# BAN502 Module 4 Assignment 1

## Kristian Marlowe

### Classification Trees

First, the correct libraries will be loaded/added. (The code was excluded from the markdown document)

Next, the parole data will be read in.

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()  
## )

#str(parole)  
#summary(parole)

Next, the variables will be mutated.

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", "otherwise" = "2" )) %>%  
 mutate(state = as\_factor(state)) %>% mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "Any Other State" = "1")) %>%  
 mutate(crime = as\_factor(crime)) %>% mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related" = "3", "driving-related" = "4", "other crime" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% mutate(multiple.offenses = fct\_recode(multiple.offenses, "incarcerated for multiple offenses" = "1", "otherwise" = "0")) %>%  
 mutate(violator=as\_factor(violator)) %>% mutate(violator=fct\_recode(violator, "Violate Parole" = "1", "No Parole Violations" ="0"))  
#str(parole)  
#summary(parole)

## Task 1

In task 1, the data will be split into training and testing sets.

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

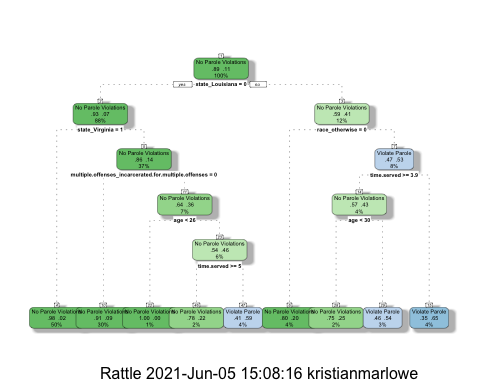
## Task 2

In task 2, a classification tree will be created to predict “violator” in the training set.

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)

fancyRpartPlot(tree, tweak = 1.2)



## Task 3

We will walk through the decision tree for a 40 -year old parolee from Louisiana who served 5 years in prison, with a sentence of 10 years, and committed multiple offenses.

First, we start at the top, with the state\_Louisiana = 0, since the parolee is from Louisiana, we go to the right, (this is where I may have a mac user issue) if the person is ‘white’ then we go to the left. Since the person served more than 4.8 years, we go to the right - then it is suggested they will violate parole.

If the person was ‘race otherwise’ then we would go to the right, then since the max sentence is less than 12 years, we go to the left, then since the parolee is older than 29 years, we go to the right, and see they it is suggested they will violate parole.

## 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 1.259259 0.1412547  
## 3 0.01388889 4 0.8888889 1.314815 0.1437987  
## 4 0.01000000 8 0.8333333 1.351852 0.1454450

The cp value that is optimal is the tree with 5 splits with a cp of 0.01000000. This produced the smallest cross validation error of 1.166667. This was the decision tree that was viewed for me.

## Task5

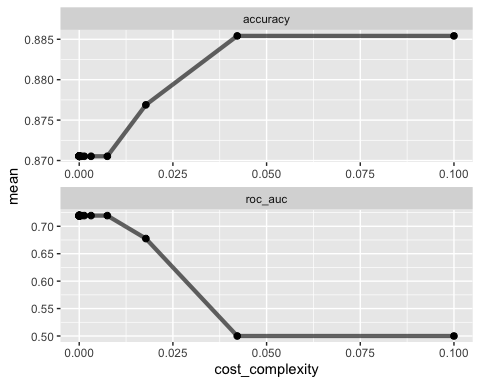
In task 5, the goal is to use a tuning grid to try 25 different values for the complexity parameter (cp).

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\_grid2 = grid\_regular(cost\_complexity(),  
 levels = 25)   
  
parole\_wflow2 =   
 workflow() %>%   
 add\_model(tree\_model2) %>%   
 add\_recipe(parole\_recipe2)  
  
tree\_res2 =   
 parole\_wflow2 %>%   
 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 [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\_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 6

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

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

The cp value that yields the ‘optimal’ accuracy value is 0.1.

## 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, tweak = 1.5)

## Task 8

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Parole Violations Violate Parole  
## No Parole Violations 417 54  
## Violate Parole 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 : Violate Parole   
##

treepred\_test = predict(final\_fit, test, type ="class")  
#head(treepred\_test)

confusionMatrix(treepred\_test$.pred\_class,test$violator,positive="Violate Parole")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Parole Violations Violate Parole  
## No Parole Violations 180 24  
## Violate Parole 0 0  
##   
## Accuracy : 0.8824   
## 95% CI : (0.83, 0.9231)  
## No Information Rate : 0.8824   
## P-Value [Acc > NIR] : 0.5541   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 2.668e-06   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8824   
## Prevalence : 0.1176   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Violate Parole  
##

In the training set, we have the same accuracy rate as the no information rate of 0.8854. In the testing set, we have the same accuracy rate as the no information rate of 0.8824. It is interesting to note that in both the training and testing sets, we have no predictions in the ‘violate parole’ row. Since this data is heavily skewed, we should go with the no information rate, especially since the accuracy is no higher than the no information rate in both the testing and training sets. We would say the accuracy would be ~88%.

## Task 9

In task 9, the Blood dataset will be read in and mutated.

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()  
## )

#str(Blood)  
#summary(Blood)

Blood = Blood %>% mutate(DonatedMarch = as\_factor(DonatedMarch)) %>%  
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1" ))   
   
#str(Blood)  
#summary(Blood)

## Task 10

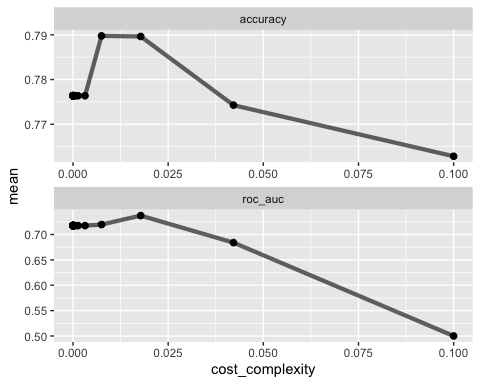
In task 10 the Blood dataset will be split into training and testing sets.

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\_model3 = 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\_model3) %>%   
 add\_recipe(blood\_recipe)  
  
tree\_res2 =   
 blood\_wflow %>%   
 tune\_grid(  
 resamples = folds2,  
 grid = tree\_grid2  
 )  
  
#tree\_res2

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 0.00750 Preprocessor1\_Model22

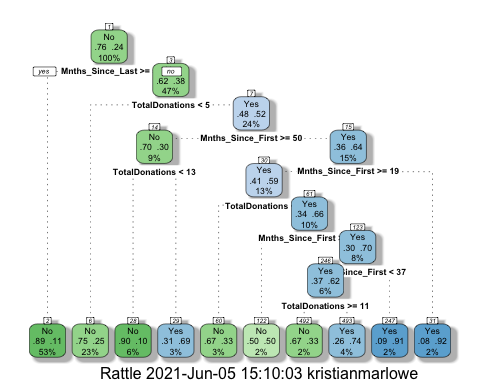
The optimal cp value seems to range from 0.0005 to 0.015. Upon further coding, the actual optimal cp value is 0.007498942.

## Task 11

In this task, the goal is to plot the tree with the optimal cp from the previous task. The optimal cp was 0.007498942.

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

final\_fit2 = fit(final\_wf2, train2)  
  
tree2 = final\_fit2 %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree2, tweak = 1.5)



## Task 12

This task will determine the accuracy of the tree from the previous task on the training and testing sets.

treepred2 = predict(final\_fit2, train2, type = "class")  
#head(treepred2)

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   
##

treepred\_test2 = predict(final\_fit2, test2, type = "class")  
#head(treepred\_test2)

confusionMatrix(treepred\_test2$.pred\_class,test2$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   
##

For the training set, the accuracy is better than the no information rate, which we want to see. The training set has an accuracy of 0.8317 and a no information rate of 0.7629. However, in the testing set, we do not see this same information. Firstly, the accuracy rates are fairly different values. The accuracy rate here is 0.7511. Secondly, in the testing set, this model is not better than the no information rate of 0.76. This could indicate a not good model on the testing set. We should want to see both the testing and training sets have similar accuracy values and both accuracy values should be an improvement on the no information rate.