Libraries

library(tidyverse)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.3.0 ──

## ✓ ggplot2 3.3.3 ✓ purrr 0.3.4  
## ✓ tibble 3.0.5 ✓ dplyr 1.0.3  
## ✓ tidyr 1.1.2 ✓ stringr 1.4.0  
## ✓ readr 1.4.0 ✓ forcats 0.5.0

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

library(tidymodels)

## ── Attaching packages ────────────────────────────────────── tidymodels 0.1.2 ──

## ✓ broom 0.7.3 ✓ recipes 0.1.15  
## ✓ dials 0.0.9 ✓ rsample 0.0.8   
## ✓ infer 0.5.4 ✓ tune 0.1.2   
## ✓ modeldata 0.1.0 ✓ workflows 0.2.1   
## ✓ parsnip 0.1.5 ✓ yardstick 0.0.7

## ── 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()

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) #for classification trees

##   
## Attaching package: 'rpart'

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

library(rpart.plot) #for plotting trees  
library(RColorBrewer) #better visualization of classification trees  
library(rattle) #better visualization of classification trees

## 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.

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 %>% mutate(male = as\_factor(male)) %>%  
 mutate(male = fct\_recode(male, "male" = "1", "female" = "0" )) %>%  
 mutate(race = as\_factor(race)) %>%  
 mutate(race = fct\_recode(race, "white" = "1", "other" = "2" )) %>%  
 mutate(state = as\_factor(state)) %>%  
 mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "other" = "1")) %>%  
 mutate(crime = as\_factor(crime)) %>%  
 mutate(crime = fct\_recode(crime, "larceny" = "2", "drug" = "3", "driving" = "4", "other" = "1")) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "multiple" = "1", "other" = "0")) %>%  
 mutate(violator = as\_factor(violator)) %>%  
 mutate(violator = fct\_recode(violator, "yes" = "1", "no" = "0"))

summary(parole)

## male race age state time.served   
## female:130 white:389 Min. :18.40 other :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 other :313 other :315 no :597   
## 1st Qu.:12.00 multiple:362 larceny:106 yes: 78   
## Median :12.00 drug :153   
## Mean :13.06 driving:101   
## 3rd Qu.:15.00   
## Max. :18.00

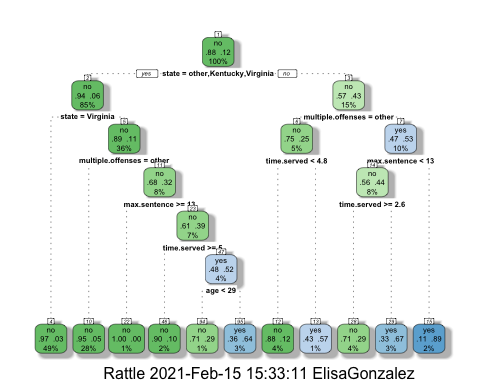
Task 1

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

Task 2

parole\_recipe = recipe(violator ~., train)  
  
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)

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



Task 3 For a parolee from Louisiana we would first go to the right for no, at multiple offenses, we go to the right for no, at max sentence we go left for yes since 10 years is less than 13, at time served we go left for yes = 57% chance they will not violate their parole.

Task 4

parole\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.05454545 0 1.0000000 1.000000 0.1267582  
## 2 0.01818182 4 0.7818182 1.090909 0.1316005  
## 3 0.01454545 5 0.7636364 1.090909 0.1316005  
## 4 0.01000000 10 0.6909091 1.127273 0.1334515

The cp value that is optimal is 0.03389831. The tree from task 2 is not associated with this optimal cp value.

Task 5

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

parole\_recipe = recipe(violator ~., train)  
  
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  
 )

##   
## Attaching package: 'rlang'

## The following objects are masked from 'package:purrr':  
##   
## %@%, as\_function, flatten, flatten\_chr, flatten\_dbl, flatten\_int,  
## flatten\_lgl, flatten\_raw, invoke, list\_along, modify, prepend,  
## splice

##   
## Attaching package: 'vctrs'

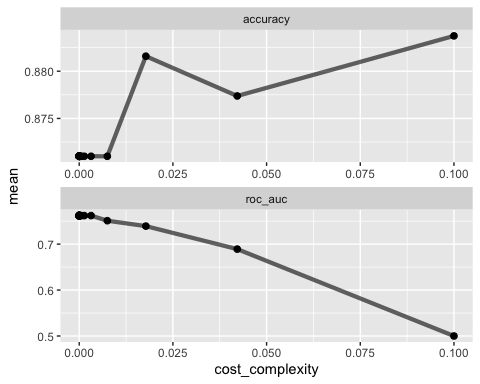
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## The following object is masked from 'package:tibble':  
##   
## data\_frame

tree\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [378/95]> Fold1 <tibble [50 × 5]> <tibble [0 × 1]>  
## 2 <split [378/95]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [378/95]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [379/94]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [379/94]> Fold5 <tibble [50 × 5]> <tibble [0 × 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 0.1 Preprocessor1\_Model25

Task 6

0.1 yields the optimal accuracy value

Task 7

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

448/507 = 88.36%

Task 9

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()  
## )

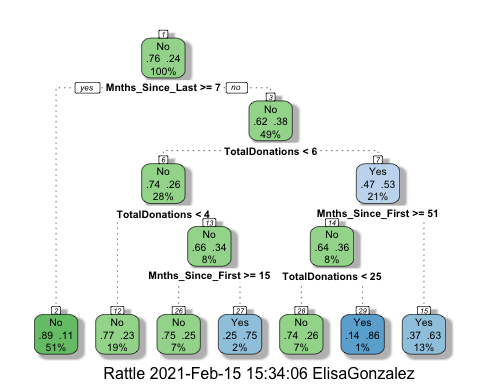
blood = blood %>% mutate(DonatedMarch = as\_factor(DonatedMarch)) %>%   
 mutate(DonatedMarch = fct\_recode(DonatedMarch, "No" = "0", "Yes" = "1" ))

Task 9

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\_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)

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



blood\_fit$fit$fit$fit$cptable

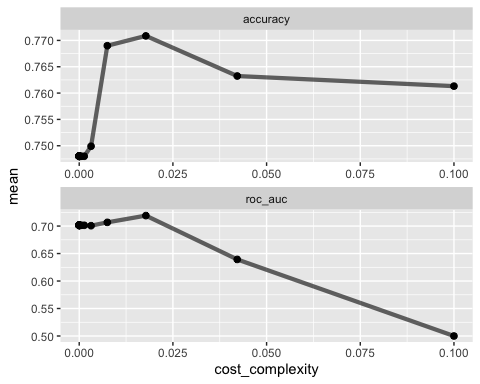
## CP nsplit rel error xerror xstd  
## 1 0.048 0 1.000 1.000 0.07804872  
## 2 0.040 3 0.856 1.000 0.07804872  
## 3 0.016 4 0.816 0.936 0.07626310  
## 4 0.010 6 0.784 0.928 0.07602973

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

blood\_recipe = recipe(DonatedMarch ~., train2)  
  
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 [419/105]> Fold1 <tibble [50 × 5]> <tibble [0 × 1]>  
## 2 <split [419/105]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [419/105]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [419/105]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [420/104]> 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)



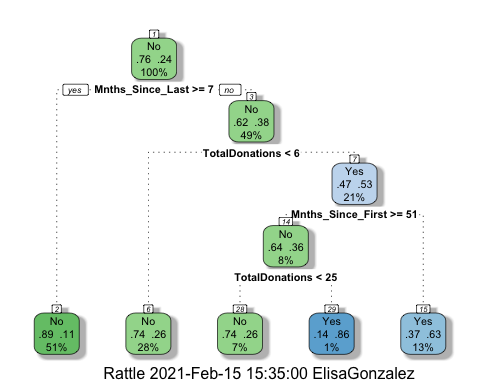
best\_tree2 = tree\_res2 %>%  
 select\_best("accuracy")  
  
best\_tree2

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.0178 Preprocessor1\_Model23

Task 10

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

final\_fit = fit(final\_wf, train2)  
  
tree2 = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
fancyRpartPlot(tree2)



Task 11 Predictions on training set

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

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

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 371 70  
## Yes 28 55  
##   
## Accuracy : 0.813   
## 95% CI : (0.7769, 0.8455)  
## No Information Rate : 0.7615   
## P-Value [Acc > NIR] : 0.002713   
##   
## Kappa : 0.4181   
##   
## Mcnemar's Test P-Value : 3.449e-05   
##   
## Sensitivity : 0.4400   
## Specificity : 0.9298   
## Pos Pred Value : 0.6627   
## Neg Pred Value : 0.8413   
## Prevalence : 0.2385   
## Detection Rate : 0.1050   
## Detection Prevalence : 0.1584   
## Balanced Accuracy : 0.6849   
##   
## 'Positive' Class : Yes   
##

Naive accuracy is 76.15% accuracy of model is 81.3% p-value is less than 0.05, does imply significance difference between the 2 models, marginal gain

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 No   
## 6 Yes

Caret confusion matrix and accuracy, etc. calcs

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

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 159 36  
## Yes 12 17  
##   
## Accuracy : 0.7857   
## 95% CI : (0.7261, 0.8376)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.2418316   
##   
## Kappa : 0.297   
##   
## Mcnemar's Test P-Value : 0.0009009   
##   
## Sensitivity : 0.32075   
## Specificity : 0.92982   
## Pos Pred Value : 0.58621   
## Neg Pred Value : 0.81538   
## Prevalence : 0.23661   
## Detection Rate : 0.07589   
## Detection Prevalence : 0.12946   
## Balanced Accuracy : 0.62529   
##   
## 'Positive' Class : Yes   
##

Accuracy went down to 78.57%, p-value however went up to 0.24