M4 Assign 1\_Trees

Corinne Walker

6/2/2021

library(readr)  
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 %>%   
 rename(gender = male) %>%   
 mutate(gender = as\_factor(gender)) %>%   
 mutate(gender = fct\_recode(gender, "female" = "0", "male" = "1")) %>%  
 mutate(race = as\_factor(race)) %>%   
 mutate(race = fct\_recode(race, "not white" = "2", "white" = "1")) %>%  
 mutate(state = as\_factor(state)) %>%   
 mutate(state = fct\_recode(state, "other state" = "1", "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4")) %>%  
 mutate(crime = as\_factor(crime)) %>%   
 mutate(crime = fct\_recode(crime, "other crime" = "1", "larceny" = "2", "drug" = "3", "driving" = "4")) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "single" = "0", "multiple" = "1")) %>%  
 mutate(violator = as\_factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "no violation" = "0", "violated" = "1"))

summary(parole)

## gender race age state time.served   
## female:130 white :389 Min. :18.40 other state:143 Min. :0.000   
## male :545 not white: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 single :313 other crime:315 no violation:597   
## 1st Qu.:12.00 multiple:362 larceny :106 violated : 78   
## Median :12.00 drug :153   
## Mean :13.06 driving :101   
## 3rd Qu.:15.00   
## Max. :18.00

## Task 1

Split the data (training and testing). 70% of the data to training. Stratified the random split by the response variable “violator”.

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

## Task 2

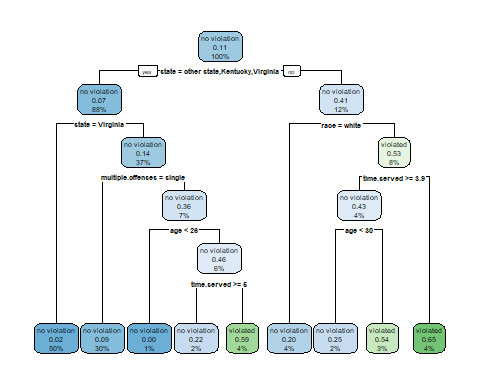
Create a classification tree to predict “violator” in the training set (using all of the other variables as predictors). Plot the tree.

parole\_recipe = recipe(violator ~ gender + race + age + state + time.served + max.sentence + multiple.offenses + crime, train)  
  
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)

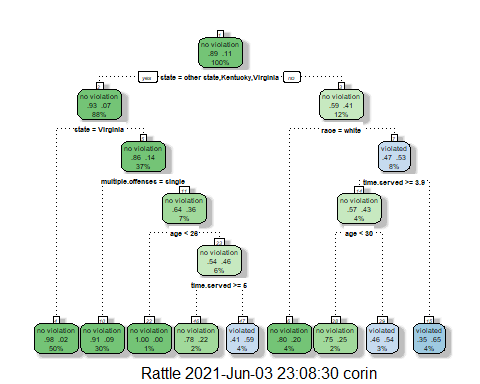
#tree's fit  
parole\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")

## n= 471   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 471 54 no violation (0.88535032 0.11464968)   
## 2) state=other state,Kentucky,Virginia 413 30 no violation (0.92736077 0.07263923)   
## 4) state=Virginia 237 5 no violation (0.97890295 0.02109705) \*  
## 5) state=other state,Kentucky 176 25 no violation (0.85795455 0.14204545)   
## 10) multiple.offenses=single 143 13 no violation (0.90909091 0.09090909) \*  
## 11) multiple.offenses=multiple 33 12 no violation (0.63636364 0.36363636)   
## 22) age< 25.85 7 0 no violation (1.00000000 0.00000000) \*  
## 23) age>=25.85 26 12 no violation (0.53846154 0.46153846)   
## 46) time.served>=4.95 9 2 no violation (0.77777778 0.22222222) \*  
## 47) time.served< 4.95 17 7 violated (0.41176471 0.58823529) \*  
## 3) state=Louisiana 58 24 no violation (0.58620690 0.41379310)   
## 6) race=white 20 4 no violation (0.80000000 0.20000000) \*  
## 7) race=not white 38 18 violated (0.47368421 0.52631579)   
## 14) time.served>=3.85 21 9 no violation (0.57142857 0.42857143)   
## 28) age< 29.75 8 2 no violation (0.75000000 0.25000000) \*  
## 29) age>=29.75 13 6 violated (0.46153846 0.53846154) \*  
## 15) time.served< 3.85 17 6 violated (0.35294118 0.64705882) \*

#extract the tree's fit from the fit object  
tree = parole\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
#plot the tree  
rpart.plot(tree)



#alternative  
fancyRpartPlot(tree)



## 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.

I arrived at my answer by selecting:  
-“no” for the state question

-race=white as it was the only option

-greater than 3.5 years for time.served

-age greater than 30

The parolee would fall into the violation category at 3%.

## 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)?  
Is the tree from Task 2 associated with this optimal cp?

“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 1.240741 0.1403860  
## 3 0.01388889 4 0.8888889 1.277778 0.1421129  
## 4 0.01000000 8 0.8333333 1.314815 0.1437987

The optimal cp value is 0.01851852 with an xerror of 1.240741 which would result in 3 splits. The tree from task 2 did not use the optimal cp value and that resulted in 8 splits in the tree.

## Task 5

Use a tuning grid (as we did in the Titanic problem) to try 25 different values for the complexity parameter (cp). R will select reasonable values. Use 5-fold k-fold cross-validation (don’t forget to set up your folds). Use a seed of 123 when setting up your folds.

Create folds

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

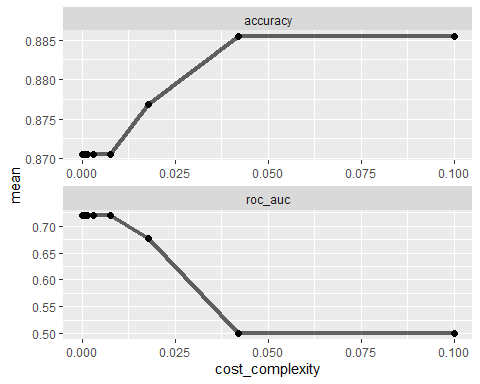
parole\_recipe = recipe(violator ~ gender + race + age + state + time.served + max.sentence + multiple.offenses + crime, train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget 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

Which cp value yields the “optimal” accuracy value?

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)



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

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

A cp value of 0.04216965 yields the optimal accuracy.

## Task 7

Try to plot the tree that corresponds to the cp value from Task 6. Don’t forget to finalize your workflow and generate your final fit before trying to plot.

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

What is the accuracy of the “root” that you generated in Task 7? Take your time and think about how to determine this value.

Develop predictions on the training set

train\_preds = predict(parole\_fit, train)  
head(train\_preds) #see first six predictions to verify that predictions are in correct form

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

Examine performance metrics on the training set.

parole\_fit %>% predict(train) %>% bind\_cols(train) %>% metrics(truth = violator, estimate = .pred\_class)

## # A tibble: 2 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.904  
## 2 kap binary 0.501

Develop predictions on the testing set

test\_preds = predict(parole\_fit, test)  
head(test\_preds) #verify that predictions are in correct form

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

Examine performance metrics on the testing set.

parole\_fit %>% predict(test) %>% bind\_cols(test) %>% metrics(truth = violator, estimate = .pred\_class)

## # A tibble: 2 x 3  
## .metric .estimator .estimate  
## <chr> <chr> <dbl>  
## 1 accuracy binary 0.887  
## 2 kap binary 0.403

The training set has a better accuracy of 0.9044586 compared to the testing set which has an accuracy of 0.8872549.

## Task 9

Read in the “Blood.csv” dataset.

library(readr)  
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()  
## )

summary(Blood)

## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :34.28   
## 3rd Qu.:14.000 3rd Qu.: 7.000 3rd Qu.: 1750 3rd Qu.:50.00   
## Max. :74.000 Max. :50.000 Max. :12500 Max. :98.00   
## DonatedMarch   
## Min. :0.000   
## 1st Qu.:0.000   
## Median :0.000   
## Mean :0.238   
## 3rd Qu.:0.000   
## Max. :1.000

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

Split the dataset into training (70%) and testing (30%) sets. You may wish to name your training and testing sets “train2” and “test2” as to not confuse them with the parole datsets. Use set.seed of 1234. Then develop a classification tree on the training set to predict “DonatedMarch”. As you did in Task 5, let R try 25 different values of cp.  Don’t forget to create new folds on the new training dataset. Use a seed of 1234 for the folds.

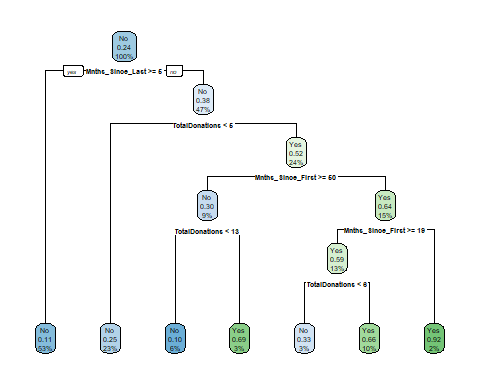
set.seed(1234)  
Blood\_split = initial\_split(Blood, prop = 0.70, strata = DonatedMarch)  
train2 = training(Blood\_split)  
test2 = testing(Blood\_split)

Blood\_recipe <- recipe(DonatedMarch ~., train2)  
  
tree\_model2 = decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
Blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model2) %>%   
 add\_recipe(Blood\_recipe)  
  
Blood\_fit = fit(Blood\_wflow, train2)

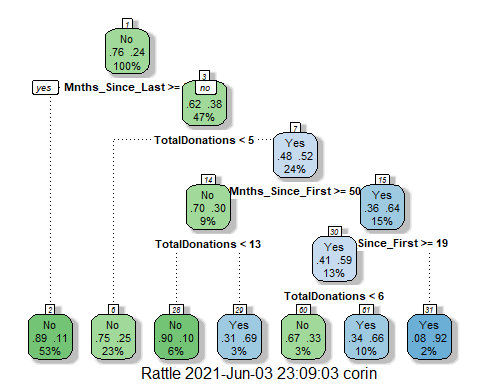
#tree's fit  
Blood\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")

## n= 523   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 523 124 No (0.76290631 0.23709369)   
## 2) Mnths\_Since\_Last>=4.5 275 29 No (0.89454545 0.10545455) \*  
## 3) Mnths\_Since\_Last< 4.5 248 95 No (0.61693548 0.38306452)   
## 6) TotalDonations< 4.5 122 30 No (0.75409836 0.24590164) \*  
## 7) TotalDonations>=4.5 126 61 Yes (0.48412698 0.51587302)   
## 14) Mnths\_Since\_First>=50 46 14 No (0.69565217 0.30434783)   
## 28) TotalDonations< 12.5 30 3 No (0.90000000 0.10000000) \*  
## 29) TotalDonations>=12.5 16 5 Yes (0.31250000 0.68750000) \*  
## 15) Mnths\_Since\_First< 50 80 29 Yes (0.36250000 0.63750000)   
## 30) Mnths\_Since\_First>=18.5 68 28 Yes (0.41176471 0.58823529)   
## 60) TotalDonations< 5.5 15 5 No (0.66666667 0.33333333) \*  
## 61) TotalDonations>=5.5 53 18 Yes (0.33962264 0.66037736) \*  
## 31) Mnths\_Since\_First< 18.5 12 1 Yes (0.08333333 0.91666667) \*

#extract the tree's fit from the fit object  
tree2 = Blood\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
#plot the tree  
rpart.plot(tree2)



#alternative  
fancyRpartPlot(tree2, tweak = 1.5)



Blood\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.05913978 0 1.0000000 1.0000000 0.07843768  
## 2 0.04838710 3 0.8225806 0.9032258 0.07565844  
## 3 0.02016129 4 0.7741935 0.8709677 0.07465577  
## 4 0.01000000 6 0.7338710 0.8145161 0.07280218

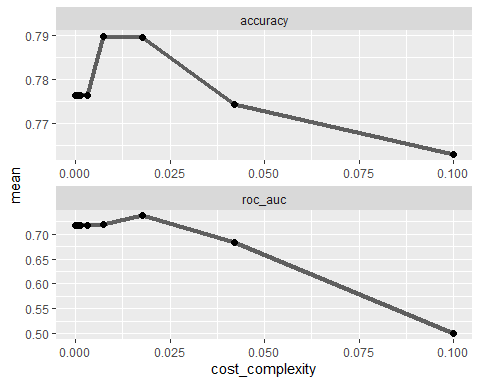
Create folds

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

Blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model2 = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
 set\_mode("classification")  
  
tree\_grid2 = grid\_regular(cost\_complexity(),   
 levels = 25) #try 25 sensible values for cp  
  
Blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model2) %>%   
 add\_recipe(Blood\_recipe)  
  
tree\_res2 =   
 Blood\_wflow %>%   
 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 [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)



Graphically examine how the relationship between cp values and accuracy. What cp value appears to be “optimal” to maximize accuracy?  
The cp value that appears optimal is approximately 0.04.

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

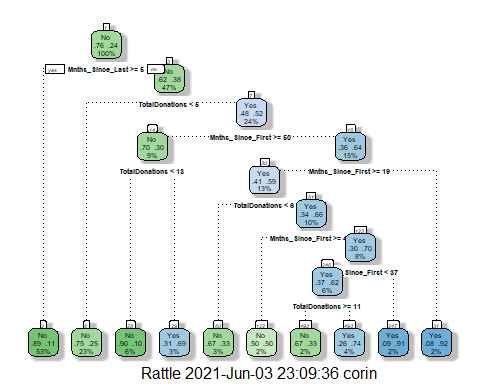
## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.00750 Preprocessor1\_Model22

## Task 10

Plot the tree with optimal cp from Task 9.

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 11

Determine the accuracy of the tree from Task 10 on the training and testing sets. Comment on the tree’s performance on these sets. Predictions on training set

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

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

Caret confusion matrix and accuracy, etc.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 375 67  
## Yes 24 57  
##   
## Accuracy : 0.826   
## 95% CI : (0.7907, 0.8575)  
## No Information Rate : 0.7629   
## P-Value [Acc > NIR] : 0.0002825   
##   
## Kappa : 0.4538   
##   
## Mcnemar's Test P-Value : 1.069e-05   
##   
## Sensitivity : 0.4597   
## Specificity : 0.9398   
## Pos Pred Value : 0.7037   
## Neg Pred Value : 0.8484   
## Prevalence : 0.2371   
## Detection Rate : 0.1090   
## Detection Prevalence : 0.1549   
## Balanced Accuracy : 0.6998   
##   
## 'Positive' Class : Yes   
##

Predictions on testing set

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

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

Caret confusion matrix and accuracy, etc.

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 148 32  
## Yes 23 22  
##   
## Accuracy : 0.7556   
## 95% CI : (0.694, 0.8102)  
## No Information Rate : 0.76   
## P-Value [Acc > NIR] : 0.5975   
##   
## Kappa : 0.2894   
##   
## Mcnemar's Test P-Value : 0.2807   
##   
## Sensitivity : 0.40741   
## Specificity : 0.86550   
## Pos Pred Value : 0.48889   
## Neg Pred Value : 0.82222   
## Prevalence : 0.24000   
## Detection Rate : 0.09778   
## Detection Prevalence : 0.20000   
## Balanced Accuracy : 0.63645   
##   
## 'Positive' Class : Yes   
##

The training set has an accuracy of 0.826 while the testing set has an accuracy of 0.7556 meaning that the testing set was more accurate.