Mod4Assign1

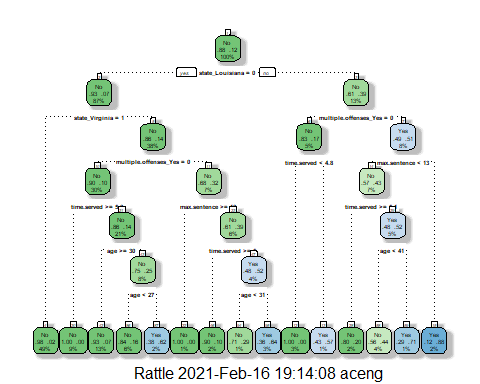
Engel, Alec

parole = read\_csv("parole.csv")

parole = parole %>%  
 mutate(male = as\_factor(male)) %>%  
 mutate(race = as\_factor(race)) %>%  
 mutate(state = as\_factor(state)) %>%  
 mutate(crime = as\_factor(crime)) %>%  
 mutate(multiple.offenses = as\_factor(multiple.offenses)) %>%  
 mutate(violator = as\_factor(violator)) %>%  
 mutate(male = fct\_recode(male, "No" = "0", "Yes" = "1")) %>%  
 mutate(race = fct\_recode(race, "White" = "1", "Non-White" = "2")) %>%  
 mutate(state = fct\_recode(state, "Kentucky" = "2", "Louisiana" = "3", "Virginia" = "4", "Other" = "1")) %>%  
 mutate(multiple.offenses = fct\_recode(multiple.offenses, "Yes" = "1", "No" = "0")) %>%  
 mutate(crime = fct\_recode(crime, "Larceny" = "2", "Drug-related" = "3", "Driving-related" = "4", "Other" = "1")) %>%  
 mutate(violator = fct\_recode(violator, "Yes" = "1", "No" = "0"))  
#str(parole)

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) %>%  
 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=1.1)



**I would classify this parolee as only 44% likely to violate parole and we classify as “no”, not probable that they violated their parole. I came to this conclusion by starting from the top of the tree and working my way to the right as this parolee is from Louisiana, right again as they have indeed committed multiple offenses, then right as their time served is less than 5.1, and then finally to the left branch as the parolee is less than 41 years old at 40.**

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 optimal CP value here looks to be 0.03389831. This CP value correlate to our smallest xerror value’s of 1 and differs from our tree above as the optimal CP would involve zero splits. What this is telling us is that more splits are causing possible overfitting of our data and we are better off not splitting at all.**

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'

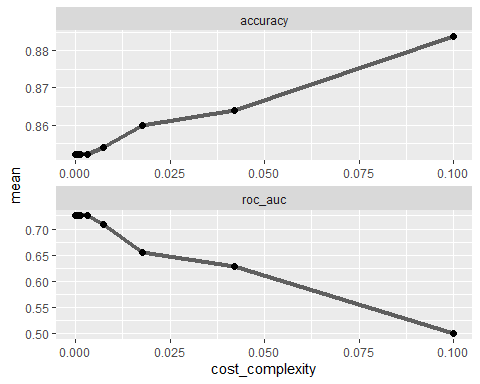
## The following object is masked from 'package:dplyr':  
##   
## data\_frame

## 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 x 5]> <tibble [0 x 1]>  
## 2 <split [405/102]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [406/101]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [406/101]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [406/101]> 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 0.1 Preprocessor1\_Model25

**A CP value of 0.1 yields the “optimal” accuracy value.**

final\_wf =   
 parole\_wflow %>%   
 finalize\_workflow(best\_tree)  
final\_fit = fit(final\_wf, train)  
  
tree = final\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")  
#fancyRpartPlot(tree)

treepred = predict(final\_fit, train, type = "class")  
head(treepred)

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

#summary(treepred)  
#summary(train)  
Accuracy = 448/507  
Accuracy

## [1] 0.8836292

**The accuracy of non parole violators was calculated as 88.36% by dividing number of correct predictions(448) by the total number(507).**

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"))  
str(blood)

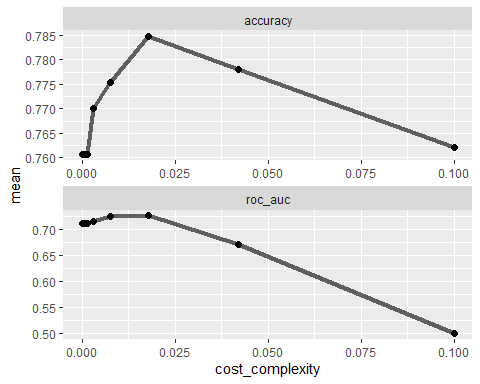
## tibble [748 x 5] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ Mnths\_Since\_Last : num [1:748] 2 0 1 2 1 4 2 1 2 5 ...  
## $ TotalDonations : num [1:748] 50 13 16 20 24 4 7 12 9 46 ...  
## $ Total\_Donated : num [1:748] 12500 3250 4000 5000 6000 1000 1750 3000 2250 11500 ...  
## $ Mnths\_Since\_First: num [1:748] 98 28 35 45 77 4 14 35 22 98 ...  
## $ DonatedMarch : Factor w/ 2 levels "No","Yes": 2 2 2 2 1 1 2 1 2 2 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. Mnths\_Since\_Last = col\_double(),  
## .. TotalDonations = col\_double(),  
## .. Total\_Donated = col\_double(),  
## .. Mnths\_Since\_First = col\_double(),  
## .. DonatedMarch = col\_double()  
## .. )

set.seed(1234)   
blood\_split = initial\_split(blood, prob = 0.70, strata = DonatedMarch)  
train = training(blood\_split)  
test = testing(blood\_split)

set.seed(1234)  
folds = vfold\_cv(blood, v = 5)  
blood\_recipe = recipe(DonatedMarch ~., 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)  
  
blood\_wflow =   
 workflow() %>%   
 add\_model(tree\_model) %>%   
 add\_recipe(blood\_recipe)  
  
tree\_res =   
 blood\_wflow %>%   
 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 [598/150]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [598/150]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [598/150]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [599/149]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [599/149]> 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)

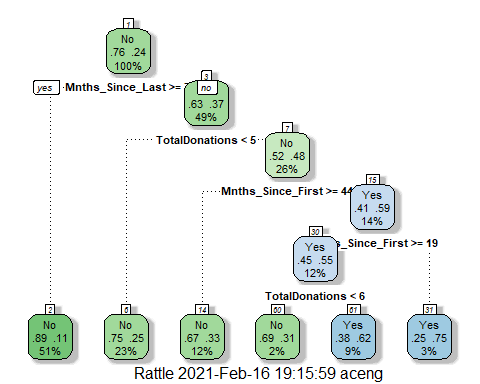


**A cp value slightly above 0.015 seems to be optimal here with the highest correlation with a high accuracy rating of 0.785.**

best\_tree = tree\_res %>%  
 select\_best("accuracy")  
  
best\_tree

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

final\_wf =   
 blood\_wflow %>%   
 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")  
head(treepred)

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

confusionMatrix(treepred$.pred\_class,train$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 404 90  
## Yes 24 44  
##   
## Accuracy : 0.7972   
## 95% CI : (0.7615, 0.8297)  
## No Information Rate : 0.7616   
## P-Value [Acc > NIR] : 0.02523   
##   
## Kappa : 0.3277   
##   
## Mcnemar's Test P-Value : 1.145e-09   
##   
## Sensitivity : 0.32836   
## Specificity : 0.94393   
## Pos Pred Value : 0.64706   
## Neg Pred Value : 0.81781   
## Prevalence : 0.23843   
## Detection Rate : 0.07829   
## Detection Prevalence : 0.12100   
## Balanced Accuracy : 0.63614   
##   
## 'Positive' Class : Yes   
##

treepred\_test = predict(final\_fit, test, type = "class")  
head(treepred\_test)

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

confusionMatrix(treepred\_test$.pred\_class,test$DonatedMarch,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 137 30  
## Yes 5 14  
##   
## Accuracy : 0.8118   
## 95% CI : (0.7481, 0.8653)  
## No Information Rate : 0.7634   
## P-Value [Acc > NIR] : 0.06858   
##   
## Kappa : 0.352   
##   
## Mcnemar's Test P-Value : 4.976e-05   
##   
## Sensitivity : 0.31818   
## Specificity : 0.96479   
## Pos Pred Value : 0.73684   
## Neg Pred Value : 0.82036   
## Prevalence : 0.23656   
## Detection Rate : 0.07527   
## Detection Prevalence : 0.10215   
## Balanced Accuracy : 0.64149   
##   
## 'Positive' Class : Yes   
##

**I am seeing an accuracy of 79.72% accuracy on the training set while the accuracy actually increases to 81.18% on the testing data. This proves a very strong performance model and I feel comfortable using this model to predict on any future data.**