# Strickland, Marie

## BAN 502 - Predictive Analytics

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3 v purrr 0.3.4  
## v tibble 3.1.0 v dplyr 1.0.5  
## 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)

## Warning: package 'tidymodels' was built under R version 4.0.5

## -- 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

## Warning: package 'broom' was built under R version 4.0.5

## Warning: package 'dials' was built under R version 4.0.5

## Warning: package 'infer' was built under R version 4.0.5

## Warning: package 'modeldata' was built under R version 4.0.5

## Warning: package 'parsnip' was built under R version 4.0.5

## Warning: package 'rsample' was built under R version 4.0.5

## Warning: package 'tune' was built under R version 4.0.5

## Warning: package 'workflows' was built under R version 4.0.5

## Warning: package 'workflowsets' was built under R version 4.0.5

## Warning: package 'yardstick' was built under R version 4.0.5

## -- 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)

## Warning: package 'caret' was built under R version 4.0.5

## 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)

## Warning: package 'rpart.plot' was built under R version 4.0.5

library(rattle)

## Warning: package 'rattle' was built under R version 4.0.5

## Loading required package: bitops

## Warning: package 'bitops' was built under R version 4.0.5

## 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)  
include = FALSE

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), race = as\_factor(race), state = as\_factor(state), crime = as\_factor(crime), multiple.offenses = as\_factor(multiple.offenses), crime = as\_factor(crime), violator = as\_factor(violator))  
  
parole <- parole %>%  
 mutate(male = fct\_recode(male,   
 'male' = '1',   
 'female' = '0'),  
 race = fct\_recode(race,  
 'white' = '1',  
 'Other' = '2'),  
 state = fct\_recode(state,  
 'Kentucky' = '2',  
 'Louisiana' = '3',  
 'Virginia' = '4',  
 'Other' = '1'),  
 multiple.offenses = fct\_recode(multiple.offenses,  
 'multi offense' = '1',  
 'Other' = '0'),  
 crime = fct\_recode(crime,  
 'larceny' = '2',  
 'drug-related' = '3',  
 'driving related' = '4',  
 'Other' = '1'),  
 violator = fct\_recode(violator,  
 'Yes' = '1',  
 'No' = '0'))

set.seed(12345)  
parole\_split <- initial\_split(parole, prop = 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\_wf <- workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(parole\_recipe)  
  
parole\_fit <- fit(parole\_wf, train)

tree <- parole\_fit %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
  
#fancyRpartPlot(tree, cex = 0.75, tweak = 0.8)

In Task 3, no race was given so I am going to walk through both scenarios. In the first, we start at top and since the parolee is from Louisiana we say no to state = Other, Kentucky, Virginia since Louisiana is a level itself and not in ‘Other’. Next is race; if the parolee was white then they are predicted to not violate parole, with 4% of our data from Louisiana and white and 41% of them did not violate parole. If the parolee was not white then we go to next classification, which is time served >=3.9. Our parolee spent 5 years so we say yes and go to next classification of age < 30. Our parolee is 40, so we go to the right for no, and get a prediction that they violated parole with 4% of our data classified the same and 46% did violate.

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 1.240741 0.1403860  
## 3 0.01388889 4 0.8888889 1.277778 0.1421129  
## 4 0.01000000 8 0.8333333 1.314815 0.1437987

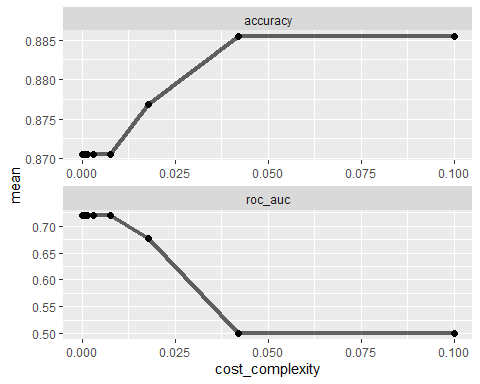
The optimal cp value is 0.0308 with an xerror of 1.0 This cp value is not associated with the tree from Task 2. The cp value is an optimal value for a best fit tree for the fit model on training set.

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\_wf <- workflow() %>%  
 add\_model(tree\_model) %>%  
 add\_recipe(parole\_recipe)  
  
tree\_res <- parole\_wf %>%  
 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]>

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 0.0422 Preprocessor1\_Model24

cp value of 0.0422 yields the optimal accuracy value.

final\_wf <- parole\_wf %>%  
 finalize\_workflow(best\_tree)

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

treepred <- predict(final\_fit, train, type = "class")  
  
confusionMatrix(treepred$.pred\_class,train$violator, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 417 54  
## Yes 0 0  
##   
## Accuracy : 0.8854   
## 95% CI : (0.8531, 0.9127)  
## No Information Rate : 0.8854   
## P-Value [Acc > NIR] : 0.5362   
##   
## Kappa : 0   
##   
## Mcnemar's Test P-Value : 5.498e-13   
##   
## Sensitivity : 0.0000   
## Specificity : 1.0000   
## Pos Pred Value : NaN   
## Neg Pred Value : 0.8854   
## Prevalence : 0.1146   
## Detection Rate : 0.0000   
## Detection Prevalence : 0.0000   
## Balanced Accuracy : 0.5000   
##   
## 'Positive' Class : Yes   
##

This root generated an accuracy of 88.54%. It is a naive tree because the NIR and this trees accuracy are the same.

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),  
 DonatedMarch = fct\_recode(  
 DonatedMarch,   
 'Yes' = '1', 'No' =  
 '0'  
 )  
 )

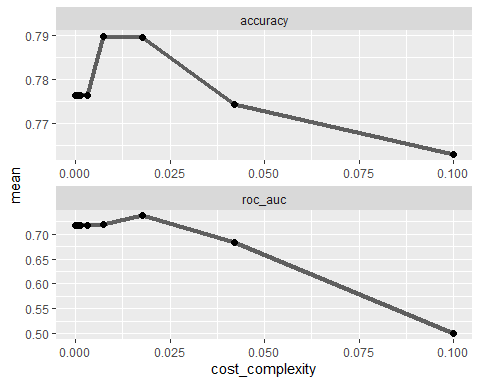
set.seed(1234)  
blood\_split <- initial\_split(blood, prop = 0.7, strata = DonatedMarch)  
train2 <- training(blood\_split)  
test2 <- testing(blood\_split)

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\_wf <- workflow() %>%  
 add\_model(tree\_model2) %>%  
 add\_recipe(blood\_recipe)  
  
tree\_res2 <- blood\_wf %>%  
 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)



starting at a cp value of 0 accuracy is around 0.775, then it begins to increase with a plateau around 0.015 cp. Then steadily decreases as cp rises

A cp value of .0075 yields the most 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 = 0.0075)  
  
blood\_wf <- workflow() %>%  
 add\_model(tree\_model2) %>%  
 add\_recipe(blood\_recipe)  
  
tree\_res2 <- blood\_wf %>%  
 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 [2 x 5]> <tibble [0 x 1]>  
## 2 <split [418/105]> Fold2 <tibble [2 x 5]> <tibble [0 x 1]>  
## 3 <split [418/105]> Fold3 <tibble [2 x 5]> <tibble [0 x 1]>  
## 4 <split [419/104]> Fold4 <tibble [2 x 5]> <tibble [0 x 1]>  
## 5 <split [419/104]> Fold5 <tibble [2 x 5]> <tibble [0 x 1]>

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

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.0075 Preprocessor1\_Model1

Confirming best cp for accuracy

final\_wf2 <- blood\_wf %>%  
 finalize\_workflow(best\_tree2)

final\_fit2 <- fit(final\_wf2, train2)  
  
tree2 <- final\_fit2 %>%  
 pull\_workflow\_fit() %>%  
 pluck("fit")  
  
#fancyRpartPlot(tree2, tweak = 1.6)

treepred2 <- predict(final\_fit2, train2, type = "class")  
  
confusionMatrix(treepred2$.pred\_class,train2$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   
##

Accuracy = 83.17%

treepred2test <- predict(final\_fit2, test2, type = "class")  
  
confusionMatrix(treepred2test$.pred\_class,test2$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   
##

Accuracy = 75.11%

The model accuracy decreased a bit for the testing set, but otherwise, it performed well. The difference between the accuracy and NIR for the training set is statistically significant. Accuracy v NIR for testing set is way closer and not significant.