# Daniel Rauscher

# Module 4 Assignment 1

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

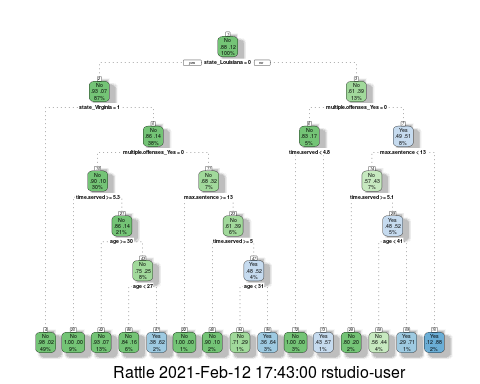
## Task 1

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

## Task 2

parole\_recipe = recipe(violator ~., train) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
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")  
  
fancyRpartPlot(tree)



## Task 3

I would classify this person as not being a violator. Only 4% of the data fits in this node, with 56% not being a violator. With the parolee being from Louisiana we started by going to the right. They did commit multiple offenses so we move to the right at that split. Their max time was was less than 13 years so we move to the left at the next split. Their time served is not greater than or equal to 5.1, so we move to the right. Their age is less than 41 so we move to the right.

## Task 4

Using the minimal xerror value, we see the optimal CP value is .0135. This results in 6 splits. The above tree is not associated with this optimal CP.

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 1.101695 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

## Task 5

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

parole\_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)  
  
parole\_wflow =  
 workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(parole\_recipe)  
  
tree\_res =  
 parole\_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'

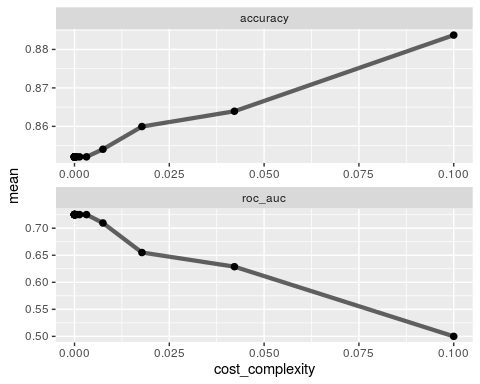
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## 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 × 5]> <tibble [0 × 1]>  
## 2 <split [405/102]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [406/101]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [406/101]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [406/101]> 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

The CP value that yeilds the optimal accuracy level is 0.10.

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

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

## Task 7

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

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

## Task 8

Using the root generated in task 7, the accuracy rate is 88.36%.

## Task 9

The CP value that seems optimal to maximize accuracy is around 0.050. Much after this the gain on accuracy plateaus.

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()  
## )

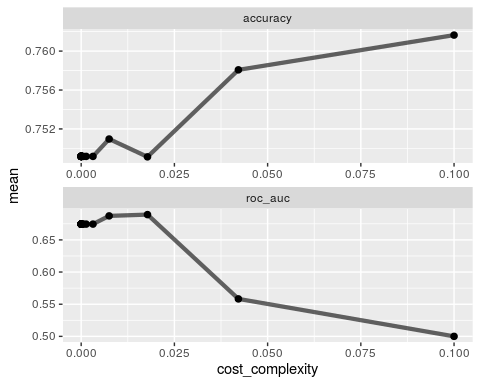
Blood <- Blood %>%  
 mutate(DonatedMarch=as\_factor(DonatedMarch)) %>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1"))   
  
set.seed(1234)  
Blood\_split <- initial\_split(Blood, prob = 0.70, strata = DonatedMarch)  
train2 = training(Blood\_split)  
test2 = testing(Blood\_split)

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

Blood\_recipe= recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
  
tree\_model2 = decision\_tree(cost\_complexity= tune()) %>%  
 set\_engine("rpart", model=TRUE) %>%  
 set\_mode("classification")  
  
  
tree\_grid2= grid\_regular(cost\_complexity(), levels = 25)  
  
Blood\_wflow =  
 workflow() %>%  
 add\_model(tree\_model2) %>%  
 add\_recipe(Blood\_recipe)  
  
tree\_res2 =  
 Blood\_wflow %>%  
 tune\_grid(resamples=folds, grid=tree\_grid2)  
  
  
tree\_res2

## # 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 × 5]> <tibble [0 × 1]>  
## 2 <split [449/113]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [450/112]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [450/112]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [450/112]> Fold5 <tibble [50 × 5]> <tibble [0 × 1]>

tree\_res2 %>%  
 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)

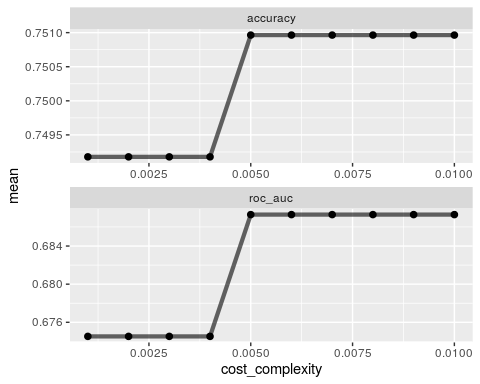


## Task 10

Blood\_recipe= recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
  
tree\_model2 = decision\_tree(cost\_complexity= tune()) %>%  
 set\_engine("rpart", model=TRUE) %>%  
 set\_mode("classification")  
  
  
tree\_grid2= expand.grid(cost\_complexity=seq(0.001,0.01,by=0.001))  
  
Blood\_wflow =  
 workflow() %>%  
 add\_model(tree\_model2) %>%  
 add\_recipe(Blood\_recipe)  
  
tree\_res2 =  
 Blood\_wflow %>%  
 tune\_grid(resamples=folds, grid=tree\_grid2)  
  
  
tree\_res2

## # 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 [20 × 5]> <tibble [0 × 1]>  
## 2 <split [449/113]> Fold2 <tibble [20 × 5]> <tibble [0 × 1]>  
## 3 <split [450/112]> Fold3 <tibble [20 × 5]> <tibble [0 × 1]>  
## 4 <split [450/112]> Fold4 <tibble [20 × 5]> <tibble [0 × 1]>  
## 5 <split [450/112]> Fold5 <tibble [20 × 5]> <tibble [0 × 1]>

tree\_res2 %>%  
 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)

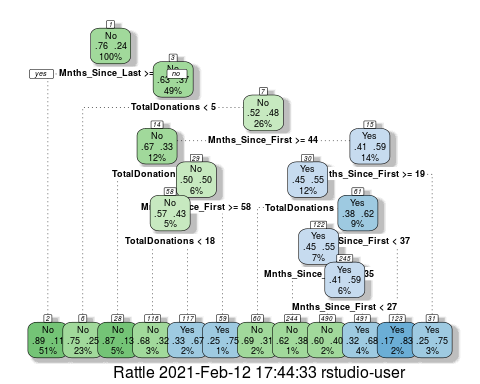


best\_tree2 = tree\_res2 %>%  
 select\_best("accuracy")  
  
best\_tree2

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.005 Preprocessor1\_Model05

final\_wf =   
 Blood\_wflow %>%   
 finalize\_workflow(best\_tree2)

final\_fit = fit(final\_wf, train2)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 1.5)



## Task 11

The accuracy of this tree from task 10 on the training set is 81.67% and the accuracy on the testing set is 80.11%. The trees accuracy is better on both sets than the naive model on both sets. However in the testing set the Pvalue is greater than .05, meaning their is not a significant difference.

treepred= predict(final\_fit, train2, type = "class")  
confusionMatrix(treepred$.pred\_class,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 410 85  
## Yes 18 49  
##   
## Accuracy : 0.8167   
## 95% CI : (0.7822, 0.8479)  
## No Information Rate : 0.7616   
## P-Value [Acc > NIR] : 0.0009708   
##   
## Kappa : 0.3907   
##   
## Mcnemar's Test P-Value : 7.864e-11   
##   
## Sensitivity : 0.36567   
## Specificity : 0.95794   
## Pos Pred Value : 0.73134   
## Neg Pred Value : 0.82828   
## Prevalence : 0.23843   
## Detection Rate : 0.08719   
## Detection Prevalence : 0.11922   
## Balanced Accuracy : 0.66181   
##   
## 'Positive' Class : Yes   
##

treepred= predict(final\_fit, test2, type = "class")  
confusionMatrix(treepred$.pred\_class,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 136 31  
## Yes 6 13  
##   
## Accuracy : 0.8011   
## 95% CI : (0.7364, 0.8559)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.1301   
##   
## Kappa : 0.315   
##   
## Mcnemar's Test P-Value : 7.961e-05   
##   
## Sensitivity : 0.29545   
## Specificity : 0.95775   
## Pos Pred Value : 0.68421   
## Neg Pred Value : 0.81437   
## Prevalence : 0.23656   
## Detection Rate : 0.06989   
## Detection Prevalence : 0.10215   
## Balanced Accuracy : 0.62660   
##   
## 'Positive' Class : Yes   
##