# David Wells

## Module 4 Assignment 3

library(tidyverse)  
library(tidymodels)  
library(caret)  
library(rpart)  
library(rpart.plot)  
library(rattle)  
library(RColorBrewer)

parole <- read\_csv("C:/Users/wdavi/Documents/BAN 502/Module 4/parole.csv")

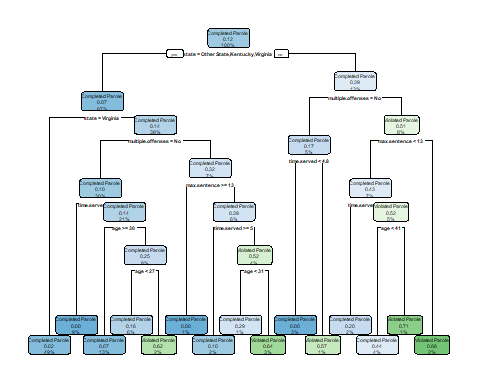
##   
## -- Column specification --------------------------------------------------------  
## cols(  
## male = col\_double(),  
## race = col\_double(),  
## age = col\_double(),  
## state = col\_double(),  
## time.served = col\_double(),  
## max.sentence = col\_double(),  
## multiple.offenses = col\_double(),  
## crime = col\_double(),  
## violator = col\_double()  
## )

parole = parole %>% mutate(male = as\_factor(male)) %>%   
 mutate(male = fct\_recode(male, "Male" = "0", "Female" = "1" ))   
  
parole = parole %>% mutate(race = as\_factor(race)) %>%   
 mutate(race = fct\_recode(race, "White" = "1", "Other" = "2" ))   
  
parole = parole %>% mutate(state = as\_factor(state)) %>%   
 mutate(state = fct\_recode(state, "Other State" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4" ))   
  
parole = parole %>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "No" = "0", "Yes" = "1" ))  
  
parole = parole %>% mutate(crime = as\_factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "Other Crime" = "1", "Larceny" = "2", "Drug-related" = "3", "Driving-related" = "4" ))  
  
parole = parole %>% mutate(violator = as\_factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "Completed Parole" = "0", "Violated Parole" = "1" ))   
  
str(parole)

## spec\_tbl\_df [675 x 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ male : Factor w/ 2 levels "Male","Female": 2 1 2 2 2 2 2 1 1 2 ...  
## $ race : Factor w/ 2 levels "White","Other": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num [1:675] 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "Other State",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num [1:675] 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : num [1:675] 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "Other Crime",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "Completed Parole",..: 1 1 1 1 1 1 1 1 1 1 ...  
## - 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()  
## .. )

set.seed(12345)   
parole\_split = initial\_split(parole, prob = 0.70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

parole\_recipe = recipe(violator ~., train)  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
parole\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(parole\_recipe)  
  
parole\_fit = fit(parole\_wflow, train)  
  
tree = parole\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
rpart.plot(tree,tweak = 1.2,gap=0,space=0)



# I preface this explanation by saying I had a hard time reading the tree, due to the size of the text. If I read the tree correctly, I would categorize this person as a parole violator. I first look to see what state this person is in, the tree groups the states into \*Louisiana\* and all others. From there I look to see if this person had multiple offenses...if so I then look to the amount of time they spent in prison, with the tree making the divide at either 4.3 or 4.8 years...I can't really tell. Either way, with the person in question having served 5 years in prison, I would then see he or she is categorized by the tree into the \*Violated Parole\* bin.

parole\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.03389831 0 1.0000000 1.000000 0.1223796  
## 2 0.02542373 3 0.8983051 1.135593 0.1292432  
## 3 0.01694915 5 0.8474576 1.135593 0.1292432  
## 4 0.01355932 6 0.8305085 1.220339 0.1332155  
## 5 0.01129944 11 0.7627119 1.288136 0.1362352  
## 6 0.01000000 14 0.7288136 1.288136 0.1362352

# It appears that 14 splits the model used provides the optimal xerror of 1.101695. However, 11 splits also seems to provide the same xerror. It could be argued 11 splits, while having a higher rel error, may be optimal as the xerror doesn't change and it provides fewer splits.

set.seed(123)  
folds = vfold\_cv(train, v = 5)  
  
parole\_recipe2 = recipe(violator ~., train)  
   
tree\_model2 = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid2 = grid\_regular(cost\_complexity(),  
 levels = 25)   
  
parole\_wflow2 =   
 workflow() %>%   
 add\_model(tree\_model2) %>%   
 add\_recipe(parole\_recipe2)  
  
tree\_res =   
 parole\_wflow2 %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid2  
 )

##   
## Attaching package: 'rlang'

## The following objects are masked from 'package:purrr':  
##   
## %@%, as\_function, flatten, flatten\_chr, flatten\_dbl, flatten\_int,  
## flatten\_lgl, flatten\_raw, invoke, list\_along, modify, prepend,  
## splice

##   
## Attaching package: 'vctrs'

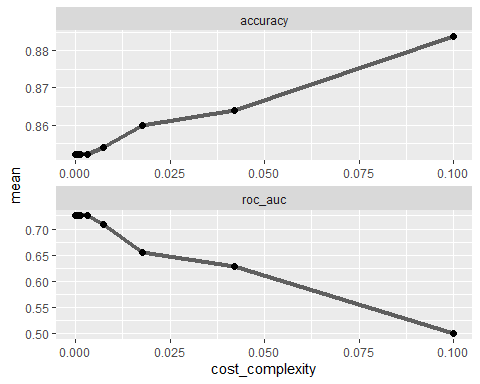
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## The following object is masked from 'package:tibble':  
##   
## data\_frame

tree\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [405/102]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [405/102]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [406/101]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [406/101]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [406/101]> Fold5 <tibble [50 x 5]> <tibble [0 x 1]>

tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)



best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.1 Preprocessor1\_Model25

# From the analysis, it appears a cp=0.1 is the optimal value.

final\_wf =   
 parole\_wflow2 %>%   
 finalize\_workflow(best\_tree)  
  
final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
#fancyRpartPlot(tree, tweak = 1.5)

treepred = predict(final\_fit, train, type = "class")  
head(treepred)

## # A tibble: 6 x 1  
## .pred\_class   
## <fct>   
## 1 Completed Parole  
## 2 Completed Parole  
## 3 Completed Parole  
## 4 Completed Parole  
## 5 Completed Parole  
## 6 Completed Parole

confusionMatrix(treepred$.pred\_class,train$violator,positive="Completed Parole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed Parole Violated Parole  
## Completed Parole 448 59  
## Violated Parole 0 0  
##   
## Accuracy : 0.8836   
## 95% CI : (0.8525, 0.9102)  
## No Information Rate : 0.8836   
## P-Value [Acc > NIR] : 0.5346   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 4.321e-14   
##   
## Sensitivity : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8836   
## Neg Pred Value : NaN   
## Prevalence : 0.8836   
## Detection Rate : 0.8836   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Completed Parole  
##

# The accuracy of the tree is 0.8836, the same as the naive rate.

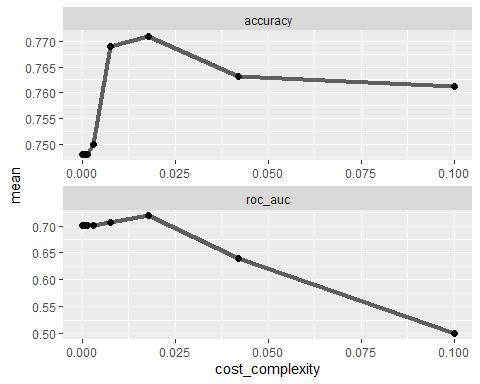
# Read In Data  
blood <- read\_csv("C:/Users/wdavi/Documents/BAN 502/Module 4/Blood.csv")

##   
## -- Column specification --------------------------------------------------------  
## cols(  
## Mnths\_Since\_Last = col\_double(),  
## TotalDonations = col\_double(),  
## Total\_Donated = col\_double(),  
## Mnths\_Since\_First = col\_double(),  
## DonatedMarch = col\_double()  
## )

# Mutate Data  
blood = blood %>% mutate(DonatedMarch = as\_factor(DonatedMarch)) %>%   
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1" ))  
  
# Split Data  
set.seed(1234)   
blood\_split = initial\_split(blood, prop = 0.7, strata = DonatedMarch)  
train2 = training(blood\_split)   
test2 = testing(blood\_split)  
  
# Create Classification Tree  
set.seed(1234)  
folds2 = vfold\_cv(train2, v = 5)  
  
blood\_recipe = recipe(DonatedMarch ~., train2)  
   
tree\_model3 = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid3 = grid\_regular(cost\_complexity(),levels = 25)   
  
blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model3) %>%   
 add\_recipe(blood\_recipe)  
  
tree\_res3 =   
 blood\_wflow %>%   
 tune\_grid(  
 resamples = folds2,  
 grid = tree\_grid3)  
  
tree\_res3

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [419/105]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [419/105]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [419/105]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [419/105]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [420/104]> Fold5 <tibble [50 x 5]> <tibble [0 x 1]>

tree\_res3 %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)

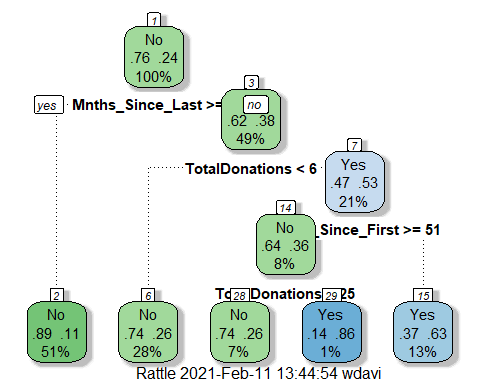


best\_tree2 = tree\_res3 %>%  
 select\_best("accuracy")  
  
best\_tree2

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.0178 Preprocessor1\_Model23

# The optimal cp appears to be around 0.015, based on the graph. The actual optimal cp = 0.017.

final\_wf2 =   
 blood\_wflow %>%   
 finalize\_workflow(best\_tree2)  
  
final\_fit2 = fit(final\_wf2, train2)  
  
tree2 = final\_fit2 %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree2, tweak = 1.5)



# Testing Set Accuracy  
  
treepred2 = predict(final\_fit2, test2, type = "class")  
head(treepred2)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 No   
## 3 Yes   
## 4 No   
## 5 No   
## 6 Yes

confusionMatrix(treepred2$.pred\_class,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 160 38  
## Yes 11 15  
##   
## Accuracy : 0.7812   
## 95% CI : (0.7213, 0.8336)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.2944580   
##   
## Kappa : 0.2653   
##   
## Mcnemar's Test P-Value : 0.0002038   
##   
## Sensitivity : 0.28302   
## Specificity : 0.93567   
## Pos Pred Value : 0.57692   
## Neg Pred Value : 0.80808   
## Prevalence : 0.23661   
## Detection Rate : 0.06696   
## Detection Prevalence : 0.11607   
## Balanced Accuracy : 0.60935   
##   
## 'Positive' Class : Yes   
##

# Training Set Accuracy  
treepred3 = predict(final\_fit2, train2, type = "class")  
head(treepred3)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 No   
## 5 No   
## 6 Yes

confusionMatrix(treepred3$.pred\_class,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 373 76  
## Yes 26 49  
##   
## Accuracy : 0.8053   
## 95% CI : (0.7688, 0.8384)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.009409   
##   
## Kappa : 0.3789   
##   
## Mcnemar's Test P-Value : 1.224e-06   
##   
## Sensitivity : 0.39200   
## Specificity : 0.93484   
## Pos Pred Value : 0.65333   
## Neg Pred Value : 0.83073   
## Prevalence : 0.23855   
## Detection Rate : 0.09351   
## Detection Prevalence : 0.14313   
## Balanced Accuracy : 0.66342   
##   
## 'Positive' Class : Yes   
##

# The training set had an accuracy of 0.8053. The testing set had an accuracy 0.7812. Both were higher than the naive accuracy rate of 0.7634. Not too great I would assume, but better than outright guessing.