## Dana Miller

### Classification Trees Assignment

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)) %>%  
 mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1"))  
  
parole = parole %>%  
 mutate(race = as\_factor(race)) %>%  
 mutate(race = fct\_recode(race, "White" = "1", "Otherwise" = "2"))  
  
parole = parole %>%  
 mutate(state = as\_factor(state)) %>%  
 mutate(state = fct\_recode(state, "Other" = "1", "KY" = "2", "LA" = "3", "VA" = "4"))  
  
parole = parole %>%  
 mutate(crime = as\_factor(crime)) %>%  
 mutate(crime = fct\_recode(crime, "Other" = "1", "Larceny" = "2", "Drug" = "3", "Driving" = "4"))  
  
parole = parole %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "0" = "No", "1" = "Yes"))

## Warning: Problem with `mutate()` input `multiple.offenses`.  
## i Unknown levels in `f`: No, Yes  
## i Input `multiple.offenses` is `fct\_recode(multiple.offenses, `0` = "No", `1` = "Yes")`.

## Warning: Unknown levels in `f`: No, Yes

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

**Task 1**

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

**Task 2**

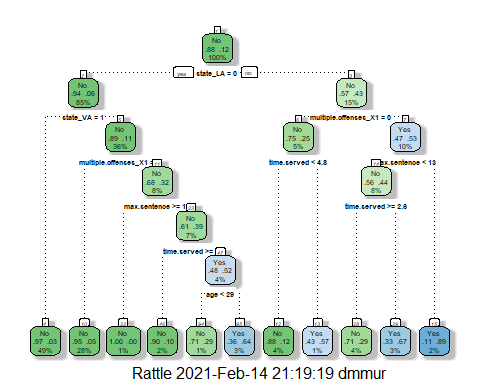
parole\_recipe = recipe(violator ~ ., train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
tree\_model = decision\_tree()%>%  
 set\_engine("rpart", model = TRUE)%>%  
 set\_mode("classification")  
  
parole\_wflow =   
 workflow()%>%  
 add\_model(tree\_model)%>%  
 add\_recipe(parole\_recipe)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

parole\_fit %>%  
 pull\_workflow\_fit()%>%  
 pluck("fit")

## n= 473   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 473 55 No (0.88372093 0.11627907)   
## 2) state\_LA< 0.5 404 25 No (0.93811881 0.06188119)   
## 4) state\_VA>=0.5 232 6 No (0.97413793 0.02586207) \*  
## 5) state\_VA< 0.5 172 19 No (0.88953488 0.11046512)   
## 10) multiple.offenses\_X1< 0.5 134 7 No (0.94776119 0.05223881) \*  
## 11) multiple.offenses\_X1>=0.5 38 12 No (0.68421053 0.31578947)   
## 22) max.sentence>=12.5 7 0 No (1.00000000 0.00000000) \*  
## 23) max.sentence< 12.5 31 12 No (0.61290323 0.38709677)   
## 46) time.served>=5 10 1 No (0.90000000 0.10000000) \*  
## 47) time.served< 5 21 10 Yes (0.47619048 0.52380952)   
## 94) age< 29.15 7 2 No (0.71428571 0.28571429) \*  
## 95) age>=29.15 14 5 Yes (0.35714286 0.64285714) \*  
## 3) state\_LA>=0.5 69 30 No (0.56521739 0.43478261)   
## 6) multiple.offenses\_X1< 0.5 24 6 No (0.75000000 0.25000000)   
## 12) time.served< 4.8 17 2 No (0.88235294 0.11764706) \*  
## 13) time.served>=4.8 7 3 Yes (0.42857143 0.57142857) \*  
## 7) multiple.offenses\_X1>=0.5 45 21 Yes (0.46666667 0.53333333)   
## 14) max.sentence< 12.5 36 16 No (0.55555556 0.44444444)   
## 28) time.served>=2.55 21 6 No (0.71428571 0.28571429) \*  
## 29) time.served< 2.55 15 5 Yes (0.33333333 0.66666667) \*  
## 15) max.sentence>=12.5 9 1 Yes (0.11111111 0.88888889) \*

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

fancyRpartPlot(tree, tweak = 1.3)



**Task 3**

**I would classify the parolee as not violating parole based off the tree. How I reached that conclusion was by determining that state\_LA = 0 is a false statement so I went to the right. Then when looking if multiple offense is X1 = 0 which is false statement therefore I went right again. Looking at max.sentance < 13, which are parole had a max sentence of 10 years, which that being true I went left. When arrived at time.served >= 2.6 that is true since our parolee served 5 years, I went to the left again. The tree brought us to the classification that the parolee described has a 71% chance of not violating parole.**

**Task 4**

parole\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.05454545 0 1.0000000 1.000000 0.1267582  
## 2 0.01818182 4 0.7818182 1.072727 0.1306571  
## 3 0.01454545 5 0.7636364 1.127273 0.1334515  
## 4 0.01000000 10 0.6909091 1.181818 0.1361424

**The optimal cp value is .05. Yes, the optimal cp value is associated with the tree above beacuse the cp value has a large enough improvement between the other values.**

**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  
 )

##   
## 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'

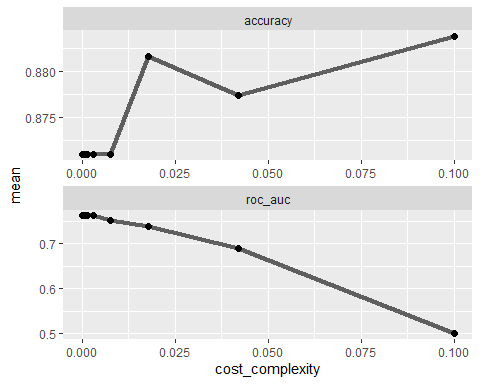
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## 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 [378/95]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [378/95]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [378/95]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [379/94]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [379/94]> 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**

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

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

**Looking at the graphs above the best cp value seems to be about 0.015 to yield the optimal accuracy of .1**

**Task 7**

final\_wf =   
 parole\_wflow2 %>%  
 finalize\_workflow(best\_tree)

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

**Task 8**

trainpredrf = predict(final\_fit, train)  
#head(trainpredrf)

confusionMatrix(trainpredrf$.pred\_class, train$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 418 55  
## Yes 0 0  
##   
## Accuracy : 0.8837   
## 95% CI : (0.8513, 0.9112)  
## No Information Rate : 0.8837   
## P-Value [Acc > NIR] : 0.5358   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 3.305e-13   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8837   
## Prevalence : 0.1163   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

**The accuracy of the root is 88%.**

**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"))

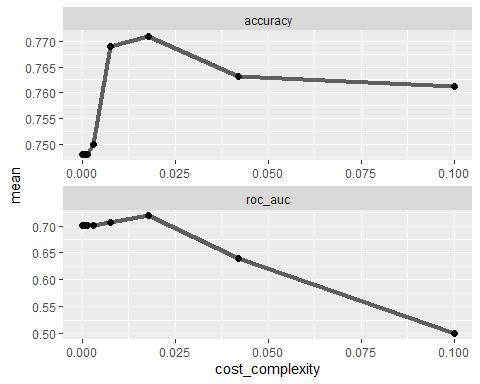
set.seed(1234)  
blood\_split = initial\_split(blood, prop = 0.70, strata = DonatedMarch)  
train2 = training(blood\_split)  
test2 = testing(blood\_split)

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

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

## # 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\_blood\_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 graphs above show a cp value of about 0.015 to be optimal in order to maximize accuracy.**

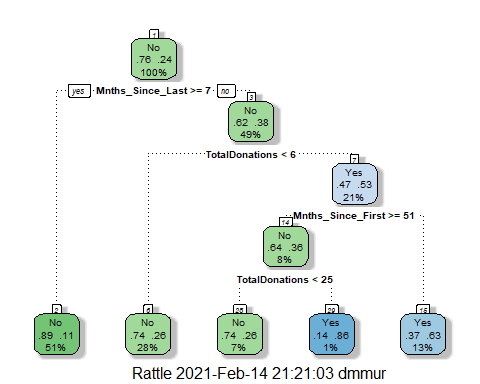
best\_blood\_tree =tree\_blood\_res %>%  
 select\_best("accuracy")  
  
best\_blood\_tree

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

**Task 10**

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

final\_blood\_fit = fit(final\_blood\_wf, train2)  
  
blood\_tree = final\_blood\_fit %>%  
 pull\_workflow\_fit()%>%  
 pluck("fit")  
  
fancyRpartPlot(blood\_tree)



**Task 11**

train2predrf = predict(final\_blood\_fit, train2)  
#head(train2predrf)

confusionMatrix(train2predrf$.pred\_class, train2$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   
##

test2predrf = predict(final\_blood\_fit, test2)  
#head(test2prefrf)

confusionMatrix(test2predrf$.pred\_class, test2$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   
##

**The training set yields about an 81% accuracy while testing set is about 78% accurate. These have very similar accuracies, therefore the model would perform well on data it hasn’t seen before.**