Parole <- read\_csv("~/Documents/School/Predictive Analytics/Module 3/Assignment 1/renters insurance.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", "Otherwise" = "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" = "3", "Driving" = "4", "Other" = "1"))  
Parole <- Parole %>% mutate(multiple.offenses= as\_factor(multiple.offenses)) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "Multiple" = "1", "Once" = "0"))  
Parole <- Parole %>% mutate(violator= as\_factor(violator))%>%  
 mutate(violator = fct\_recode(violator, "Violation" = "1", "No Violation" = "0"))

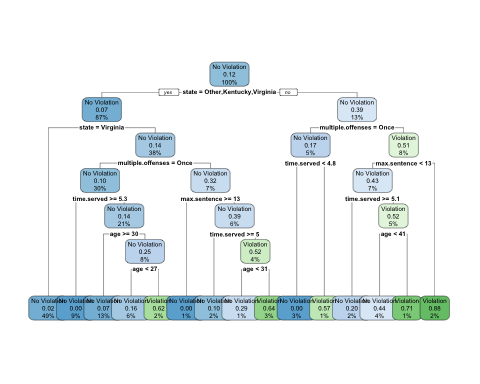
set.seed(12345)  
Parole\_Split = initial\_split(Parole, prob = 0.70, strata = violator)  
train = training(Parole\_Split)  
test = testing(Parole\_Split)

Parole\_recipe = recipe(violator ~ ., 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)

Parole\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")

## n= 507   
##   
## node), split, n, loss, yval, (yprob)  
## \* denotes terminal node  
##   
## 1) root 507 59 No Violation (0.88362919 0.11637081)   
## 2) state=Other,Kentucky,Virginia 441 33 No Violation (0.92517007 0.07482993)   
## 4) state=Virginia 250 6 No Violation (0.97600000 0.02400000) \*  
## 5) state=Other,Kentucky 191 27 No Violation (0.85863874 0.14136126)   
## 10) multiple.offenses=Once 153 15 No Violation (0.90196078 0.09803922)   
## 20) time.served>=5.25 45 0 No Violation (1.00000000 0.00000000) \*  
## 21) time.served< 5.25 108 15 No Violation (0.86111111 0.13888889)   
## 42) age>=30.35 68 5 No Violation (0.92647059 0.07352941) \*  
## 43) age< 30.35 40 10 No Violation (0.75000000 0.25000000)   
## 86) age< 26.7 32 5 No Violation (0.84375000 0.15625000) \*  
## 87) age>=26.7 8 3 Violation (0.37500000 0.62500000) \*  
## 11) multiple.offenses=Multiple 38 12 No Violation (0.68421053 0.31578947)   
## 22) max.sentence>=12.5 7 0 No Violation (1.00000000 0.00000000) \*  
## 23) max.sentence< 12.5 31 12 No Violation (0.61290323 0.38709677)   
## 46) time.served>=5 10 1 No Violation (0.90000000 0.10000000) \*  
## 47) time.served< 5 21 10 Violation (0.47619048 0.52380952)   
## 94) age< 31.05 7 2 No Violation (0.71428571 0.28571429) \*  
## 95) age>=31.05 14 5 Violation (0.35714286 0.64285714) \*  
## 3) state=Louisiana 66 26 No Violation (0.60606061 0.39393939)   
## 6) multiple.offenses=Once 23 4 No Violation (0.82608696 0.17391304)   
## 12) time.served< 4.8 16 0 No Violation (1.00000000 0.00000000) \*  
## 13) time.served>=4.8 7 3 Violation (0.42857143 0.57142857) \*  
## 7) multiple.offenses=Multiple 43 21 Violation (0.48837209 0.51162791)   
## 14) max.sentence< 12.5 35 15 No Violation (0.57142857 0.42857143)   
## 28) time.served>=5.05 10 2 No Violation (0.80000000 0.20000000) \*  
## 29) time.served< 5.05 25 12 Violation (0.48000000 0.52000000)   
## 58) age< 40.8 18 8 No Violation (0.55555556 0.44444444) \*  
## 59) age>=40.8 7 2 Violation (0.28571429 0.71428571) \*  
## 15) max.sentence>=12.5 8 1 Violation (0.12500000 0.87500000) \*

tree = Parole\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
  
rpart.plot(tree, tweak=1.5)



### The first step in the classification tree is state, which would be no since were looking at lousina which means we would go to the right. Multiple offenses were yes so then we would go to the left. Since the time served was greater than 5 years we go to the right which leads us to a chance of violation of 57%

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 1.135593 0.1292432  
## 3 0.01694915 5 0.8474576 1.135593 0.1292432  
## 4 0.01355932 6 0.8305085 1.220339 0.1332155  
## 5 0.01129944 11 0.7627119 1.288136 0.1362352  
## 6 0.01000000 14 0.7288136 1.288136 0.1362352

###The optimal cp value would be either .025 or .017. The tree is associated with these values.

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

Parole\_recipe2 = recipe(violator ~., train) %>%  
 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)   
  
Parole\_wflow2 =   
 workflow() %>%   
 add\_model(tree\_model2) %>%   
 add\_recipe(Parole\_recipe2)  
  
tree\_res =   
 Parole\_wflow2 %>%   
 tune\_grid(  
 resamples = folds,  
 grid = tree\_grid2  
 )

##   
## 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'

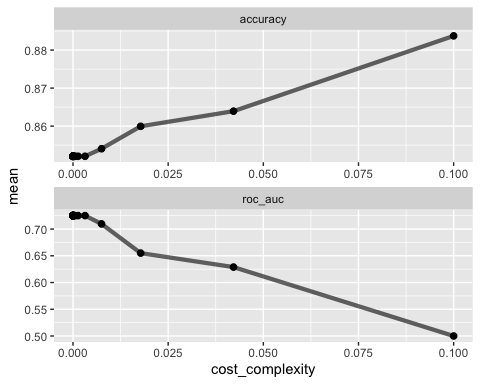
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## 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 [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 = 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

final\_wf =   
 Parole\_wflow2 %>%   
 finalize\_workflow(best\_tree)

final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
###fancyRpartPlot(tree, tweak = 1.5)

###The accuracy of the root is not completely accurate because there is only really one option of non violator or violator. This creates a naive prediction.

Blood <- read\_csv("~/Documents/School/Predictive Analytics/Module 4/Classification Tree/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, "Yes" = "1", "No" = "0"))

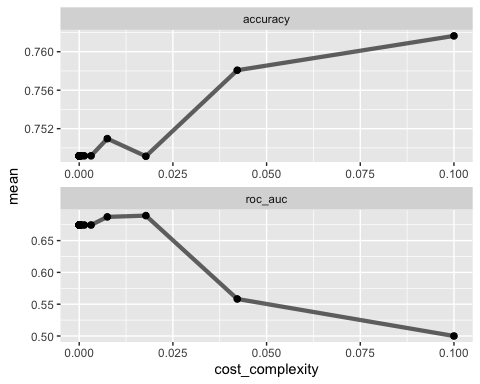
set.seed(1234)  
blood\_Split2 = initial\_split(Blood, prob = 0.70, strata = DonatedMarch)  
train2 = training(blood\_Split2)  
test2 = testing(blood\_Split2)

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

Blood\_recipe = recipe(DonatedMarch ~., train2) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model3 = decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
tree\_grid3 = grid\_regular(cost\_complexity(),  
 levels = 25)   
  
Blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model3) %>%   
 add\_recipe(Blood\_recipe)  
  
tree\_res2 =   
 Blood\_wflow %>%   
 tune\_grid(  
 resamples = folds2,  
 grid = tree\_grid3  
 )  
  
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 = 0.6) +  
 geom\_point(size = 2) +  
 facet\_wrap(~ .metric, scales = "free", nrow = 2)

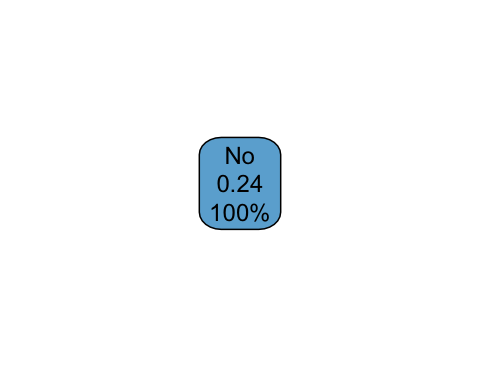


best\_tree2 = tree\_res2 %>%  
 select\_best("accuracy")  
  
best\_tree2

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

final\_wf2 =   
 Blood\_wflow %>%   
 finalize\_workflow(best\_tree2)

final\_fit = fit(final\_wf2, train2)  
  
tree3 = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
  
rpart.plot(tree3, tweak=1.5)



###The tree is very simple and is prediciting just no. This lets us know that this data is naive and it is difficult to predict.