# Module 4: Assignment 1

## Maliszewski, Angela

## Classification Tree Assignment

In this assignment you will complete a variety of tasks related to binary classification with classification trees.

The dataset that we will be using is related to criminal justice and deals specifically with parole violations.

Deliverable: All of your work for this assignment should be done in an R Markdown document. Knit your document into a Word file and submit the Word file as the deliverable for this assignment.

Libraries: For this assignment you will need the following libraries: tidyverse, tidymodels, caret, rpart, rpart.plot, rattle, and RColorBrewer.

Before beginning the assignment tasks, you should read-in the data for the assignment into a data frame called parole. Carefully convert the male, race, state, crime, multiple.offenses, and violator variables to factors. Recode (rename) the factor levels of each of these variables. Note: You did this in a previous assignment. I would encourage you to re-use your code.

parole <- read\_csv("parole.csv")

## Parsed with 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)

## tibble [675 x 9] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ male : num [1:675] 1 0 1 1 1 1 1 0 0 1 ...  
## $ race : num [1:675] 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 : num [1:675] 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: num [1:675] 0 0 0 0 0 0 0 0 0 0 ...  
## $ crime : num [1:675] 4 3 3 1 1 4 3 1 3 2 ...  
## $ violator : num [1:675] 0 0 0 0 0 0 0 0 0 0 ...  
## - 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   
## Min. :0.0000 Min. :1.000 Min. :18.40 Min. :1.000   
## 1st Qu.:1.0000 1st Qu.:1.000 1st Qu.:25.35 1st Qu.:2.000   
## Median :1.0000 Median :1.000 Median :33.70 Median :3.000   
## Mean :0.8074 Mean :1.424 Mean :34.51 Mean :2.887   
## 3rd Qu.:1.0000 3rd Qu.:2.000 3rd Qu.:42.55 3rd Qu.:4.000   
## Max. :1.0000 Max. :2.000 Max. :67.00 Max. :4.000   
## time.served max.sentence multiple.offenses crime   
## Min. :0.000 Min. : 1.00 Min. :0.0000 Min. :1.000   
## 1st Qu.:3.250 1st Qu.:12.00 1st Qu.:0.0000 1st Qu.:1.000   
## Median :4.400 Median :12.00 Median :1.0000 Median :2.000   
## Mean :4.198 Mean :13.06 Mean :0.5363 Mean :2.059   
## 3rd Qu.:5.200 3rd Qu.:15.00 3rd Qu.:1.0000 3rd Qu.:3.000   
## Max. :6.000 Max. :18.00 Max. :1.0000 Max. :4.000   
## violator   
## Min. :0.0000   
## 1st Qu.:0.0000   
## Median :0.0000   
## Mean :0.1156   
## 3rd Qu.:0.0000   
## Max. :1.0000

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, "Other" = "2", "White" = "1" )) %>%   
 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", "drugrelated" = "3", "drivingrelated" = "4" )) %>%   
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%   
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "One" = "0", "Multiple" = "1" )) %>%  
 mutate(violator = as\_factor(violator)) %>%   
 mutate(violator = fct\_recode(violator, "NoViolation" = "0", "Violation" = "1" ))  
str(parole)

## tibble [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","Other": 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 "One","Multiple": 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 "NoViolation",..: 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 Other: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 One :313 OtherCrime :315 NoViolation:597   
## 1st Qu.:12.00 Multiple:362 larceny :106 Violation : 78   
## Median :12.00 drugrelated :153   
## Mean :13.06 drivingrelated:101   
## 3rd Qu.:15.00   
## Max. :18.00

#Task 1:

Split the data into training and testing sets. Your training set should have 70% of the data. Use a random number (set.seed) of 12345.

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

#Task 2:

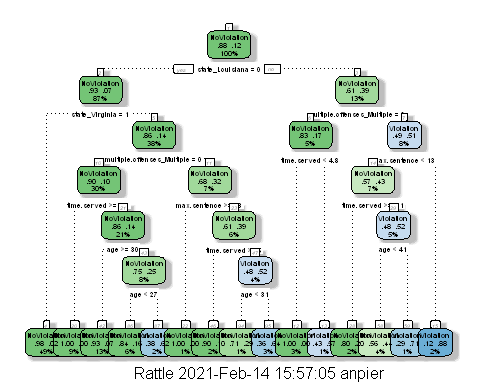
Create a classification tree to predict “violator” in the training set (using all of the other variables as predictors). Plot the tree. You do not need to tune the complexity parameter (i.e., it’s OK to allow R to try different cp values on its own).

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)

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

Plot the tree.

fancyRpartPlot(tree, tweak=1.5)



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

parole\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.03389831 0 1.0000000 1.000000 0.1223796  
## 2 0.02542373 3 0.8983051 1.101695 0.1275886  
## 3 0.01694915 5 0.8474576 1.084746 0.1267465  
## 4 0.01355932 6 0.8305085 1.084746 0.1267465  
## 5 0.01129944 11 0.7627119 1.186441 0.1316539  
## 6 0.01000000 14 0.7288136 1.152542 0.1300561

#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.

ANSWER: No violation, 56% of time, 4% of cases. Steps taken to come to this conclusion are: 1) where state of Louisiana = 0 go to right, 2) multiple offenses = 0 go to right, 3) max sentence < 13 years go to left, 4) time served >= 5.1 go to right 5) age < 41 go to left

#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)?

ANSWER: No, the smallest xerror is actually 1.0 at no splits.

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

ANSWER: No

#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.

Use the code from the lecture to graphically examine model performance for the different values of cp. Hint: You can reuse the vast majority of the code that I provided for you. Be careful to change names and you should be “good to go”. Note: This model took about two minutes to run on my computer. Your run time will vary by your computational power :)

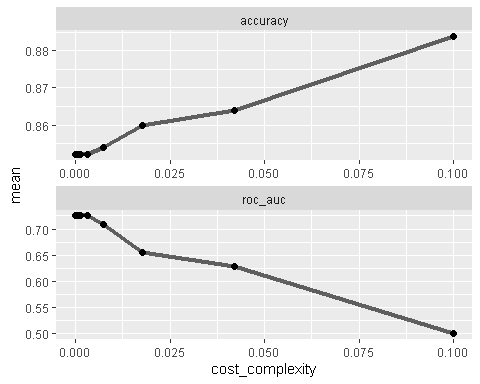
Create our folds

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

parole\_recipe = recipe(violator ~., train) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 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 [405/102]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [405/102]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [406/101]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [406/101]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [406/101]> 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:

Which cp value yields the “optimal” accuracy value?

ANSWER: 0.1 Cp value

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

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.1 Preprocessor1\_Model25

#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.

MASSIVE HINT: If you did Task 6 correctly, and you do this task correctly you will get an error (when you try to fancyRpartPlot the tree) that looks something like: “Error in apply(modelyval2[, yval2per], 1, function(x) x[1 + x[1]]) : dim(X) must have a positive length”.

Admittedly, this error is not particularly helpful. What is means is that the cp value yielded a tree with no splits. A tree with no splits is called a “root” and results in all observations in the dataset being predicted to belong to the “majority class” of the response variable. This type of prediction is often referred to as the “naive” prediction or model. In this dataset, the “majority class” of the response variable is associated with parolees that did NOT violate parole. We can tell this by looking at the summary of our training dataframe and seeing that there are more non-violators than there are violators.

A tree with no splits (i.e., a root) is a common result of a classification tree when the response variable is unbalanced (i.e., there are many more rows associated with one level of the response than with the other).

MASSIVE HINT 2: Comment out the fancyRpartPlot code that results in the error. Failure to do so will not let you knit!!

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

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

#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. ANSWER: 0.8836

Predictions on training set

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

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 NoViolation  
## 2 NoViolation  
## 3 NoViolation  
## 4 NoViolation  
## 5 NoViolation  
## 6 NoViolation

Caret confusion matrix and accuracy, etc. calcs

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction NoViolation Violation  
## NoViolation 448 59  
## Violation 0 0  
##   
## Accuracy : 0.8836   
## 95% CI : (0.8525, 0.9102)  
## No Information Rate : 0.8836   
## P-Value [Acc > NIR] : 0.5346   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 4.321e-14   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8836   
## Prevalence : 0.1164   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Violation   
##

#Task 9:

Read in the “Blood.csv” dataset. The dataset contains five variables: Mnths\_Since\_Last: Months since last donation TotalDonations: Total number of donation Total\_Donated: Total amount of blood donated Mnths\_Since\_First: Months since first donation DonatedMarch: Binary variable representing whether he/she donated blood in March (1 = Yes, 0 = No) Convert the DonatedMarch variable to a factor and recode the variable so 0 = “No” and 1 = “Yes”.

blood <- read\_csv("Blood.csv")

## Parsed with 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)

## tibble [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 : num [1:748] 1 1 1 1 0 0 1 0 1 1 ...  
## - 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)

## 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" ))  
str(blood)

## tibble [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)

## 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  
## No :570   
## Yes:178   
##   
##   
##   
##

#Task 9:

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.

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

ANSWER: 0.1

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

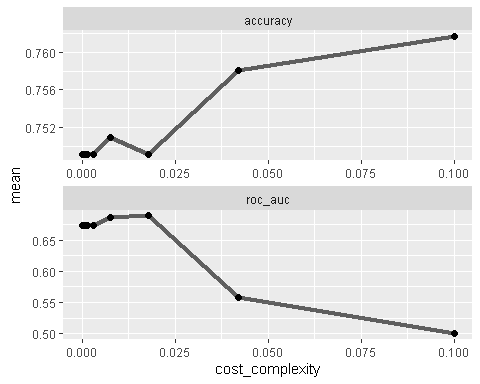
Create our folds

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

blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model2 = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 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)  
  
tree2\_res =   
 blood\_wflow %>%   
 tune\_grid(  
 resamples = folds2,  
 grid = tree\_grid2  
 )  
  
tree2\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [449/113]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [449/113]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [450/112]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [450/112]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [450/112]> Fold5 <tibble [50 x 5]> <tibble [0 x 1]>

tree2\_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\_tree2 = tree2\_res %>%  
 select\_best("accuracy")  
  
best\_tree2

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.1 Preprocessor1\_Model25

#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)

#Task 11:

You’ve almost made it to the end :) Determine the accuracy of the tree from Task 10 on the training and testing sets. Comment on the tree’s performance on these sets.

ANSWER: Tree performance is not great on either the parole dataset or the blood dataset because of having a dominate variable in both sets.

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

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

confusionMatrix(treepred2$.pred\_class,train2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 428 134  
## Yes 0 0  
##   
## Accuracy : 0.7616   
## 95% CI : (0.7241, 0.7962)  
## No Information Rate : 0.7616   
## P-Value [Acc > NIR] : 0.5232   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.7616   
## Prevalence : 0.2384   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

treepred2 = predict(final\_fit2, test2, type = "class")  
head(treepred2)

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

confusionMatrix(treepred2$.pred\_class,test2$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 142 44  
## Yes 0 0  
##   
## Accuracy : 0.7634   
## 95% CI : (0.6957, 0.8225)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.5403   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 9.022e-11   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.7634   
## Prevalence : 0.2366   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##