# Module 4 - Assignment 1

## Classification Trees

### Nicole Westrick

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

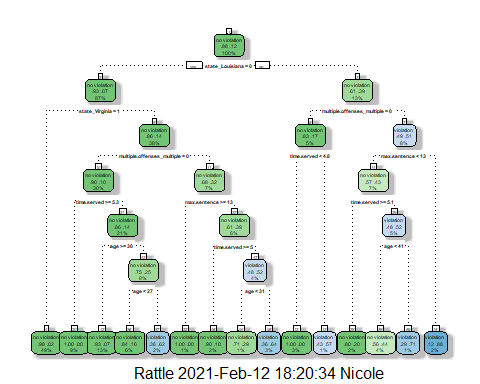
**Task 1:**

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

**Task 2:**

train\_recipe = recipe(violator ~., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
train\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(train\_recipe)  
  
train\_fit = fit(train\_wflow, train)

tree = train\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 1.2)



**Task 3:**

A 40-year old parolee from Louisiana who served 5 years in prison, with a sentence of 10 years, and committed multiple offenses has 56% chance of not violating probation (44% chance of violating). Starting at the top of the tree and working down, Louisiana is not in the states of Other, Kentucky, or Virginia. Moving to the right, the parolee has multiple offenses. Staying to the right, the parolee’s max sentence was 10 years, which is less than 13. Moving to the left, the parolee served 5 years, which is less than 5.1. Moving to the right, the parolee is 40 years old, which is less than 41. Finally, moving to the left and down to the bottom of the tree, the resulting values are .56 for no violation and .44 for violation.

**Task 4:**

train\_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 1.101695 0.1275886  
## 3 0.01694915 5 0.8474576 1.084746 0.1267465  
## 4 0.01355932 6 0.8305085 1.084746 0.1267465  
## 5 0.01129944 11 0.7627119 1.186441 0.1316539  
## 6 0.01000000 14 0.7288136 1.152542 0.1300561

The optimal cp value is 0.0339. This value is not associated with the above classification tree. The optimal cp value is with 0 splits, unlike the created tree.

**Task 5:**

set.seed(123)  
folds = vfold\_cv(train, v = 5)  
  
train\_recipe = recipe(violator ~., train) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25)   
  
train\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(train\_recipe)  
  
tree\_res =   
 train\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )

##   
## 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'

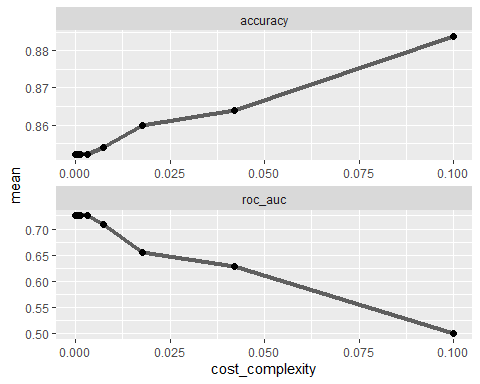
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## 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)



It appears that the optimal cp value for maximum accuracy is 0.100.

**Task 6:**

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

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

This confirms 0.1 as optimal cp value.

**Task 7:**

final\_wf =   
 train\_wflow %>%   
 finalize\_workflow(best\_tree)  
  
final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

#fancyRpartPlot(tree, tweak = 1.5)

Error for plot due to 0.1 cp value is with 0 splits, or considered a “root.”

**Task 8:**

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

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

confusionMatrix(treepred$.pred\_class,train$violator)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction no violation violation  
## no violation 448 59  
## violation 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 : no violation   
##

This model has 88% accuracy. This, however, is equal to the no information rate which is not good.

**Task 9:**

Blood <- read\_csv("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()  
## )

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

**Task 9 cont.:**

set.seed(1234)  
Blood\_split = initial\_split(Blood, prob = 0.70, strata = DonatedMarch)  
train2 = training(Blood\_split)  
test2 = testing(Blood\_split)  
  
train2\_recipe = recipe(DonatedMarch ~., train2)  
  
tree2\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
train2\_wflow =   
 workflow() %>%   
 add\_model(tree2\_model) %>%   
 add\_recipe(train2\_recipe)  
  
train2\_fit = fit(train2\_wflow, train2)  
  
tree2 = train2\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

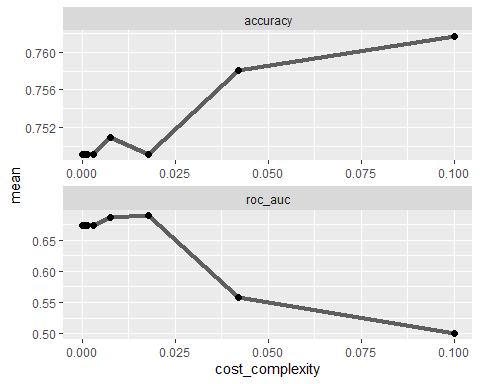
train2\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.03731343 0 1.0000000 1.000000 0.07538784  
## 2 0.01865672 3 0.8880597 1.059701 0.07687690  
## 3 0.01492537 5 0.8507463 1.059701 0.07687690  
## 4 0.01000000 8 0.7985075 1.037313 0.07633165

set.seed(1234)  
folds = vfold\_cv(train2, v = 5)  
  
train2\_recipe = recipe(DonatedMarch ~., train2)  
  
tree2\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree2\_grid = grid\_regular(cost\_complexity(),  
 levels = 25)   
  
train2\_wflow =   
 workflow() %>%   
 add\_model(tree2\_model) %>%   
 add\_recipe(train2\_recipe)  
  
tree2\_res =   
 train2\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree2\_grid  
 )  
  
tree2\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [449/113]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [449/113]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [450/112]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [450/112]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [450/112]> Fold5 <tibble [50 x 5]> <tibble [0 x 1]>

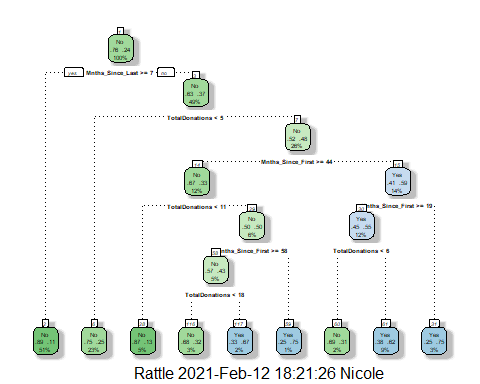
tree2\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size = 1.5, alpha = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)



The cp value 0.01, with a xerror of 1.037313, appears to be optimal to maximize accuracy.

**Task 10:**

fancyRpartPlot(tree2, tweak = 1.2)



**Task 11:**

tree2pred = predict(train2\_fit, train2, type = "class")  
head(tree2pred)

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

confusionMatrix(tree2pred$.pred\_class,train2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 399 78  
## Yes 29 56  
##   
## Accuracy : 0.8096   
## 95% CI : (0.7747, 0.8413)  
## No Information Rate : 0.7616   
## P-Value [Acc > NIR] : 0.00368   
##   
## Kappa : 0.4004   
##   
## Mcnemar's Test P-Value : 3.478e-06   
##   
## Sensitivity : 0.9322   
## Specificity : 0.4179   
## Pos Pred Value : 0.8365   
## Neg Pred Value : 0.6588   
## Prevalence : 0.7616   
## Detection Rate : 0.7100   
## Detection Prevalence : 0.8488   
## Balanced Accuracy : 0.6751   
##   
## 'Positive' Class : No   
##

We are 81% accurate on the training set. This is good, because it is higher than the no information rate.

tree2pred\_test = predict(train2\_fit, test2, type = "class")  
head(tree2pred\_test)

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

confusionMatrix(tree2pred\_test$.pred\_class,test2$DonatedMarch)

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 135 25  
## Yes 7 19  
##   
## Accuracy : 0.828   
## 95% CI : (0.7659, 0.8792)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.020853   
##   
## Kappa : 0.4454   
##   
## Mcnemar's Test P-Value : 0.002654   
##   
## Sensitivity : 0.9507   
## Specificity : 0.4318   
## Pos Pred Value : 0.8438   
## Neg Pred Value : 0.7308   
## Prevalence : 0.7634   
## Detection Rate : 0.7258   
## Detection Prevalence : 0.8602   
## Balanced Accuracy : 0.6913   
##   
## 'Positive' Class : No   
##

We are 83% accurate on the testing set. This is good, because it is higher than the no information rate.