ClassificationTreesAssignment

Je’Kolby Worthy

2/17/2021

# Module 4

# BAN 502 Classification Trees Assignment

### Je’Kolby Worthy

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)

##   
## 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)

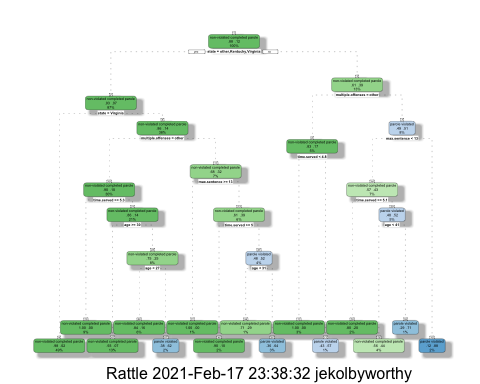
parole <- read\_csv("parole.csv")

##   
## ── 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" = "3", "driving" = "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, "parole violated" = "1", "non-violated completed parole" = "0"))

set.seed(12345)  
parole\_split = initial\_split(parole, prob = 0.7, 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)  
  
tree = parole\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
fancyRpartPlot(tree)



**Task 3**

Starting at the top, we will move to the right because this person is from Louisiana. If they have several crimes, then transfer to the right. Next, move to the left since they have less than 13 sentences and, because their time served was more than 5.1 years, move to the right. Finally, because this individual is less than 41 years old, we will shift to the left to infer that this individual falls into the category of a non-violated completed parole.

**Task 4**

The optimal cp value is 0.01 according to the complexity parameter and the tree from Task 2 is associated with the 14 splits shown in this complexity parameter.

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

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)  
titanic\_wflow =  
 workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(parole\_recipe)  
tree\_res =   
 titanic\_wflow %>%  
 tune\_grid(  
 resamples = folds,   
 grid = tree\_grid  
 )

##   
## Attaching package: 'rlang'

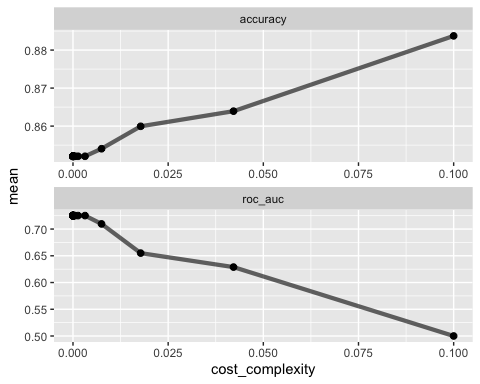
## The following objects are masked from 'package:purrr':  
##   
## %@%, as\_function, flatten, flatten\_chr, flatten\_dbl, flatten\_int,  
## flatten\_lgl, flatten\_raw, invoke, list\_along, modify, prepend,  
## splice

##   
## 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 %>%  
 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** 0.1 yields is the optimal accuracy 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

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** The accuracy is 88%

table(train$violator)

##   
## non-violated completed parole parole violated   
## 448 59

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"))  
summary(blood)

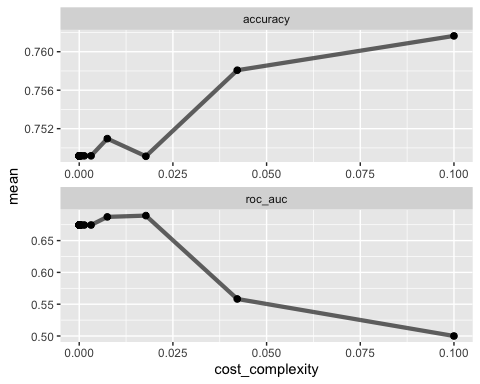
## Mnths\_Since\_Last TotalDonations Total\_Donated Mnths\_Since\_First  
## Min. : 0.000 Min. : 1.000 Min. : 250 Min. : 2.00   
## 1st Qu.: 2.750 1st Qu.: 2.000 1st Qu.: 500 1st Qu.:16.00   
## Median : 7.000 Median : 4.000 Median : 1000 Median :28.00   
## Mean : 9.507 Mean : 5.515 Mean : 1379 Mean :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   
##   
##   
##   
##

set.seed(1234)  
blood\_split = initial\_split(blood, prob = 0.70, strata = DonatedMarch)  
train2 = training(blood\_split)  
test2 = testing(blood\_split)  
  
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 = 0.6) +  
 geom\_point(size = 2) +   
 facet\_wrap(~ .metric, scales = "free", nrow = 2)



Optimal cp value is between .015 and .02

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)

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

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

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

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

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

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

Within the testing and training set, an improvement in accuracy can be seen and the overall performance can be considered good.