## Classification Trees

### Libraries & Data

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

parole <- read\_csv("parole.csv")  
#str(parole)  
#summary(parole)

parole <- parole %>%  
 mutate(male = as.factor(male)) %>%  
 mutate(race = as.factor(race)) %>%  
 mutate(state = as.factor(state)) %>%  
 mutate(crime = as.factor(crime)) %>%  
 mutate(multiple.offenses = as.factor(multiple.offenses)) %>%  
 mutate(violator = as.factor(violator)) %>%  
 mutate(male = fct\_recode(male, "male" = "1", "female" = "0")) %>%  
 mutate(race = fct\_recode(race, "white" = "1", "other" = "2")) %>%  
 mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4")) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "yes" = "1", "no" = "0")) %>%  
 mutate(crime = fct\_recode(crime, "Other" = "1", "larceny" = "2", "drug-related crime" = "3", "driving-related crime" = "4")) %>%  
 mutate(violator = fct\_recode(violator, "yes" = "1", "no" = "0"))  
  
#str(parole)

### Task 1: Testing and Training Sets

set.seed(12345)  
parole\_split = initial\_split(parole, prop = 0.7, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

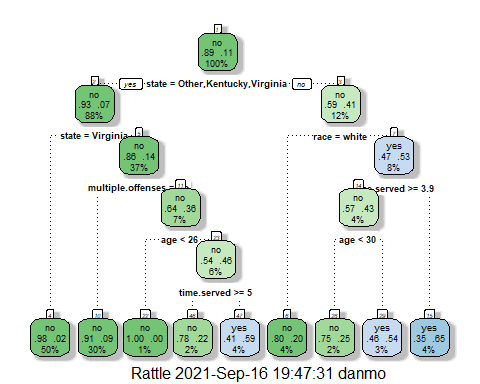
### Task 2: Classification Tree

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")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

fancyRpartPlot(tree, tweak = 1.2)



### Task 3

40 year old parolee from Louisiana who served 5 years in prison with a sentence of 10 years and multiple offenses

**state = Other, Kentucky, Virginia:** no  
**race = white:** yes

If the race = white, the parolee will not violate parole according to the model. 2% of the data matches this case and 80% do not violate parole.

**state = Other, Kentucky, Virginia:** no  
**race = white:** no  
**time\_served >= 3.9:** yes  
**age<30:** no

If the race = Other, the model predicts that the parolee will violate parole but just barely. 3% of the data fit’s into this exact category and there is a 54% chance the parolee will violate parole.

### Task 4

It chose a CP value of 0.01000000 which corresponds to an xerror of 1.314815 and 8 splits. The minimal xerror value actually occurs with 0 folds. Every additional fold increases the xerror. The minimal xerror value of a tree with splits is 1.240741 which corresponds to 3 splits and a CP value of 0.01851852. The model did not use the optimal CP value

parole\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.03086420 0 1.0000000 1.000000 0.1280444  
## 2 0.01851852 3 0.9074074 1.240741 0.1403860  
## 3 0.01388889 4 0.8888889 1.277778 0.1421129  
## 4 0.01000000 8 0.8333333 1.314815 0.1437987

### Task 5

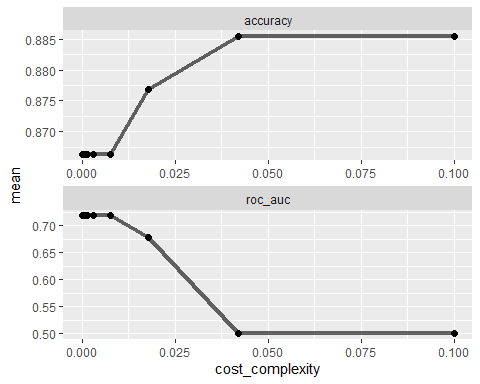
set.seed(123)  
folds = vfold\_cv(train, v = 5)

parole\_recipe = recipe(violator ~ ., train)  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%  
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(), levels = 25)  
  
parole\_wflow =  
 workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(parole\_recipe)  
  
tree\_res =  
 parole\_wflow %>%  
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )

tree\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [376/95]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [377/94]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [377/94]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [377/94]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [377/94]> 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)



### Task 6

A CP value of 0.04216965 yeilded the optimal accuracy in this model

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

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.0422 Preprocessor1\_Model24

### Task 7

final\_wf =  
 parole\_wflow %>%  
 finalize\_workflow(best\_tree)

final\_fit = fit(final\_wf,train)  
  
tree= final\_fit %>%  
 extract\_fit\_parsnip() %>%   
 pluck("fit")  
  
#fancyRpartPlot(tree, tweak = 1.2)

### Task 8

table(train$violator)

##   
## no yes   
## 417 54

Since the cp value generated a tree with no splits, the decision tree is telling us that you can categorize everyone as “not a violator”. Because the data set is unbalanced, there are more data points that are not violators. In this case, predicting that any parolee will not violate parole will result in 88.5% accuracy solely based on the fact that 88.5% of the data points are non-violators in this data set.

417/(417+54)

## [1] 0.8853503

### Task 9

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

## Rows: 748 Columns: 5

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (5): Mnths\_Since\_Last, TotalDonations, Total\_Donated, Mnths\_Since\_First,...

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

#str(Blood)  
#summary(Blood)

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

### Task 9 cont.

set.seed(1234)  
blood\_split = initial\_split(Blood, prop = 0.7, strata = DonatedMarch)  
train2 = training(blood\_split)  
test2 = testing(blood\_split)

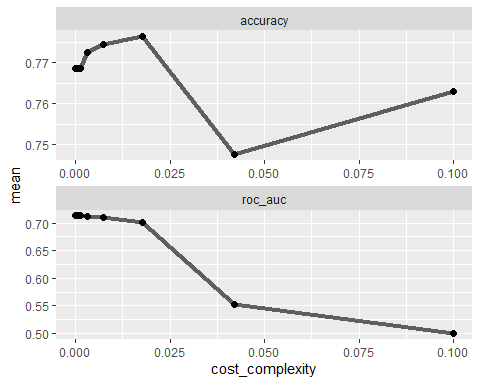
set.seed(1234)  
folds = vfold\_cv(train2, v = 5)

blood\_recipe = recipe(DonatedMarch ~ ., train2)  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%  
 set\_engine("rpart", model = TRUE) %>%  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(), levels = 25)  
  
blood\_wflow =  
 workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(blood\_recipe)  
  
tree\_res =  
 blood\_wflow %>%  
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )

tree\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [418/105]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [418/105]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [418/105]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [419/104]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [419/104]> 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)



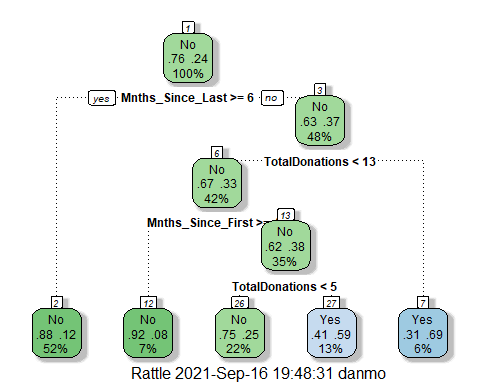
best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

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

### Task 10

final\_wf =  
 blood\_wflow %>%  
 finalize\_workflow(best\_tree)

final\_fit = fit(final\_wf,train2)  
  
tree= final\_fit %>%  
 extract\_fit\_parsnip() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 1.2)



## Task 11

The “No Information Rate” accuracy is 76.29% meaning that if we categorized everyone as “no”, the model would be 76.29% accurate. The training set resulted in an accuracy of 80.69% which is statistically significant according to the p-value. The testing set resulted in an accuracy of 78.22% which is not significant according to the p-value. This makes me question the usability of this model because the accuracy decreased when tested with new data and accuracy is not significant better than assuming “no” for everyone. You would have to consider the effects of assuming “no” for everyone and missing a “yes”. In this case, we are assuming that a certain person did not donate blood in March which doesn’t have an obvious consequence so it would depend on how this data needs to be used.

treepred = predict(final\_fit, train2, type = "class")  
head(treepred)

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 361 63  
## Yes 38 61  
##   
## Accuracy : 0.8069   
## 95% CI : (0.7704, 0.8399)  
## No Information Rate : 0.7629   
## P-Value [Acc > NIR] : 0.009214   
##   
## Kappa : 0.4263   
##   
## Mcnemar's Test P-Value : 0.016936   
##   
## Sensitivity : 0.4919   
## Specificity : 0.9048   
## Pos Pred Value : 0.6162   
## Neg Pred Value : 0.8514   
## Prevalence : 0.2371   
## Detection Rate : 0.1166   
## Detection Prevalence : 0.1893   
## Balanced Accuracy : 0.6983   
##   
## 'Positive' Class : Yes   
##

treepred\_test = predict(final\_fit, test2, type = "class")  
head(treepred\_test)

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 150 28  
## Yes 21 26  
##   
## Accuracy : 0.7822   
## 95% CI : (0.7225, 0.8343)  
## No Information Rate : 0.76   
## P-Value [Acc > NIR] : 0.2434   
##   
## Kappa : 0.3753   
##   
## Mcnemar's Test P-Value : 0.3914   
##   
## Sensitivity : 0.4815   
## Specificity : 0.8772   
## Pos Pred Value : 0.5532   
## Neg Pred Value : 0.8427   
## Prevalence : 0.2400   
## Detection Rate : 0.1156   
## Detection Prevalence : 0.2089   
## Balanced Accuracy : 0.6793   
##   
## 'Positive' Class : Yes   
##