# Module 5 - Assignment 1 Classification Trees

## Samantha Carter

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? Describe how you “walk through” the classification tree to arrive at your answer.

1. Is the individual not from Louisiana, no so you go right.
2. If the parolee is white then he doesn’t violate parole. Which is in keeping with the systemic racism we know exists in our prison and parole system.
3. If the parolee is not white, then did he serve at least 3.9 year, yes so you go to the left
4. Is the individual under 30, no so you go right. Which means he does violate parole.

Task 4: Examine the complexity parameter (cp) values tried by R. Which cp value is optimal (recall that the optimal cp corresponds to the minimized “xerror” value)? 0.03086420  
Is the tree from Task 2 associated with this optimal cp? no

Task 6: Which cp value yields the “optimal” accuracy value? 0.04216965

Task 8: What is the accuracy of the “root” that you generated in Task 7? 0.8854 Take your time and think about how to determine this value.Since this is a root and all the observations are associated with the majority class then this would be the same as the naive model so we can use the No Information Rate

Task 9: What cp value appears to be“optimal” to maximize accuracy? 0.003162278

Task 11: Determine the accuracy of the tree from Task 10 on the training set 0.82 and testing sets 0.78. Comment on the tree’s performance on these sets. There isn’t too much of a dip between the training and testing set less than 0.04 difference in accuracy I would say this is a pretty good model. Given that the naive set has an accuracy of 0.75 it not a huge difference, but if you consider that every percentage point is more blood donated in a month I would use this model as a good predictor model, but not great.

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" )) %>%  
 mutate(race = as\_factor(race)) %>%   
 mutate(race = fct\_recode(race, "white" = "1", "nonwhite" = "2" )) %>%  
 mutate(state = as\_factor(state))%>%  
 mutate(state = fct\_recode(state, "OtherState" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4" )) %>%  
 mutate(crime = as\_factor(crime))%>%  
 mutate(crime = fct\_recode(crime, "OtherCrime" = "1", "larceny" = "2", "drugs" = "3", "driving" ="4" )) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses))%>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "No" = "0", "Yes" = "1" )) %>%  
 mutate(violator = as\_factor(violator))%>%  
mutate(violator = fct\_recode(violator, "No" = "0", "Yes" = "1" ))   
str(parole)

## spec\_tbl\_df[,9] [675 x 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ male : Factor w/ 2 levels "female","male": 2 1 2 2 2 2 2 1 1 2 ...  
## $ race : Factor w/ 2 levels "white","nonwhite": 1 1 2 1 2 2 1 1 1 2 ...  
## $ age : num [1:675] 33.2 39.7 29.5 22.4 21.6 46.7 31 24.6 32.6 29.1 ...  
## $ state : Factor w/ 4 levels "OtherState","Kentucky",..: 1 1 1 1 1 1 1 1 1 1 ...  
## $ time.served : num [1:675] 5.5 5.4 5.6 5.7 5.4 6 6 4.8 4.5 4.7 ...  
## $ max.sentence : num [1:675] 18 12 12 18 12 18 18 12 13 12 ...  
## $ multiple.offenses: Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "OtherCrime","larceny",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "No","Yes": 1 1 1 1 1 1 1 1 1 1 ...  
## - attr(\*, "spec")=  
## .. 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()  
## .. )

summary(parole)

## male race age state time.served   
## female:130 white :389 Min. :18.40 OtherState:143 Min. :0.000   
## male :545 nonwhite:286 1st Qu.:25.35 Kentucky :120 1st Qu.:3.250   
## Median :33.70 Louisiana : 82 Median :4.400   
## Mean :34.51 Virginia :330 Mean :4.198   
## 3rd Qu.:42.55 3rd Qu.:5.200   
## Max. :67.00 Max. :6.000   
## max.sentence multiple.offenses crime violator   
## Min. : 1.00 No :313 OtherCrime:315 No :597   
## 1st Qu.:12.00 Yes:362 larceny :106 Yes: 78   
## Median :12.00 drugs :153   
## Mean :13.06 driving :101   
## 3rd Qu.:15.00   
## Max. :18.00

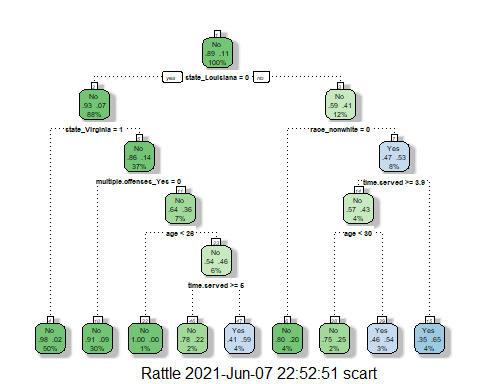
#### Task 1

set.seed(12345, kind = "Mersenne-Twister", normal.kind = "Inversion")  
parole\_split = initial\_split(parole, prop = 0.70, strata = "violator" )  
train = training(parole\_split)  
test = testing(parole\_split)

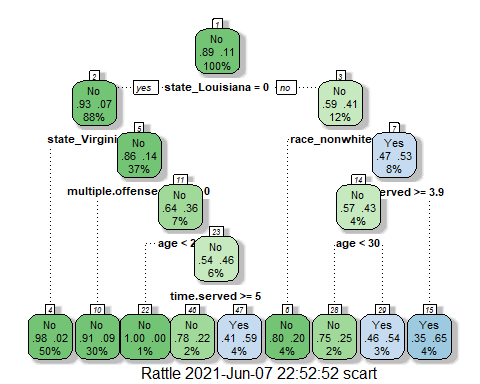
#### Task 2

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

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



fancyRpartPlot(tree, tweak=1.5)



#### 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

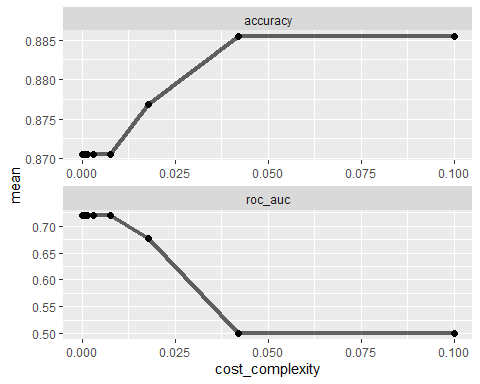
#### Task 5

set.seed(123, kind = "Mersenne-Twister", normal.kind = "Inversion")  
folds = vfold\_cv(train, v = 5)

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)  
  
set.seed(123)  
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[,5] [50 x 5]> <tibble[,1] [0 x 1]>  
## 2 <split [377/94]> Fold2 <tibble[,5] [50 x 5]> <tibble[,1] [0 x 1]>  
## 3 <split [377/94]> Fold3 <tibble[,5] [50 x 5]> <tibble[,1] [0 x 1]>  
## 4 <split [377/94]> Fold4 <tibble[,5] [50 x 5]> <tibble[,1] [0 x 1]>  
## 5 <split [377/94]> Fold5 <tibble[,5] [50 x 5]> <tibble[,1] [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 7

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

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

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

treepred\_train = predict(parole\_fit, train, type = "class")  
#head(treepred\_train)

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 398 26  
## Yes 19 28  
##   
## Accuracy : 0.9045   
## 95% CI : (0.8742, 0.9295)  
## No Information Rate : 0.8854   
## P-Value [Acc > NIR] : 0.1075   
##   
## Kappa : 0.5012   
##   
## Mcnemar's Test P-Value : 0.3711   
##   
## Sensitivity : 0.51852   
## Specificity : 0.95444   
## Pos Pred Value : 0.59574   
## Neg Pred Value : 0.93868   
## Prevalence : 0.11465   
## Detection Rate : 0.05945   
## Detection Prevalence : 0.09979   
## Balanced Accuracy : 0.73648   
##   
## 'Positive' Class : Yes   
##

#### 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[,5] [748 x 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()  
## .. )

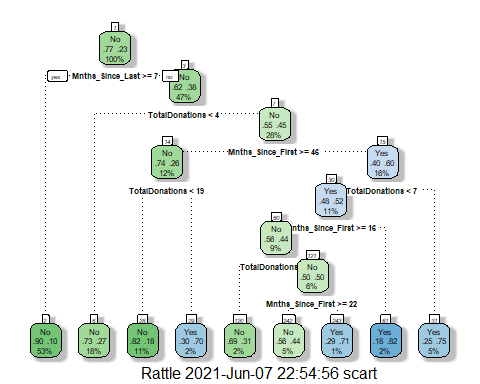
#summary(Blood)

#### Task 9-2

set.seed(1234, kind = "Mersenne-Twister", normal.kind = "Inversion")  
Blood\_split = initial\_split(Blood, prop = 0.70 )  
train2 = training(Blood\_split)  
test2 = testing(Blood\_split)

Blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
Blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(Blood\_recipe)  
set.seed(1234)  
Blood\_fit = fit(Blood\_wflow, train2)

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



Blood\_fit$fit$fit$fit$cptable

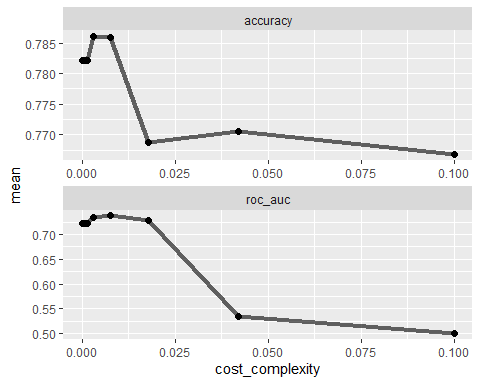
## CP nsplit rel error xerror xstd  
## 1 0.04371585 0 1.0000000 1.0000000 0.07927594  
## 2 0.03278689 3 0.8688525 0.9508197 0.07787813  
## 3 0.02049180 4 0.8360656 0.9918033 0.07904876  
## 4 0.01229508 6 0.7950820 0.9672131 0.07835340  
## 5 0.01000000 8 0.7704918 0.9426230 0.07763693

set.seed(1234, kind = "Mersenne-Twister", normal.kind = "Inversion")  
folds = vfold\_cv(train2, v = 5)

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)  
set.seed(1234)  
tree\_res2 =   
 Blood\_wflow %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid  
 )  
  
tree\_res2

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



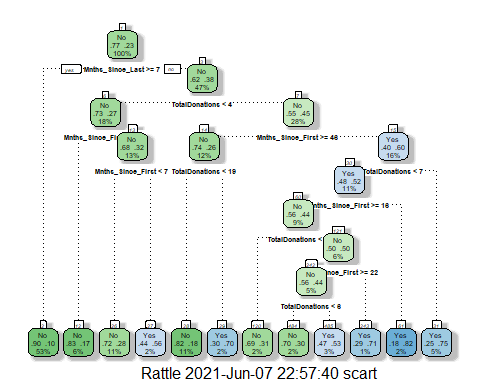
best\_tree2 = tree\_res2 %>%  
 select\_best("accuracy")  
  
best\_tree2

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00316 Preprocessor1\_Model21

#### Task 10

final\_wf2 =   
 Blood\_wflow %>%   
 finalize\_workflow(best\_tree2)

final\_fit2 = fit(final\_wf2, train2)  
  
tree = final\_fit2 %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 1.5)



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

#### Task 11

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 387 80  
## Yes 14 42  
##   
## Accuracy : 0.8203   
## 95% CI : (0.7846, 0.8522)  
## No Information Rate : 0.7667   
## P-Value [Acc > NIR] : 0.001765   
##   
## Kappa : 0.3811   
##   
## Mcnemar's Test P-Value : 2.025e-11   
##   
## Sensitivity : 0.34426   
## Specificity : 0.96509   
## Pos Pred Value : 0.75000   
## Neg Pred Value : 0.82869   
## Prevalence : 0.23327   
## Detection Rate : 0.08031   
## Detection Prevalence : 0.10707   
## Balanced Accuracy : 0.65467   
##   
## 'Positive' Class : Yes   
##

treepred\_test2 = predict(Blood\_fit, test2, type = "class")  
#head(treepred\_test2)

confusionMatrix(treepred\_test2$.pred\_class,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 156 36  
## Yes 13 20  
##   
## Accuracy : 0.7822   
## 95% CI : (0.7225, 0.8343)  
## No Information Rate : 0.7511   
## P-Value [Acc > NIR] : 0.158044   
##   
## Kappa : 0.3248   
##   
## Mcnemar's Test P-Value : 0.001673   
##   
## Sensitivity : 0.35714   
## Specificity : 0.92308   
## Pos Pred Value : 0.60606   
## Neg Pred Value : 0.81250   
## Prevalence : 0.24889   
## Detection Rate : 0.08889   
## Detection Prevalence : 0.14667   
## Balanced Accuracy : 0.64011   
##   
## 'Positive' Class : Yes   
##