaching the bottom tier which classifies paroles as completers or violators.**

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.

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

**It appears that a value of 0.030303 should be selected for the complexity parameter.**

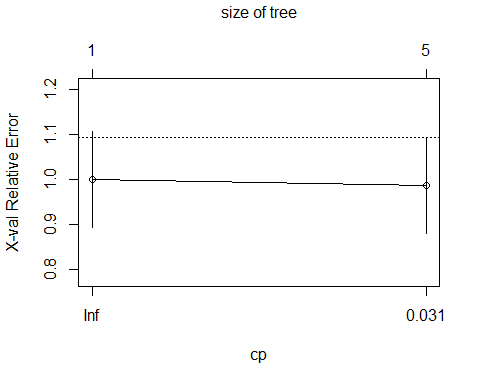
Task 5: Prune the tree from Task 2 back to the cp value that you selected in Task 4. Do not attempt to plot the tree. You will find that the resulting tree is known as a “root”. A tree that takes the form of a root is essentially a naive model that assumes that the prediction for all observations is the majority class. Which class (category) in the training set is the majority class (i.e., has the most observations)?

tree2 = rpart(violator ~., parole, cp=0.030303, method="class")

printcp(tree2)

##   
## Classification tree:  
## rpart(formula = violator ~ ., data = parole, method = "class",   
## cp = 0.030303)  
##   
## Variables actually used in tree construction:  
## [1] max.sentence multiple.offenses state time.served   
##   
## Root node error: 78/675 = 0.11556  
##   
## n= 675   
##   
## CP nsplit rel error xerror xstd  
## 1 0.032051 0 1.00000 1.00000 0.10648  
## 2 0.030303 4 0.83333 0.98718 0.10589

plotcp(tree2)



**It appears that the class in the training set that is the majority class is state.**

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 sensitivty 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.

train\_preds = predict(tree1, train, type="class")  
confusionMatrix(train\_preds,train$violator,positive="ViolatedParole")

## 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   
##

**We see that the accuracy of this model is 0.903, the sensitivity is 0.491, and the specificity is 0.957.**

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 sensitivty of this tree on the testing data. Comment on the quality of the model.

test\_preds = predict(tree1, test, type="class")  
confusionMatrix(test\_preds,test$violator,positive="ViolatedParole")

## 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   
##

**We see that the accuracy of this model is 0.896, the sensitivity is 0.434, and the specificity is 0.955.**

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”.

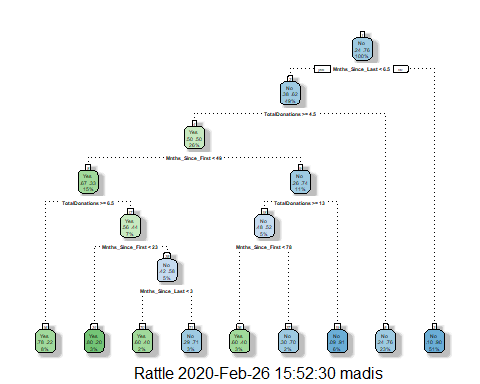
blood <- read\_csv("Blood.csv")

## 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))) %>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch,  
 "No" = "0",  
 "Yes" = "1"))

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.

set.seed(1234)  
  
train2.rows = createDataPartition(y = blood$DonatedMarch, p=0.7, list = FALSE) #70% in training  
train2 = blood[train2.rows,]   
test2 = blood[-train2.rows,]  
  
tree3 = rpart(DonatedMarch ~., train2, method="class")  
fancyRpartPlot(tree3)



train\_preds3 = predict(tree3, train2, type="class")  
confusionMatrix(train\_preds3,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 58 22  
## No 67 377  
##   
## Accuracy : 0.8302   
## 95% CI : (0.7952, 0.8613)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 8.243e-05   
##   
## Kappa : 0.4665   
##   
## Mcnemar's Test P-Value : 3.101e-06   
##   
## Sensitivity : 0.4640   
## Specificity : 0.9449   
## Pos Pred Value : 0.7250   
## Neg Pred Value : 0.8491   
## Prevalence : 0.2385   
## Detection Rate : 0.1107   
## Detection Prevalence : 0.1527   
## Balanced Accuracy : 0.7044   
##   
## 'Positive' Class : Yes   
##

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

**The CP level that best fits this model is 0.016.**

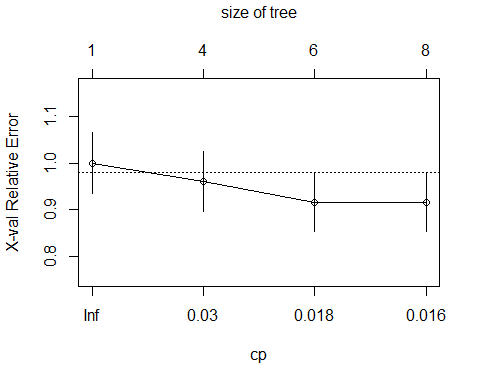
Task 10: Prune the tree back to the optimal cp value, make predictions, and use the confusionMatrix function on the both training and testing sets. Comment on the quality of the predictions.

tree4 = rpart(DonatedMarch ~., blood, cp=0.016, method="class")

printcp(tree4)

##   
## Classification tree:  
## rpart(formula = DonatedMarch ~ ., data = blood, method = "class",   
## cp = 0.016)  
##   
## 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 0.96067 0.064523  
## 3 0.016854 5 0.82022 0.91573 0.063431  
## 4 0.016000 7 0.78652 0.91573 0.063431

plotcp(tree4)



train\_preds4 = predict(tree4, train2, type="class")  
confusionMatrix(train\_preds4,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Yes No  
## Yes 47 13  
## No 78 386  
##   
## Accuracy : 0.8263   
## 95% CI : (0.7911, 0.8578)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.0001954   
##   
## Kappa : 0.4181   
##   
## Mcnemar's Test P-Value : 1.959e-11   
##   
## Sensitivity : 0.37600   
## Specificity : 0.96742   
## Pos Pred Value : 0.78333   
## Neg Pred Value : 0.83190   
## Prevalence : 0.23855   
## Detection Rate : 0.08969   
## Detection Prevalence : 0.11450   
## Balanced Accuracy : 0.67171   
##   
## 'Positive' Class : Yes   
##

**The quality of this model looks foor, with an accuracy of 0.8263 and a sensitivty of 0.376, and specificity level of 0.967.**