# Module 4 Assignment 1

## Katherine Schumann

### Classification Trees

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, "Male" = "1", "Female" = "0"))  
  
  
parole = parole%>% mutate(race = as\_factor(race))%>% mutate(race = fct\_recode(race, "White" = "1", "Other" = "2"))  
  
parole = parole%>% mutate(state = as\_factor(state))%>% mutate(state = fct\_recode(state, "Other" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4"))  
  
parole = parole%>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses, "Yes" = "1", "No" = "0"))  
  
parole = parole%>% mutate(crime = as\_factor(crime)) %>% mutate(crime = fct\_recode(crime, "Other" = "1", "Larceny" = "2", "Drug-related" = "3", "Driving-related" = "4"))  
  
parole = parole %>% mutate(violator = as.character.numeric\_version(violator)) %>% mutate(violator = fct\_recode(violator, "Yes" = "1", "No" = "0"))

### Task 1

Split the data (training and testing)

set.seed(12345)  
parole\_split = initial\_split(parole, prop = 0.7, strata = violator) #70% in training  
train = training(parole\_split)  
test = testing(parole\_split)

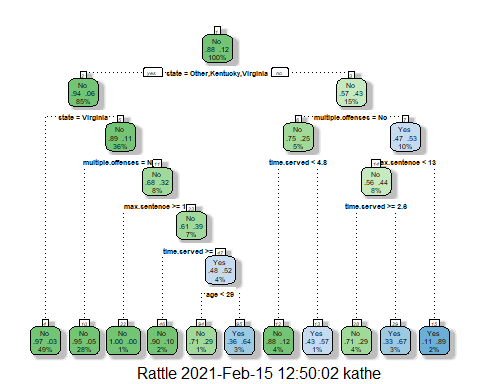
### Task 2

Create regression tree

parole\_recipe = recipe(violator ~., train)  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification") #notice different mode here for a regression tree  
  
parole\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(parole\_recipe)  
  
parole\_fit = fit(parole\_wflow, train)

Plot the tree

tree = parole\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak=1.25)



### Task 3

First, the state he is from is not Kentucky, Virginia, or Other, so I would go to the no-decision. Then the parolee has committed multiple offenses, so I would select no classification. With time served over 5 years, I would select the No, which has a probability of 88% not violating the parole. This means we would classify this parolee as not violating parole.

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 1.090909 0.1316005  
## 3 0.01454545 5 0.7636364 1.090909 0.1316005  
## 4 0.01000000 10 0.6909091 1.127273 0.1334515

### Task 4

No, the tree is not associated with the optimal value. Since the classification tree that R used has 10 splits, we can see that this has one of the higher xerrors from the above cp table. The optimal cp value is .052, which has a xerror of 1, but a nsplit of 0, meaning no splits or a naive model.

### Task 5

Create our folds

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

parole\_recipe1 = recipe(violator ~., train)  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification") #notice different mode here for a regression tree  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25) #try 25 sensible values for cp  
parole\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(parole\_recipe1)  
  
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'

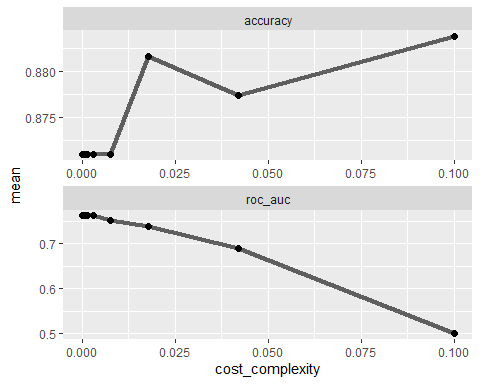
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## 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 [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

The best tree yields a cp optimal, accurate value is .1.

### 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

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

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred$.pred\_class,train$violator,positive="Yes") #predictions first then actual

## 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   
##

Since the optimal accuracy value is the naive model, we can use either the accuracy of 88.37% or the no information rate.

### 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"))

### Task 10

Split the data (training2 and testing2)

set.seed(1234)  
Blood\_split = initial\_split(Blood, prop = 0.7, strata = DonatedMarch) #70% in training  
train2 = training(Blood\_split)  
test2 = testing(Blood\_split)

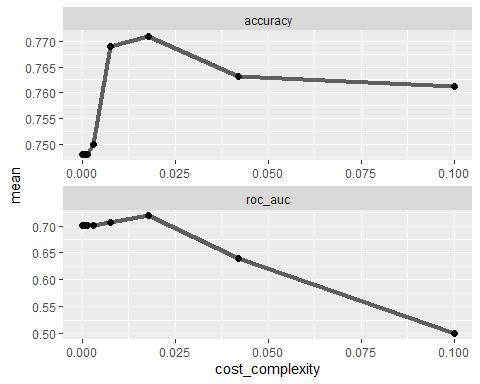
Create our folds

set.seed(1234)  
folds2 = vfold\_cv(train2, v = 5)

Blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree1\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification") #notice different mode here for a regression tree  
  
tree1\_grid = grid\_regular(cost\_complexity(),  
 levels = 25) #try 25 sensible values for cp  
Blood\_wflow =   
 workflow() %>%   
 add\_model(tree1\_model) %>%   
 add\_recipe(Blood\_recipe)  
  
tree1\_res =   
 Blood\_wflow %>%   
 tune\_grid(  
 resamples = folds2,  
 grid = tree1\_grid  
 )  
  
tree1\_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]>

tree1\_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 hits an optimal around approx. .017, then it starts to go down and balance out.

best\_tree1 = tree1\_res %>%  
 select\_best("accuracy")  
  
best\_tree1

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

### Task 11

final1\_wf =   
 Blood\_wflow %>%   
 finalize\_workflow(best\_tree1)

final1\_fit = fit(final1\_wf, train2)  
  
tree1 = final1\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree1, tweak = 1.5)

### Task 12

Predictions on training set

tree1pred = predict(final1\_fit, train2, type = "class")  
head(tree1pred)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(tree1pred$.pred\_class,train2$DonatedMarch,positive="Yes") #predictions first then actual

## 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   
##

Predictions on testing set

tree1pred\_test = predict(final1\_fit, test2, type = "class")  
head(tree1pred\_test)

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(tree1pred\_test$.pred\_class,test2$DonatedMarch,positive="Yes") #predictions first then actual

## 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 model ran exceptionally well on the training set with an accuracy of 80.53%, a p-value of 0.009, and the naive model 76.15% accurate. However, the model did not do so well on the testing set. The accuracy was at 78.12%. The naive model of 76.34% with a p-value of .29, meaning that the model's difference to the naive model was insignificant. It did yield a two percent increase in accuracy but not quite enough to make it a significantly better predictor.