#install.packages("caret")  
#install.packages("rpart")  
#install.packages("rpart.plot")  
#install.packages("rattle")  
#install.packages("RColorBrewer")

library(tidymodels)

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

## ✓ broom 0.7.2 ✓ recipes 0.1.15  
## ✓ dials 0.0.9 ✓ rsample 0.0.8   
## ✓ dplyr 1.0.2 ✓ tibble 3.0.4   
## ✓ ggplot2 3.3.2 ✓ tidyr 1.1.2   
## ✓ infer 0.5.4 ✓ tune 0.1.2   
## ✓ modeldata 0.1.0 ✓ workflows 0.2.1   
## ✓ parsnip 0.1.5 ✓ yardstick 0.0.7   
## ✓ purrr 0.3.4

## ── Conflicts ───────────────────────────────────────── tidymodels\_conflicts() ──  
## x purrr::discard() masks scales::discard()  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()  
## x recipes::step() masks stats::step()

library(tidyverse)

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

## ✓ readr 1.4.0 ✓ forcats 0.5.0  
## ✓ stringr 1.4.0

## ── Conflicts ────────────────────────────────────────── tidyverse\_conflicts() ──  
## x readr::col\_factor() masks scales::col\_factor()  
## x purrr::discard() masks scales::discard()  
## x dplyr::filter() masks stats::filter()  
## x stringr::fixed() masks recipes::fixed()  
## x dplyr::lag() masks stats::lag()  
## x readr::spec() masks yardstick::spec()

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)

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

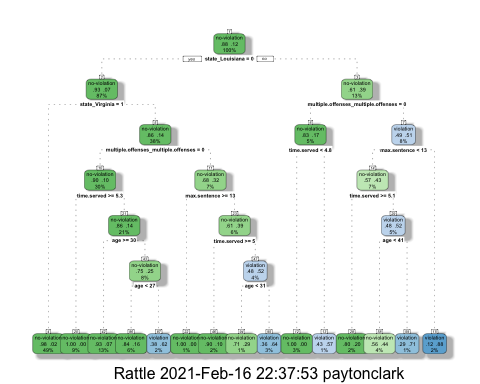
##   
## ── Column specification ────────────────────────────────────────────────────────  
## cols(  
## male = col\_double(),  
## race = col\_double(),  
## age = col\_double(),  
## state = col\_double(),  
## time.served = col\_double(),  
## max.sentence = col\_double(),  
## multiple.offenses = col\_double(),  
## crime = col\_double(),  
## violator = col\_double()  
## )

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

set.seed(12345)  
parole\_split = initial\_split(parole, prob = .70, strata = violator)  
train = training(parole\_split)  
test = testing(parole\_split)

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)

tree = parole\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
fancyRpartPlot(tree, tweak = 1.2)

 Start with the top, state\_Louisiana = 0, you would go left. At multipleoffenses = 0, you would go left. Next, at max sentence you would go right. Then at time served you would go left. At age <41, you would go right. You will arrive at no violation, .56 to .44.

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

The cp value that is optimal is around .0169 and .0136.

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  
 )

##   
## 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'

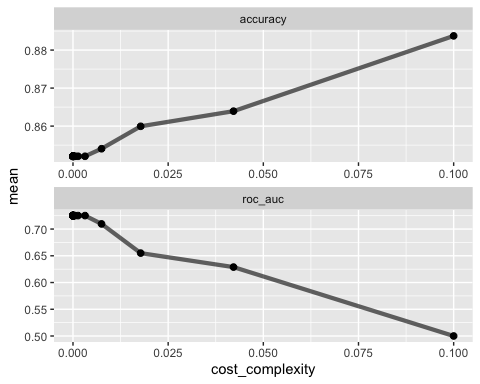
## The following object is masked from 'package:tibble':  
##   
## data\_frame

## The following object is masked from 'package:dplyr':  
##   
## 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 [405/102]> Fold1 <tibble [50 × 5]> <tibble [0 × 1]>  
## 2 <split [405/102]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [406/101]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [406/101]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [406/101]> Fold5 <tibble [50 × 5]> <tibble [0 × 1]>

tree\_res %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size=1.5, alpha=.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

.1

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:

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()  
## )

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

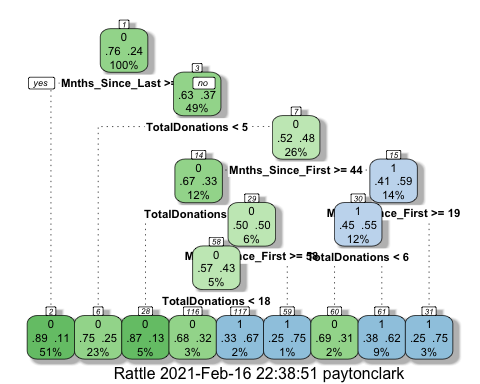
## Warning: Problem with `mutate()` input `DonatedMarch`.  
## ℹ Unknown levels in `f`: Yes, No  
## ℹ Input `DonatedMarch` is `fct\_recode(DonatedMarch, `1` = "Yes", `0` = "No")`.

## Warning: Unknown levels in `f`: Yes, No

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

tree2 = blood\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
fancyRpartPlot(tree2, tweak = 1.5)

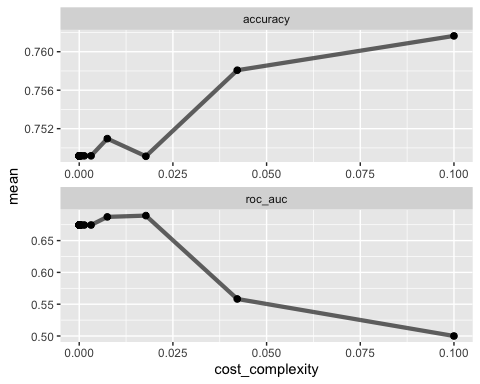


set.seed(1234)  
folds = vfold\_cv(train2, v=5)

blood\_recipe = recipe(DonatedMarch ~ .,train2) %>%  
 step\_dummy(all\_nominal(), -all\_outcomes())  
  
tree\_model2 = decision\_tree(cost\_complexity = tune()) %>%  
 set\_engine("rpart", model = TRUE) %>%  
 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 = 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 [449/113]> Fold1 <tibble [50 × 5]> <tibble [0 × 1]>  
## 2 <split [449/113]> Fold2 <tibble [50 × 5]> <tibble [0 × 1]>  
## 3 <split [450/112]> Fold3 <tibble [50 × 5]> <tibble [0 × 1]>  
## 4 <split [450/112]> Fold4 <tibble [50 × 5]> <tibble [0 × 1]>  
## 5 <split [450/112]> Fold5 <tibble [50 × 5]> <tibble [0 × 1]>

tree\_res2 %>%  
 collect\_metrics() %>%  
 ggplot(aes(cost\_complexity, mean)) +  
 geom\_line(size=1.5, alpha=.6) +  
 geom\_point(size=2) +  
 facet\_wrap(~ .metric, scales= "free", nrow=2)

 .04 seems to be optimal

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

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

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)