# ClassTreeAssign1

## Mod 4 Assin 1

### Evan Hildreth

library(tidyverse)

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v ggplot2 3.3.5 v purrr 0.3.4  
## v tibble 3.1.6 v dplyr 1.0.7  
## v tidyr 1.2.0 v stringr 1.4.0  
## v readr 2.1.2 v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse\_conflicts() --  
## x dplyr::filter() masks stats::filter()  
## x dplyr::lag() masks stats::lag()

library(tidymodels)

## Registered S3 method overwritten by 'tune':  
## method from   
## required\_pkgs.model\_spec parsnip

## -- Attaching packages -------------------------------------- tidymodels 0.1.4 --

## v broom 0.7.12 v rsample 0.1.1   
## v dials 0.1.0 v tune 0.1.6   
## v infer 1.0.0 v workflows 0.2.4   
## v modeldata 0.1.1 v workflowsets 0.1.0   
## v parsnip 0.1.7 v yardstick 0.0.9   
## v recipes 0.1.17

## -- 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()  
## \* Dig deeper into tidy modeling with R at https://www.tmwr.org

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(RColorBrewer)

heart\_disease <- read\_csv("heart\_disease-1.csv")

## Rows: 918 Columns: 12

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## chr (5): Sex, ChestPainType, RestingECG, ExerciseAngina, ST\_Slope  
## dbl (7): Age, RestingBP, Cholesterol, FastingBS, MaxHR, Oldpeak, HeartDisease

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

heart\_disease <- heart\_disease %>% mutate(Sex = as\_factor(Sex)) %>%   
 mutate(ChestPainType = as\_factor(ChestPainType)) %>%   
 mutate(RestingECG = as\_factor(RestingECG)) %>%   
 mutate(ExerciseAngina = as\_factor(ExerciseAngina)) %>%   
 mutate(HeartDisease = as\_factor(HeartDisease)) %>%   
 mutate(HeartDisease = fct\_recode(HeartDisease, "No" = "0", "Yes" = "1"))

### Task 1

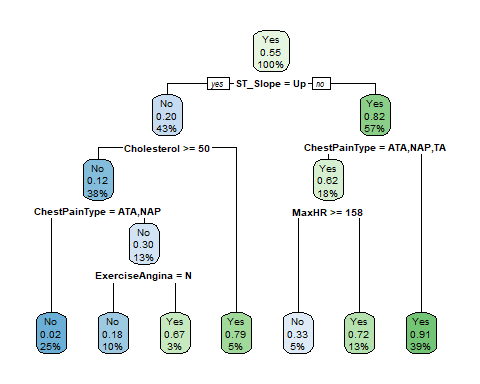
set.seed(12345)  
tearing\_my\_heart\_up <- initial\_split(heart\_disease, prop = 0.7, strata = HeartDisease)  
heart\_disease\_training <- training(tearing\_my\_heart\_up)  
heart\_disease\_testing <- testing(tearing\_my\_heart\_up)

### Task 2

heart\_disease\_recipe1 <- recipe(HeartDisease ~ ., heart\_disease)  
  
Tree\_model <- decision\_tree() %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
heart\_disease\_wflow1 <-   
 workflow() %>%   
 add\_model(Tree\_model) %>%   
 add\_recipe(heart\_disease\_recipe1)  
  
heart\_disease\_fit <- fit(heart\_disease\_wflow1, heart\_disease\_training)  
  
heart\_disease\_pluck <- heart\_disease\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

rpart.plot(heart\_disease\_pluck)



### Task 3

heart\_disease\_fit$fit$fit$fit$cptable

## CP nsplit rel error xerror xstd  
## 1 0.57491289 0 1.0000000 1.0000000 0.04389406  
## 2 0.06620209 1 0.4250871 0.4250871 0.03463635  
## 3 0.01742160 2 0.3588850 0.3588850 0.03240139  
## 4 0.01219512 4 0.3240418 0.3902439 0.03350407  
## 5 0.01000000 6 0.2996516 0.3937282 0.03362154

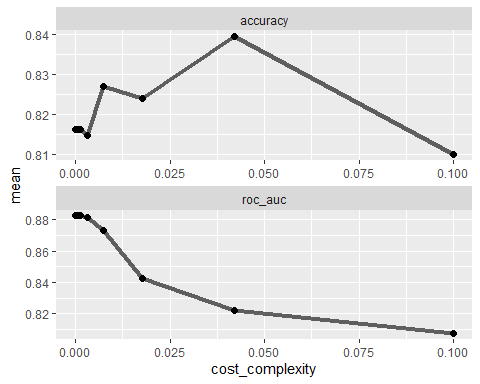
The CP value that corresponds with the smallest xerror value (0.3588850) is 0.01742160.

### Task 4

set.seed(123)  
folds <- vfold\_cv(heart\_disease\_training, v = 5)  
  
heart\_disease\_recipe2 <- recipe(HeartDisease ~., heart\_disease) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model2 <- decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
  
heart\_disease\_grid <- grid\_regular(cost\_complexity(), levels = 25)  
   
heart\_disease\_wflow2 <-   
 workflow() %>%   
 add\_model(tree\_model2) %>%   
 add\_recipe(heart\_disease\_recipe2)  
  
heart\_disease\_res <-   
 heart\_disease\_wflow2 %>%   
 tune\_grid(  
 resamples = folds,  
 grid = heart\_disease\_grid  
 )  
  
heart\_disease\_res

## # Tuning results  
## # 5-fold cross-validation   
## # A tibble: 5 x 4  
## splits id .metrics .notes   
## <list> <chr> <list> <list>   
## 1 <split [513/129]> Fold1 <tibble [50 x 5]> <tibble [0 x 1]>  
## 2 <split [513/129]> Fold2 <tibble [50 x 5]> <tibble [0 x 1]>  
## 3 <split [514/128]> Fold3 <tibble [50 x 5]> <tibble [0 x 1]>  
## 4 <split [514/128]> Fold4 <tibble [50 x 5]> <tibble [0 x 1]>  
## 5 <split [514/128]> Fold5 <tibble [50 x 5]> <tibble [0 x 1]>

heart\_disease\_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 5 Looking at the above graphs it appears that CP value ~ 0.04 looks to provide the optimal accuracy value.

### Task 6

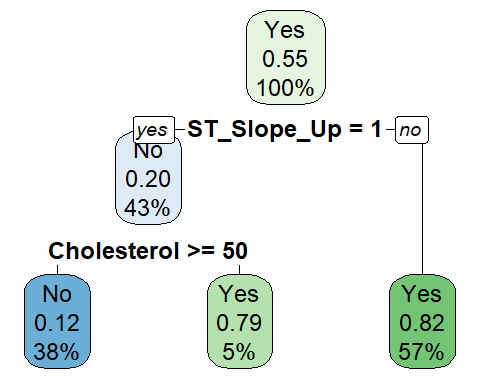
best\_tree\_HDR <- heart\_disease\_res %>%  
 select\_best("accuracy")  
  
best\_tree\_HDR

## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.0422 Preprocessor1\_Model24

final\_wf <-   
 heart\_disease\_wflow2 %>%   
 finalize\_workflow(best\_tree\_HDR)  
  
final\_fit\_HDR = fit(final\_wf, heart\_disease\_training)  
  
tree\_HDR = final\_fit\_HDR %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

rpart.plot(tree\_HDR, tweak = 1.5)



### Task 7

HDR\_predict <- predict(final\_fit\_HDR, heart\_disease\_training, type = "class")  
head(HDR\_predict)

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

confusionMatrix(HDR\_predict$.pred\_class, heart\_disease\_training$HeartDisease, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 213 29  
## Yes 74 326  
##   
## Accuracy : 0.8396   
## 95% CI : (0.8088, 0.8671)  
## No Information Rate : 0.553   
## P-Value [Acc > NIR] : < 2.2e-16   
##   
## Kappa : 0.6705   
##   
## Mcnemar's Test P-Value : 1.455e-05   
##   
## Sensitivity : 0.9183   
## Specificity : 0.7422   
## Pos Pred Value : 0.8150   
## Neg Pred Value : 0.8802   
## Prevalence : 0.5530   
## Detection Rate : 0.5078   
## Detection Prevalence : 0.6231   
## Balanced Accuracy : 0.8302   
##   
## 'Positive' Class : Yes   
##

Running a confusion matrix on the mode (prior to it being plucked) we find that the model has an accuracy of 0.8396.

### Task 8

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

## Rows: 748 Columns: 5

## -- Column specification --------------------------------------------------------  
## Delimiter: ","  
## dbl (5): Mnths\_Since\_Last, TotalDonations, Total\_Donated, Mnths\_Since\_First,...

##   
## i Use `spec()` to retrieve the full column specification for this data.  
## i Specify the column types or set `show\_col\_types = FALSE` to quiet this message.

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

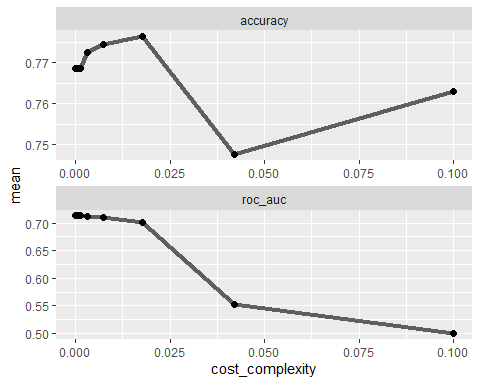
### Task 9

set.seed(1234)  
Blood\_spilt <- initial\_split(Blood, prop = 0.7, strata = DonatedMarch)  
Blood\_training <- training(Blood\_spilt)  
Blood\_testing <- testing(Blood\_spilt)

set.seed(1234)  
blood\_folds <- vfold\_cv(Blood\_training, v = 5)  
  
blood\_recipe <- recipe(DonatedMarch ~., Blood) %>%  
 step\_dummy(all\_nominal(),-all\_outcomes())  
  
blood\_model <- decision\_tree(cost\_complexity = tune()) %>%   
 set\_engine("rpart", model = TRUE) %>%   
 set\_mode("classification")  
  
  
blood\_grid <- grid\_regular(cost\_complexity(), levels = 25)  
   
blood\_wflow <-   
 workflow() %>%   
 add\_model(blood\_model) %>%   
 add\_recipe(blood\_recipe)  
  
Blood\_res <-   
 blood\_wflow %>%   
 tune\_grid(  
 resamples = blood\_folds,  
 grid = blood\_grid  
 )  
  
Blood\_res

## # 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]>

Blood\_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)

 Looking at the chart above, it seems the cp value of roughly ~0.020 possesses the highest accuracy rating.

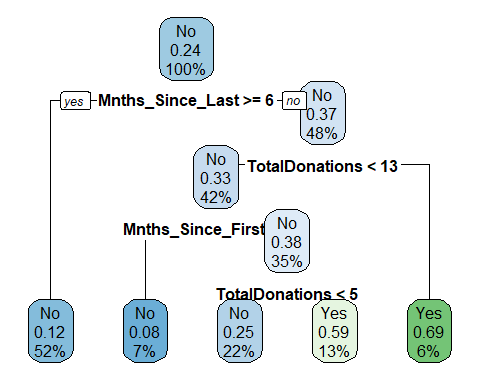
best\_bloody\_tree <- Blood\_res %>%  
 select\_best("accuracy")  
  
best\_bloody\_tree

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

Bloody\_final\_wf <-   
 blood\_wflow %>%   
 finalize\_workflow(best\_bloody\_tree)  
  
bloody\_fit = fit(Bloody\_final\_wf, Blood\_training)  
  
Bloody\_tree\_fit = bloody\_fit %>%   
 pull\_workflow\_fit() %>%   
 pluck("fit")

## Warning: `pull\_workflow\_fit()` was deprecated in workflows 0.2.3.  
## Please use `extract\_fit\_parsnip()` instead.

rpart.plot(Bloody\_tree\_fit, tweak = 1.5)

 ### Task 11

Blood\_predict <- predict(bloody\_fit, Blood\_training, type = "class")  
head(Blood\_predict)

## # A tibble: 6 x 1  
## .pred\_class  
## <fct>   
## 1 Yes   
## 2 Yes   
## 3 Yes   
## 4 Yes   
## 5 Yes   
## 6 Yes

confusionMatrix(Blood\_predict$.pred\_class, Blood\_training$DonatedMarch, positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 361 63  
## Yes 38 61  
##   
## Accuracy : 0.8069   
## 95% CI : (0.7704, 0.8399)  
## No Information Rate : 0.7629   
## P-Value [Acc > NIR] : 0.009214   
##   
## Kappa : 0.4263   
##   
## Mcnemar's Test P-Value : 0.016936   
##   
## Sensitivity : 0.4919   
## Specificity : 0.9048   
## Pos Pred Value : 0.6162   
## Neg Pred Value : 0.8514   
## Prevalence : 0.2371   
## Detection Rate : 0.1166   
## Detection Prevalence : 0.1893   
## Balanced Accuracy : 0.6983   
##   
## 'Positive' Class : Yes   
##

With the blood dataset and the blood model (essentially the same as the tree\_model2 just renamed) we get an accuracy of 0.8069 which is slightly lower than the accuracy received on the heart\_disease dataset (0.8396 accuracy).

In addition, while the P value for both datasets with the model is significant, it is much better on heart disease dataset than the blood one.