# Classification Tree Assignment

## Drew Rabil

### Module 4 - Assignment 1

parole <- read\_csv("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), race = as\_factor(race), state = as\_factor(state), crime = as\_factor(crime), multiple.offenses = as\_factor(multiple.offenses), 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, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "Other" = "1")) %>%  
 mutate(crime = fct\_recode(crime, "Larceny" = "2", "Drug-related crime" = "3", "Driving-related crime" = "4", "Other" = "1")) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "Yes" = "1", "No" = "0")) %>%  
 mutate(violator = fct\_recode(violator, "Violator" = "1", "nonViolator" = "0"))  
  
#str(parole)  
#summary(parole)

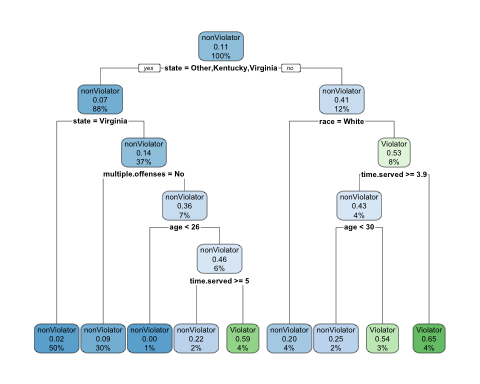
## Task 1

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

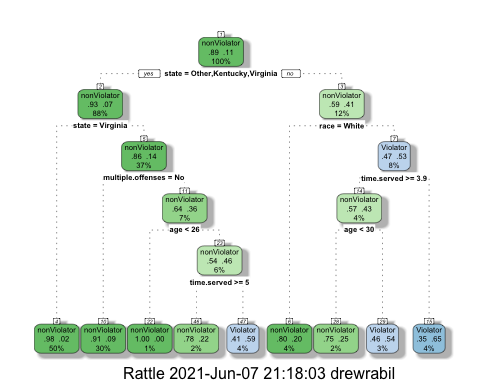
## Task 2

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)  
  
set.seed(12345)  
parole\_fit = fit(parole\_wflow, train)

tree = parole\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
rpart.plot(tree)



fancyRpartPlot(tree)



## Task 3

**For the tree created in Task 2, how would you classify a 40 year-old parolee from Louisiana who served 5 years in prison, with a sentence of 10 years, and committed multiple offenses?**

I started at the top of the tree where it focused on the state variable. Since the parolee was from Louisiana, and it says “state= Other, Kentucky, Virginia, I answered NO and moved to the right. We do not know the race of the parolee. If the parolee was white, we would move to the left and classify him as a NonViolator. If the parolee was any other race, we move to the right. The next split asks us if time served is greater than or equal to 3.9. Since he has served 5 years, we answer yes and move to the left. In the last split, we’re asked if the parolee’s age is less than 50. Since the parolee is 40 years old, we move to the right and conclude we should classify him as a Violator.

The multiple offenses variable did not appear as part of the decision tree, so I did not use that information in my decision process.

## Task 4

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 1.148148 0.1358793  
## 3 0.01388889 4 0.8888889 1.277778 0.1421129  
## 4 0.01000000 8 0.8333333 1.296296 0.1429608

**Which cp value is optimal (recall that the optimal cp corresponds to the minimized “xerror” value)? Is the tree from Task 2 associated with this optimal cp?**

The optimal cp is 0.01851852 with 3 splits because it has the lowest “xerror”. The tree from Task 2 is not associated with this optimal cp because it is the tree with 8 splits instead of 3.

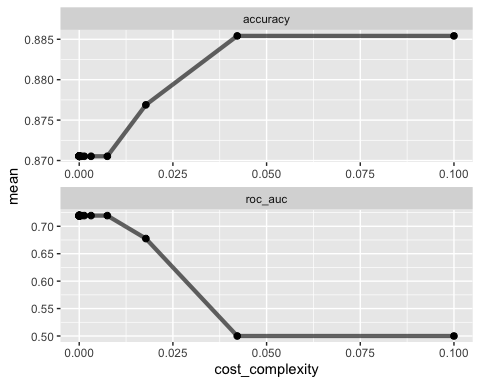
## Task 5

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

parole\_recipe2 = recipe(violator ~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model2 = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid = 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\_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 × 5]> <tibble [0 × 1]>  
## 2 <split [377/94]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [377/94]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [377/94]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [377/94]> Fold5 <tibble [50 × 5]> <tibble [0 × 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

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

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

The cp value of 0.4216965 yields the “optimal” accuracy value.

## Task 7

final\_wf =   
 parole\_wflow2 %>%   
 finalize\_workflow(best\_tree)  
  
final\_fit = fit(final\_wf, train)  
  
tree2 = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
  
#fancyRpartPlot(tree2) #naive model

## Task 8

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

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction nonViolator Violator  
## nonViolator 417 54  
## Violator 0 0  
##   
## Accuracy : 0.8854   
## 95% CI : (0.8531, 0.9127)  
## No Information Rate : 0.8854   
## P-Value [Acc > NIR] : 0.5362   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 5.498e-13   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8854   
## Prevalence : 0.1146   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Violator   
##

**What is the accuracy of the “root” that you generated in Task 7?**

The accuracy of the “root” is also known as the no information rate (or naive model) that appears after you run a caret confusion matrix.

For the “root” in Task 7, the accuracy is 0.8854.

## 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, "No" = "0", "Yes" = "1"))  
  
  
str(Blood)

## spec\_tbl\_df [748 × 5] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ Mnths\_Since\_Last : num [1:748] 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num [1:748] 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num [1:748] 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num [1:748] 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 2 1 2 2 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Mnths\_Since\_Last = col\_double(),  
## .. TotalDonations = col\_double(),  
## .. Total\_Donated = col\_double(),  
## .. Mnths\_Since\_First = col\_double(),  
## .. DonatedMarch = col\_double()  
## .. )

summary(Blood)

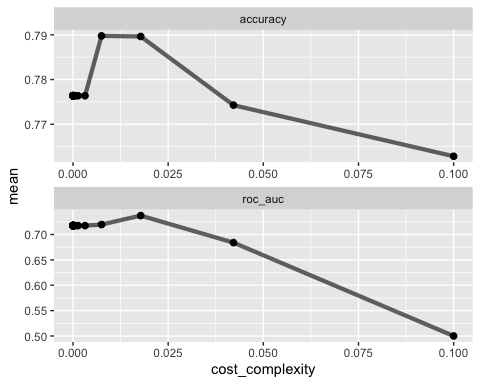
## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :34.28   
## 3rd Qu.:14.000 3rd Qu.: 7.000 3rd Qu.: 1750 3rd Qu.:50.00   
## Max. :74.000 Max. :50.000 Max. :12500 Max. :98.00   
## DonatedMarch  
## No :570   
## Yes:178   
##   
##   
##   
##

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

set.seed(1234)  
folds2 = vfold\_cv(train2, v = 5)  
  
blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model\_blood = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid\_blood = grid\_regular(cost\_complexity(),  
 levels = 25)   
  
blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model\_blood) %>%   
 add\_recipe(blood\_recipe)  
  
tree\_res\_blood =   
 blood\_wflow %>%   
 tune\_grid(  
 resamples = folds2,  
 grid = tree\_grid\_blood  
 )  
  
tree\_res\_blood

## # 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 × 5]> <tibble [0 × 1]>  
## 2 <split [418/105]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [418/105]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [419/104]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [419/104]> Fold5 <tibble [50 × 5]> <tibble [0 × 1]>

tree\_res\_blood %>%  
 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)

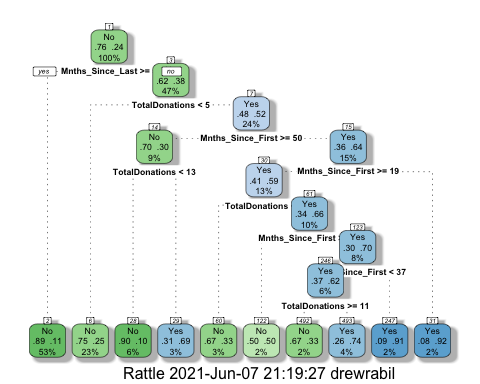


best\_tree\_blood = tree\_res\_blood %>%  
 select\_best("accuracy")  
  
best\_tree\_blood

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00750 Preprocessor1\_Model22

## Task 10

final\_wf\_blood =   
 blood\_wflow %>%   
 finalize\_workflow(best\_tree\_blood)  
  
final\_fit\_blood = fit(final\_wf\_blood, train2)  
  
tree3 = final\_fit\_blood %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree3, tweak=1.5)



## Task 11

treepred2 = predict(final\_fit\_blood, train2, type = "class")  
head(treepred2)

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

confusionMatrix(treepred2$.pred\_class,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 386 75  
## Yes 13 49  
##   
## Accuracy : 0.8317   
## 95% CI : (0.7969, 0.8628)  
## No Information Rate : 0.7629   
## P-Value [Acc > NIR] : 7.816e-05   
##   
## Kappa : 0.4381   
##   
## Mcnemar's Test P-Value : 7.893e-11   
##   
## Sensitivity : 0.39516   
## Specificity : 0.96742   
## Pos Pred Value : 0.79032   
## Neg Pred Value : 0.83731   
## Prevalence : 0.23709   
## Detection Rate : 0.09369   
## Detection Prevalence : 0.11855   
## Balanced Accuracy : 0.68129   
##   
## 'Positive' Class : Yes   
##

treepred3 = predict(final\_fit\_blood, test2, type = "class")  
head(treepred)

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

confusionMatrix(treepred3$.pred\_class,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 151 36  
## Yes 20 18  
##   
## Accuracy : 0.7511   
## 95% CI : (0.6893, 0.8062)  
## No Information Rate : 0.76   
## P-Value [Acc > NIR] : 0.65600   
##   
## Kappa : 0.2408   
##   
## Mcnemar's Test P-Value : 0.04502   
##   
## Sensitivity : 0.3333   
## Specificity : 0.8830   
## Pos Pred Value : 0.4737   
## Neg Pred Value : 0.8075   
## Prevalence : 0.2400   
## Detection Rate : 0.0800   
## Detection Prevalence : 0.1689   
## Balanced Accuracy : 0.6082   
##   
## 'Positive' Class : Yes   
##

The accuracy of the tree from Task 10 on the training set is 0.8317. The accuracy of the tree on the testing set is 0.7511. The accuracy did decrease slightly by 8% for the testing set compared to the training set. However, the drop-off in accuracy was not too large resulting in me being able to conclude the the tree’s performance is good.