Module 4 Assignment 1

MTracka

6/7/2021

**#Dataset preparation**

#a) Load libraries

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.2 v dplyr 1.0.6  
## v tidyr 1.1.3 v stringr 1.4.0  
## v readr 1.4.0 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidymodels)

## -- Attaching packages -------------------------------------- tidymodels 0.1.3 --

## v broom 0.7.6 v rsample 0.1.0   
## v dials 0.0.9 v tune 0.1.5   
## v infer 0.5.4 v workflows 0.2.2   
## v modeldata 0.1.0 v workflowsets 0.0.2   
## v parsnip 0.1.5 v yardstick 0.0.8   
## v recipes 0.1.16

## -- Conflicts ----------------------------------------- tidymodels\_conflicts() --  
## x scales::discard() masks purrr::discard()  
## x dplyr::filter() masks stats::filter()  
## x recipes::fixed() masks stringr::fixed()  
## x dplyr::lag() masks stats::lag()  
## x yardstick::spec() masks readr::spec()  
## x recipes::step() masks stats::step()  
## \* Use tidymodels\_prefer() to resolve common conflicts.

library(caret)

## Loading required package: lattice

##   
## Attaching package: 'caret'

## The following objects are masked from 'package:yardstick':  
##   
## precision, recall, sensitivity, specificity

## The following object is masked from 'package:purrr':  
##   
## lift

library(rpart)

##   
## Attaching package: 'rpart'

## The following object is masked from 'package:dials':  
##   
## prune

library(rpart.plot)  
library(rattle)

## Loading required package: bitops

## Rattle: A free graphical interface for data science with R.  
## Version 5.4.0 Copyright (c) 2006-2020 Togaware Pty Ltd.  
## Type 'rattle()' to shake, rattle, and roll your data.

library(RColorBrewer)

#Read in data into data frame “parole”

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()  
## )

#b) Convert characters to factors

parole = parole %>% mutate(male = as\_factor(male))  
parole = parole %>% mutate(race = as\_factor(race))  
parole = parole %>% mutate(state = as\_factor(state))  
parole = parole %>% mutate(crime = as\_factor(crime))  
parole = parole %>% mutate(multiple.offenses = as\_factor(multiple.offenses))  
parole = parole %>% mutate(violator = as\_factor(violator))

#c) Recode factor levels

parole = parole %>% mutate(male = fct\_recode(male, "male" = "1", "female" = "0"))  
parole = parole %>% mutate(race = fct\_recode(race, "white" = "1", "otherwise" = "2"))  
parole = parole %>% mutate(state = fct\_recode(state, "Kentucky" = "2", "Lousiana" = "3", "Virginia" = "4", "any other state" = "1"))  
parole = parole %>% mutate(crime = fct\_recode(crime, "larceny" = "2", "drug-related" = "3", "driving-related" = "4", "any other-crime" = "1"))  
parole = parole %>% mutate(multiple.offenses = fct\_recode(multiple.offenses,"otherwise" = "0", "multiple-offense" = "1"))  
parole = parole %>% mutate(violator = fct\_recode(violator, "no-violated" = "0", "violated" = "1"))

# Assignment 1 Execution

**#Task 1**

#Split the data into training and testing set, 70%/30%, random number 12345

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

**#Task 2**

#Classification tree to predict “violator” in the training set

parole\_recipe = recipe(violator ~., train)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())

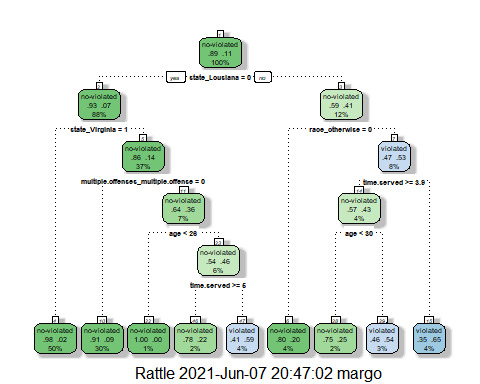
tree\_model= decision\_tree() %>%  
set\_engine("rpart", model = TRUE) %>%  
set\_mode("classification")

parole\_wflow =  
workflow() %>%  
add\_model(tree\_model) %>%  
add\_recipe(parole\_recipe)

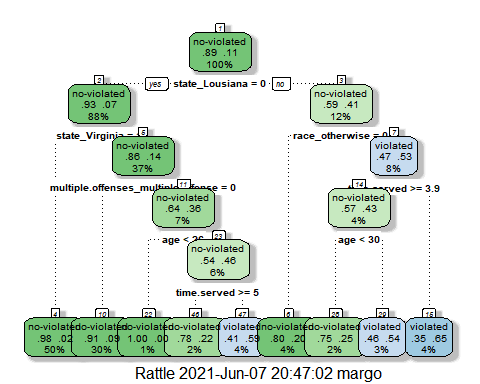
set.seed(12345)  
parole\_fit = fit(parole\_wflow, train)  
tree = parole\_fit %>%  
pull\_workflow\_fit() %>%  
pluck("fit")

#Plot the tree

fancyRpartPlot(tree)



fancyRpartPlot(tree, tweak = 1.3)



**# Task 3**

#40 year old parolee from Luisiana , served 5 years with sentence 10 years, multiple offenses would follow the classification path:

# Top of the tree [Box1], state\_Lousiana=0,–>NO (Box3),

race\_otherwise=0 #Since the race is unknown:

# scenario 1, if race is white (race\_otherwise=0)YES it should go to the bottom of the tree [Box 6]- non-violated rate of 20%

# scenario 2 if race\_otherwise (other than white) to move to NO [Box 7] continue to the next step

#time served greater or equal to 3.9 years is YES continue to [Box 14]

# age less than 30 NO go to [Box 29] violated with violation rate of 54%

**#Task 4**

#Complexity parameter (cp) evaluation

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 1.148148 0.1358793  
## 3 0.01388889 4 0.8888889 1.277778 0.1421129  
## 4 0.01000000 8 0.8333333 1.296296 0.1429608

#The tree from Task 2 is not associated with the optimal cp since the xerror is not minimized.

**#Task 5**

#Tuning grid for 25 different values of cp

#Create 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)

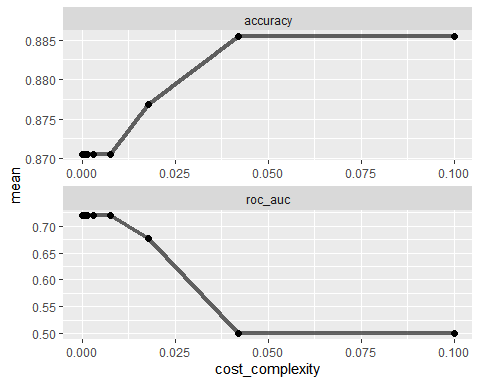
parole\_wflow =   
workflow() %>%   
add\_model(tree\_model) %>%   
add\_recipe(parole\_recipe)

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 [50 x 5]> <tibble [0 x 1]>  
## 2 <split [377/94]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [377/94]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [377/94]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [377/94]> Fold5 <tibble [50 x 5]> <tibble [0 x 1]>

#Use codes from <https://www.tidymodels.org/start/tuning/> to look at the complexity models

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**

# Optimal accuracy value

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

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

#The best accuracy value is obtained with cp of 0.04216965

**#Task 7**

#Final tree

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)

#As described in the MASSIVE HINT fancyRpartPlot(tree, tweak = 1.5) gave ###Error in apply(modelframeyval2[, yval2per], 1, function(x) x[1 + x[1]]) : dim(X) must have a positive length### #Consequently, fancyRpartPlot(tree, tweak = 1.5) was removed from model execution. #Above is also true for the second time execution

**#Task 8**

#Accuracy of the “root”

tree\_res %>%  
 collect\_metrics()

## # A tibble: 50 x 7  
## cost\_complexity .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 1 e-10 accuracy binary 0.871 5 0.00899 Preprocessor1\_Model01  
## 2 1 e-10 roc\_auc binary 0.719 5 0.0428 Preprocessor1\_Model01  
## 3 2.37e-10 accuracy binary 0.871 5 0.00899 Preprocessor1\_Model02  
## 4 2.37e-10 roc\_auc binary 0.719 5 0.0428 Preprocessor1\_Model02  
## 5 5.62e-10 accuracy binary 0.871 5 0.00899 Preprocessor1\_Model03  
## 6 5.62e-10 roc\_auc binary 0.719 5 0.0428 Preprocessor1\_Model03  
## 7 1.33e- 9 accuracy binary 0.871 5 0.00899 Preprocessor1\_Model04  
## 8 1.33e- 9 roc\_auc binary 0.719 5 0.0428 Preprocessor1\_Model04  
## 9 3.16e- 9 accuracy binary 0.871 5 0.00899 Preprocessor1\_Model05  
## 10 3.16e- 9 roc\_auc binary 0.719 5 0.0428 Preprocessor1\_Model05  
## # ... with 40 more rows

#The accuracy associated with the cost complexity of 0.04216965 can be determined from tree metrics and is 0.8854199.

Alternatively, we can do some tuning to find the best cp value for the highest accuracy.

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\_model = decision\_tree(cost\_complexity = tune()) %>%   
set\_engine("rpart", model = TRUE) %>%   
set\_mode("classification")

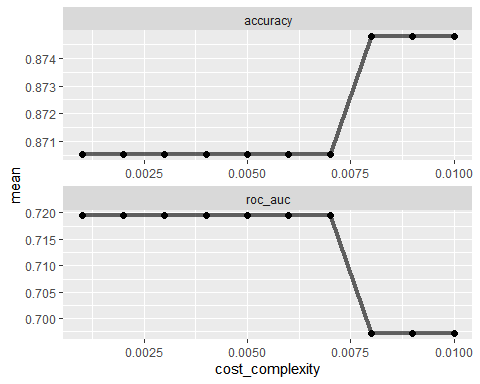
tree\_grid = expand.grid(cost\_complexity= seq(0.001,0.01, by=0.001))

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 [20 x 5]> <tibble [0 x 1]>  
## 2 <split [377/94]> Fold2 <tibble [20 x 5]> <tibble [0 x 1]>  
## 3 <split [377/94]> Fold3 <tibble [20 x 5]> <tibble [0 x 1]>  
## 4 <split [377/94]> Fold4 <tibble [20 x 5]> <tibble [0 x 1]>  
## 5 <split [377/94]> Fold5 <tibble [20 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)



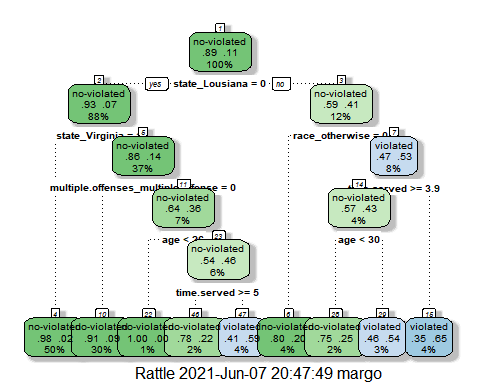
best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.008 Preprocessor1\_Model08

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 =1.3)



**#Task 9**

#Prepare the data set

# Read in dataset “Blood.csc”. Convert “DonatedMarch” variable to a factor, recode the variable so “0” = “No” and “1” =“Yes”.

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))

#Recode DonatedMarch

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

#Split the dataset into train2 (70%) and test2(30%)

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)%>%  
 step\_dummy(all\_nominal(), -all\_outcomes())

tree\_model2= decision\_tree() %>%  
set\_engine("rpart", model = TRUE) %>%  
set\_mode("classification")

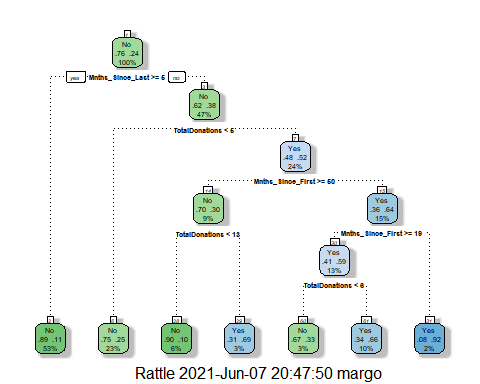
blood\_wflow =  
workflow() %>%  
add\_model(tree\_model2) %>%  
add\_recipe(blood\_recipe)

set.seed(1234)  
blood\_fit = fit(blood\_wflow, train2)

tree2 = blood\_fit %>%  
pull\_workflow\_fit() %>%  
pluck("fit")

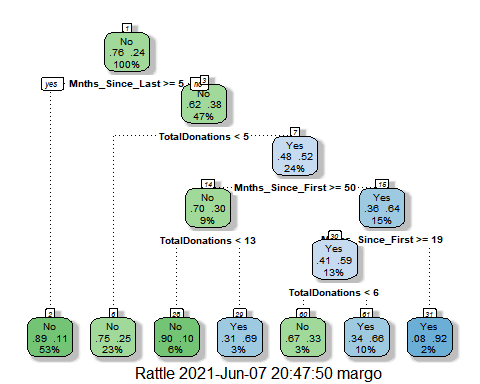
#Plot the tree

fancyRpartPlot(tree2)



#Adding clarity

fancyRpartPlot(tree2, tweak = 1.3)



set.seed(1234)  
folds2 = vfold\_cv(train2, v = 5)  
blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
step\_dummy(all\_nominal(), -all\_outcomes())

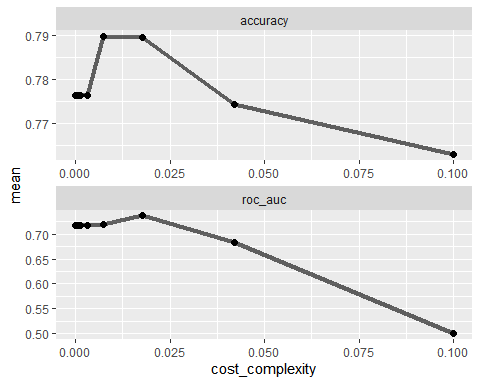
tree\_model2 = decision\_tree(cost\_complexity = tune()) %>%   
set\_engine("rpart", model = TRUE) %>%   
set\_mode("classification")  
tree\_grid2 = grid\_regular(cost\_complexity(),   
 levels = 25)

blood\_wflow =   
workflow() %>%   
add\_model(tree\_model2) %>%   
add\_recipe(blood\_recipe)

tree\_res2 =   
blood\_wflow %>%   
tune\_grid(  
resamples = folds2,  
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 [50 x 5]> <tibble [0 x 1]>  
## 2 <split [418/105]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [418/105]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [419/104]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [419/104]> Fold5 <tibble [50 x 5]> <tibble [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)



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

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

#Accuracy of tree2 using automated approach (grid with 25 levels)

tree\_res2 %>%  
 collect\_metrics()

## # A tibble: 50 x 7  
## cost\_complexity .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 1 e-10 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model01  
## 2 1 e-10 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model01  
## 3 2.37e-10 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model02  
## 4 2.37e-10 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model02  
## 5 5.62e-10 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model03  
## 6 5.62e-10 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model03  
## 7 1.33e- 9 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model04  
## 8 1.33e- 9 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model04  
## 9 3.16e- 9 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model05  
## 10 3.16e- 9 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model05  
## # ... with 40 more rows

#Tuning to find best accuracy

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 = expand.grid(cost\_complexity= seq(0.001,0.01, by=0.001))

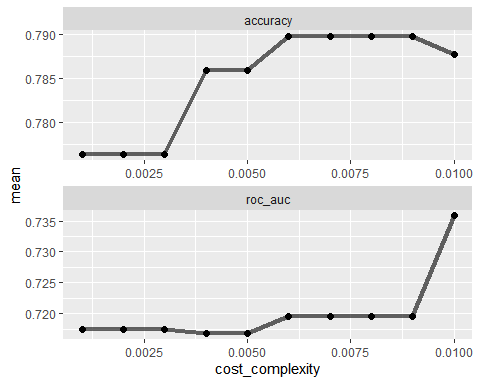
blood\_wflow =   
workflow() %>%   
add\_model(tree\_model2) %>%   
add\_recipe(blood\_recipe)

tree\_res2 =   
blood\_wflow %>%   
tune\_grid(  
resamples = folds2,  
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 [20 x 5]> <tibble [0 x 1]>  
## 2 <split [418/105]> Fold2 <tibble [20 x 5]> <tibble [0 x 1]>  
## 3 <split [418/105]> Fold3 <tibble [20 x 5]> <tibble [0 x 1]>  
## 4 <split [419/104]> Fold4 <tibble [20 x 5]> <tibble [0 x 1]>  
## 5 <split [419/104]> Fold5 <tibble [20 x 5]> <tibble [0 x 1]>

#Use codes from <https://www.tidymodels.org/start/tuning/> to look at the complexity models

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 0.006 Preprocessor1\_Model06

tree\_res2 %>%  
 collect\_metrics()

## # A tibble: 20 x 7  
## cost\_complexity .metric .estimator mean n std\_err .config   
## <dbl> <chr> <chr> <dbl> <int> <dbl> <chr>   
## 1 0.001 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model01  
## 2 0.001 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model01  
## 3 0.002 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model02  
## 4 0.002 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model02  
## 5 0.003 accuracy binary 0.776 5 0.0255 Preprocessor1\_Model03  
## 6 0.003 roc\_auc binary 0.717 5 0.0244 Preprocessor1\_Model03  
## 7 0.004 accuracy binary 0.786 5 0.0250 Preprocessor1\_Model04  
## 8 0.004 roc\_auc binary 0.717 5 0.0225 Preprocessor1\_Model04  
## 9 0.005 accuracy binary 0.786 5 0.0250 Preprocessor1\_Model05  
## 10 0.005 roc\_auc binary 0.717 5 0.0225 Preprocessor1\_Model05  
## 11 0.006 accuracy binary 0.790 5 0.0231 Preprocessor1\_Model06  
## 12 0.006 roc\_auc binary 0.720 5 0.0232 Preprocessor1\_Model06  
## 13 0.007 accuracy binary 0.790 5 0.0231 Preprocessor1\_Model07  
## 14 0.007 roc\_auc binary 0.720 5 0.0232 Preprocessor1\_Model07  
## 15 0.008 accuracy binary 0.790 5 0.0231 Preprocessor1\_Model08  
## 16 0.008 roc\_auc binary 0.720 5 0.0232 Preprocessor1\_Model08  
## 17 0.009 accuracy binary 0.790 5 0.0231 Preprocessor1\_Model09  
## 18 0.009 roc\_auc binary 0.720 5 0.0232 Preprocessor1\_Model09  
## 19 0.01 accuracy binary 0.788 5 0.0177 Preprocessor1\_Model10  
## 20 0.01 roc\_auc binary 0.736 5 0.0257 Preprocessor1\_Model10

#The optimal cp is equal to 0.006 for maximum accuracy of 0.7897619.

**#Task 10**

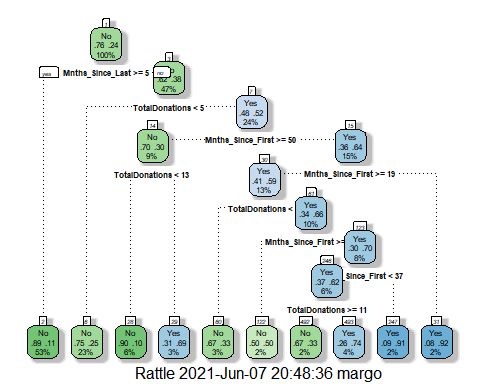
# Final model using cp from Task 9

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 = 1.8)



**#Task 11**

#Accuracy of the tree on the training and testing sets

# Predictions on training set

treepred=predict(final\_fit2, 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 calculations on train2 set

confusionMatrix(treepred$.pred\_class, train2$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   
##

#Predictions on test2 set

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

## # 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 calculations on the test2 set

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

#The accuracy is lower for the test2 set (0.7511) in comparison to the accuracy of the training set of 0.8317