Date: 06/07/2021

BAN 502-802

Course: Predictive Analytics

Author: Gregory Carmichael

Assignment: Module 4 Classification Trees

Load libraries

Read-in the Dataset

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()  
## )

Factor conversion. Convert and recode.

parole = parole %>% mutate(male = as\_factor(male)) %>% # 1 if the parolee is male, 0 if female  
 mutate(male = fct\_recode(male, "Female" = "0", "Male" = "1"))  
  
parole = parole %>% mutate(race = as\_factor(race)) %>% # 1 if the parolee is white, 2 otherwise  
 mutate(race = fct\_recode(race, "white" = "1", "otherwise" = "2"))  
  
parole = parole %>% mutate(state = as\_factor(state)) %>% # codes for the parolee’s state.  
 mutate(state = fct\_recode(state, "Other State" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4"))  
  
parole = parole %>% mutate(crime = as\_factor(crime)) %>% # code for the parolee’s main crime leading to incarceration.  
 mutate(crime = fct\_recode(crime, "Other Crime" = "1", "larceny" = "2", "drug-related" = "3", "drivig related" = "4"))  
  
parole = parole %>% mutate(multiple.offenses = as\_factor(multiple.offenses)) %>% # 1 if the parolee was incarcerated for multiple offenses, 0 otherwise  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "otherwise" = "0", "incarcerated" = "1"))  
  
parole = parole %>% mutate(violator = as\_factor(violator)) %>% # 1 if the parolee violated the parole, and 0 if the parolee completed the parole without violation.  
 mutate(violator = fct\_recode(violator, "Completed parole without violation" = "0", "violated parole" = "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","otherwise": 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 "Other State",..: 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 "otherwise","incarcerated": 1 1 1 1 1 1 1 1 1 1 ...  
## $ crime : Factor w/ 4 levels "Other Crime",..: 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : Factor w/ 2 levels "Completed parole without violation",..: 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()  
## .. )

##TASK 1##

Split the data in ‘training’ and ‘testing’ sets. Training set will have 70% of the data.

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

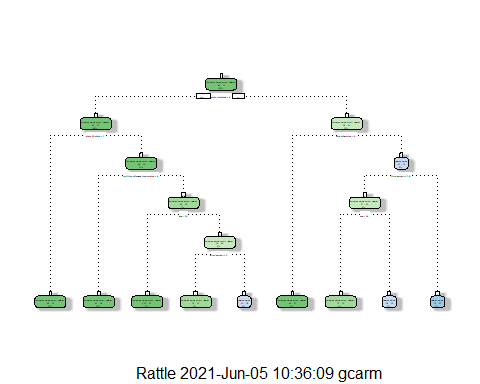
##TASK 2##

Build the Classification Tree to predict “violator”.

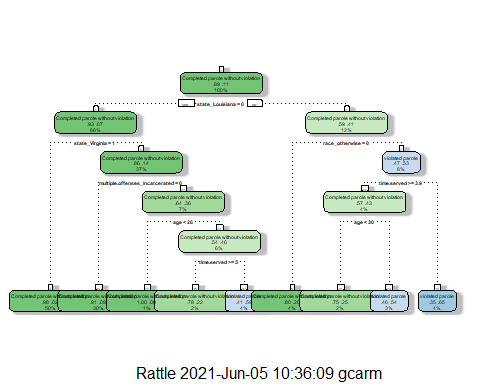
parole\_recipe = recipe(violator ~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
parole\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(parole\_recipe)  
  
parole\_fit = fit(parole\_wflow, train)

Plot the tree.

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

 Replot the tree by using tweak = 1.5

fancyRpartPlot(tree, tweak=2)



##TASK 3##

Classification Tree walk-Through

Top of tree is asking if ‘state\_louisiana = 0’, which implies that someone is not from the state of Louisiana. Therefore, I go right. At this point, the parolee is still considered ‘completed parole without violation’ ‘race\_otherwise’ implies that someone is white, therefore i go right again. ‘Time.served >= 3.9’, which in this case, this person served 5 years. Therefore, I go left ‘age < 30’, which in this case is not the case, so I go right.

Conclusion: 3 % of our data is in the light blue node, second from the right. This parole is a white male where 46% are classified as violating their parole, while the remaining 54% are classified as not violating their parole.

##TASK 4##

Look at the “rpart” complexity parameter “cp”.

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

After going through 8 spilts, it tried a different values, starting at 0.03 to the default .010 We see that 1.00 is the smallest validated error. The tree from task 2 is not associated with this optimal CP because smaller values for CP bring about the larger and more complex trees. This is a large CP compared to those in our lecture, yet our tree is 8 levels.

##TASK 5##

#to tune cross complexity itself, we create our fold. Create our folds

set.seed(123) #using k-fold cross validation  
folds = vfold\_cv(train, v = 5) #doing it on the training set - we are going to do 5 fold cross validation on the training set

#Differences in the Model and the resamples (Tuning grid) this one is running 25 CP values doing 5 fold cross validaton # runnin gmultiple models

parole\_recipe = recipe(violator ~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>% #Here, we are tuning cost complexity, tune() - allows R to try different values  
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25) #try 25 sensible values for cp  
  
parole\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(parole\_recipe)  
  
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]>

##TASK 6##

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

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

Answer: A CP value of 0.042 yields the optimal accuracy value.

##TASK 7##

#Finalize the Model first - because we choose the complexity parameter

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

#we then fit the model, then plot the tree

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

NOTE: I got the error as stated in the requirements

##TASK 8##

Predictions on training set

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

## # A tibble: 6 x 1  
## .pred\_class   
## <fct>   
## 1 Completed parole without violation  
## 2 Completed parole without violation  
## 3 Completed parole without violation  
## 4 Completed parole without violation  
## 5 Completed parole without violation  
## 6 Completed parole without violation

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred$.pred\_class,train$violator,positive="Completed parole without violation") #predictions first then actual

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction Completed parole without violation  
## Completed parole without violation 417  
## violated parole 0  
## Reference  
## Prediction violated parole  
## Completed parole without violation 54  
## violated parole 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 : 1.0000   
## Specificity : 0.0000   
## Pos Pred Value : 0.8854   
## Neg Pred Value : NaN   
## Prevalence : 0.8854   
## Detection Rate : 0.8854   
## Detection Prevalence : 1.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Completed parole without violation  
##

Answer: We are 88.5% accurate on the training set, which is pretty good.

##TASK 9##

Read-in the Dataset

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()  
## )

Factor conversion. Convert and recode.

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

Split the data in ‘training’ and ‘testing’ sets. Training set will have 70% of the data.

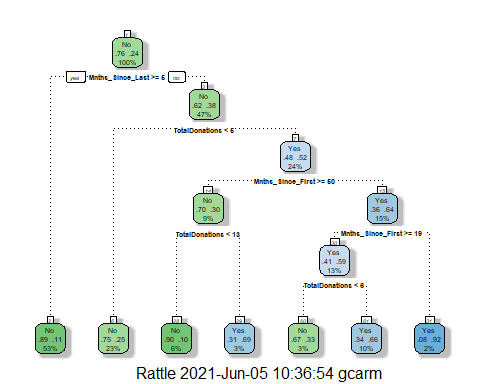
set.seed(1234)  
blood\_split = initial\_split(blood, prop = 0.70, strata = DonatedMarch)##Tidymodels function for splitting  
train2 = training(blood\_split)  
test2 = testing(blood\_split)

Build the Classification Tree to predict “violator”.

blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(blood\_recipe)  
  
blood\_fit = fit(blood\_wflow, train2)

Plot the tree.

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



#to tune cross complexity itself, we create our fold. Create our folds

set.seed(1234) #using k-fold cross validation  
folds = vfold\_cv(train2, v = 5) #doing it on the training set - we are going to do 5 fold cross validation on the training set

#Differences in the Model and the resamples (Tuning grid) this one is running 25 CP values doing 5 fold cross validaton # runnin gmultiple models

blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>% #Here, we are tuning cost complexity, tune() - allows R to try different values  
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid = grid\_regular(cost\_complexity(),  
 levels = 25) #try 25 sensible values for cp  
  
blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(blood\_recipe)  
  
tree\_res =   
 blood\_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 [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]>

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

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00750 Preprocessor1\_Model22

Answer: in this case, a CP value of 0.0075 yields the optimal accuracy value.

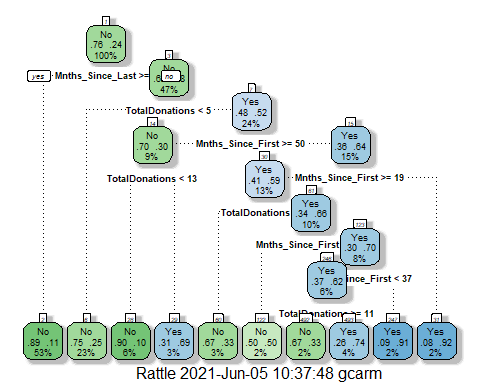
##TASK 10##

#Finalize the Model first - because we choose the complexity parameter

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

#we then fit the model, then plot the tree

final\_fit = fit(final\_wf, train2)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree, tweak = 2)



##TASK 11##

Predictions on training set

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

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

Caret confusion matrix and accuracy, etc. calcs

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

## 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   
##

#We are 83% accurate on the training set, which is pretty good.

Predictions on testing set

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

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

Caret confusion matrix and accuracy, etc. calcs

confusionMatrix(treepred\_test2$.pred\_class,test2$DonatedMarch,positive="Yes") #predictions first then actual

## 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   
##

#On the testing set, we are 75% accurate, which is good. #The naive is .656, the accuracy on both the training and testing is better that the naive (no information rate) which is very good.