# Team Chapman & Niez

## CP phase 2

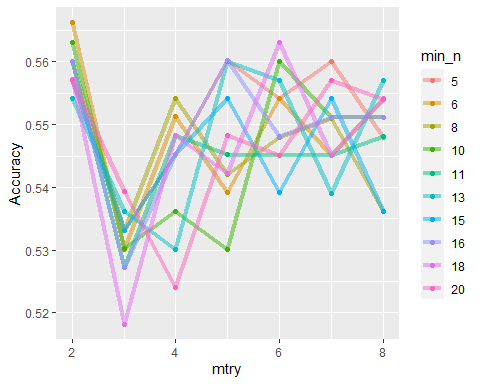
## Random Forest Approach

set.seed(1234)  
shark\_split = initial\_split(sharktank\_clean, prop = 0.7, strata = Deal\_Yes) #70% in training  
train = training(shark\_split)  
test = testing(shark\_split)

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

shark\_recipe = recipe(Deal\_Yes ~., train) %>%  
step\_dummy(all\_nominal(), -all\_outcomes())  
  
rf\_model = rand\_forest(mtry = tune(), min\_n = tune(), trees = 500) %>%  
set\_engine("ranger", importance = "permutation") %>% #added importance metric  
set\_mode("classification")  
  
shark\_wflow =  
workflow() %>%  
add\_model(rf\_model) %>%  
add\_recipe(shark\_recipe)  
  
rf\_grid = grid\_regular(  
mtry(range = c(2, 8)), #these values determined through significant trial and error  
min\_n(range = c(5, 20)), #these values determined through significant trial and error  
levels = 10  
)  
set.seed(123)  
rf\_res = tune\_grid(  
shark\_wflow,  
resamples = rf\_folds,  
grid = rf\_grid  
)

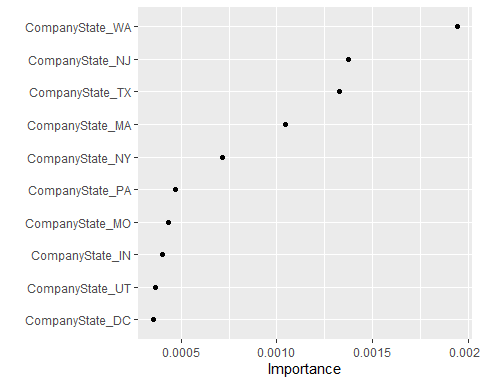
rf\_res %>%  
collect\_metrics() %>%  
filter(.metric == "accuracy") %>%  
mutate(min\_n = factor(min\_n)) %>%  
ggplot(aes(mtry, mean, color = min\_n)) +  
geom\_line(alpha = 0.5, size = 1.5) +  
geom\_point() +  
labs(y = "Accuracy")



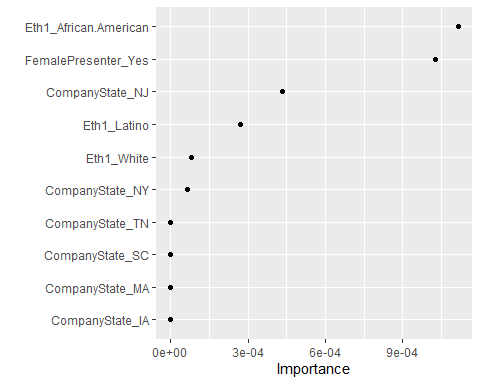
best\_rf = select\_best(rf\_res, "accuracy")  
final\_rf = finalize\_workflow(  
shark\_wflow,  
best\_rf  
)  
final\_rf

## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: rand\_forest()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 1 Recipe Step  
##   
## \* step\_dummy()  
##   
## -- Model -----------------------------------------------------------------------  
## Random Forest Model Specification (classification)  
##   
## Main Arguments:  
## mtry = 2  
## trees = 500  
## min\_n = 6  
##   
## Engine-Specific Arguments:  
## importance = permutation  
##   
## Computational engine: ranger

final\_rf\_fit = fit(final\_rf, train)  
final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



final\_rf\_fit = fit(final\_rf, test)  
final\_rf\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



trainpredrf = predict(final\_rf\_fit, train)  
head(trainpredrf)

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

confusionMatrix(trainpredrf$.pred\_class, train$Deal\_Yes,  
positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 10 6  
## Yes 142 174  
##   
## Accuracy : 0.5542   
## 95% CI : (0.499, 0.6085)  
## No Information Rate : 0.5422   
## P-Value [Acc > NIR] : 0.3504   
##   
## Kappa : 0.0349   
##   
## Mcnemar's Test P-Value : <2e-16   
##   
## Sensitivity : 0.96667   
## Specificity : 0.06579   
## Pos Pred Value : 0.55063   
## Neg Pred Value : 0.62500   
## Prevalence : 0.54217   
## Detection Rate : 0.52410   
## Detection Prevalence : 0.95181   
## Balanced Accuracy : 0.51623   
##   
## 'Positive' Class : Yes   
##

testpredrf = predict(final\_rf\_fit, test)  
head(testpredrf)

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

confusionMatrix(testpredrf$.pred\_class, test$Deal\_Yes,  
positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 11 1  
## Yes 55 77  
##   
## Accuracy : 0.6111   
## 95% CI : (0.5264, 0.6912)  
## No Information Rate : 0.5417   
## P-Value [Acc > NIR] : 0.05548   
##   
## Kappa : 0.1642   
##   
## Mcnemar's Test P-Value : 1.417e-12   
##   
## Sensitivity : 0.9872   
## Specificity : 0.1667   
## Pos Pred Value : 0.5833   
## Neg Pred Value : 0.9167   
## Prevalence : 0.5417   
## Detection Rate : 0.5347   
## Detection Prevalence : 0.9167   
## Balanced Accuracy : 0.5769   
##   
## 'Positive' Class : Yes   
##

## Classification Tree Approach

set.seed(12345)  
shark\_split2 = initial\_split(sharktank\_clean, prop = 0.7, strata = Deal\_Yes) #70% in training  
train2 = training(shark\_split2)  
test2 = testing(shark\_split2)

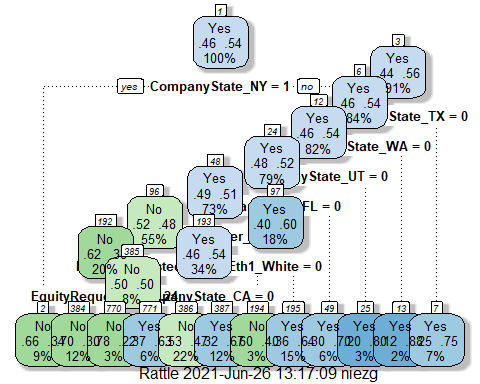
shark\_recipe2 = recipe(Deal\_Yes ~., train2) %>%  
step\_dummy(all\_predictors(), -all\_outcomes())  
  
tree\_model2 = decision\_tree() %>%  
set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
set\_mode("classification")  
  
shark\_wflow =  
workflow() %>%  
add\_model(tree\_model2) %>%  
add\_recipe(shark\_recipe2)  
1

## [1] 1

shark\_fit2 = fit(shark\_wflow, train2)

## Warning: The following variables are not factor vectors and will be ignored:  
## `NumberOfPresenters`, `EquityRequested`

tree = shark\_fit2 %>%  
pull\_workflow\_fit() %>%  
pluck("fit")  
  
#plot the tree  
fancyRpartPlot(tree, tweak = 3)



shark\_fit2$fit$fit$fit$cptable

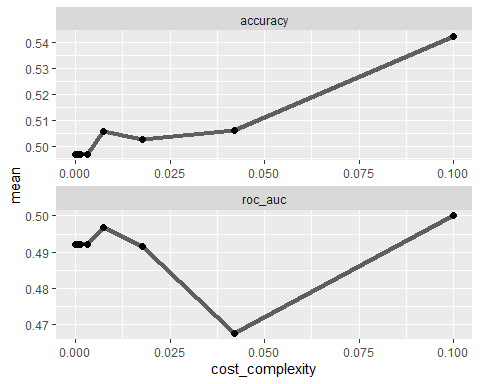
## CP nsplit rel error xerror xstd  
## 1 0.05921053 0 1.0000000 1.0000000 0.05972352  
## 2 0.01754386 1 0.9407895 1.0723684 0.05992727  
## 3 0.01644737 8 0.8026316 1.0131579 0.05978025  
## 4 0.01315789 10 0.7697368 1.0131579 0.05978025  
## 5 0.01000000 11 0.7565789 0.9934211 0.05969186

**Tuning**

set.seed(123)  
folds = vfold\_cv(train2, v = 5)

shark\_recipe3 = recipe(Deal\_Yes ~., train2) %>%  
step\_dummy(all\_nominal(),-all\_outcomes())  
  
tree\_model3 = decision\_tree(cost\_complexity = tune()) %>%  
set\_engine("rpart", model = TRUE) %>% #don't forget the model = TRUE flag  
set\_mode("classification")  
  
tree\_grid3 = grid\_regular(cost\_complexity(),  
levels = 25) #try 25 sensible values for cp  
  
shark\_wflow3 =  
workflow() %>%  
add\_model(tree\_model3) %>%  
add\_recipe(shark\_recipe3)  
  
tree\_res =  
shark\_wflow3 %>%  
tune\_grid(  
resamples = folds,  
grid = tree\_grid3  
)

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

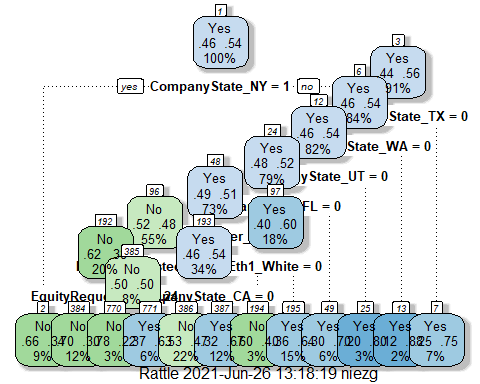
## # A tibble: 1 x 2  
## cost\_complexity .config   
## <dbl> <chr>   
## 1 0.1 Preprocessor1\_Model25

final\_wf =  
shark\_wflow %>%  
finalize\_workflow(best\_tree)

final\_fit2 = fit(final\_wf, train2)

## Warning: The following variables are not factor vectors and will be ignored:  
## `NumberOfPresenters`, `EquityRequested`

tree2 = final\_fit2 %>%  
pull\_workflow\_fit() %>%  
pluck("fit")  
  
fancyRpartPlot(tree2, tweak = 3)



treepred = predict(final\_fit2, train2, type = "class")  
  
confusionMatrix(treepred$.pred\_class,train2$Deal\_Yes,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 99 62  
## Yes 53 118  
##   
## Accuracy : 0.6536   
## 95% CI : (0.5997, 0.7047)  
## No Information Rate : 0.5422   
## P-Value [Acc > NIR] : 2.445e-05   
##   
## Kappa : 0.3055   
##   
## Mcnemar's Test P-Value : 0.4557   
##   
## Sensitivity : 0.6556   
## Specificity : 0.6513   
## Pos Pred Value : 0.6901   
## Neg Pred Value : 0.6149   
## Prevalence : 0.5422   
## Detection Rate : 0.3554   
## Detection Prevalence : 0.5151   
## Balanced Accuracy : 0.6534   
##   
## '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 Yes   
## 3 Yes   
## 4 No   
## 5 No   
## 6 Yes

confusionMatrix(treepred\_test$.pred\_class,test2$Deal\_Yes,positive="Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 28 28  
## Yes 38 50  
##   
## Accuracy : 0.5417   
## 95% CI : (0.4567, 0.6249)  
## No Information Rate : 0.5417   
## P-Value [Acc > NIR] : 0.5342   
##   
## Kappa : 0.066   
##   
## Mcnemar's Test P-Value : 0.2679   
##   
## Sensitivity : 0.6410   
## Specificity : 0.4242   
## Pos Pred Value : 0.5682   
## Neg Pred Value : 0.5000   
## Prevalence : 0.5417   
## Detection Rate : 0.3472   
## Detection Prevalence : 0.6111   
## Balanced Accuracy : 0.5326   
##   
## 'Positive' Class : Yes   
##

### XG Boost Approach

set.seed(123)   
shark\_split = initial\_split(sharktank\_clean, prop = 0.7, strata = Deal\_Yes) #70% in training  
train = training(shark\_split)   
test = testing(shark\_split)

use\_xgboost(Deal\_Yes ~., train)

## xgboost\_recipe <-   
## recipe(formula = Deal\_Yes ~ ., data = train) %>%   
## step\_novel(all\_nominal(), -all\_outcomes()) %>%   
## step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE) %>%   
## step\_zv(all\_predictors())   
##   
## xgboost\_spec <-   
## boost\_tree(trees = tune(), min\_n = tune(), tree\_depth = tune(), learn\_rate = tune(),   
## loss\_reduction = tune(), sample\_size = tune()) %>%   
## set\_mode("classification") %>%   
## set\_engine("xgboost")   
##   
## xgboost\_workflow <-   
## workflow() %>%   
## add\_recipe(xgboost\_recipe) %>%   
## add\_model(xgboost\_spec)   
##   
## set.seed(23308)  
## xgboost\_tune <-  
## tune\_grid(xgboost\_workflow, resamples = stop("add your rsample object"), grid = stop("add number of candidate points"))

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

start\_time = Sys.time() #for timing  
  
xgboost\_recipe <-   
 recipe(formula = Deal\_Yes ~ ., data = train) %>%   
 #step\_novel(all\_nominal(), -all\_outcomes()) %>%   
 step\_dummy(all\_nominal(), -all\_outcomes(), one\_hot = TRUE) %>%   
 step\_zv(all\_predictors())   
  
xgboost\_spec <-   
 boost\_tree(trees = tune(), min\_n = tune(), tree\_depth = tune(), learn\_rate = tune(),   
 loss\_reduction = tune(), sample\_size = tune()) %>%   
 set\_mode("classification") %>%   
 set\_engine("xgboost")   
  
xgboost\_workflow <-   
 workflow() %>%   
 add\_recipe(xgboost\_recipe) %>%   
 add\_model(xgboost\_spec)   
  
set.seed(77680)  
xgboost\_tune <-  
 tune\_grid(xgboost\_workflow, resamples = folds, grid = 25)  
  
end\_time = Sys.time()  
end\_time - start\_time

## Time difference of 2.93447 mins

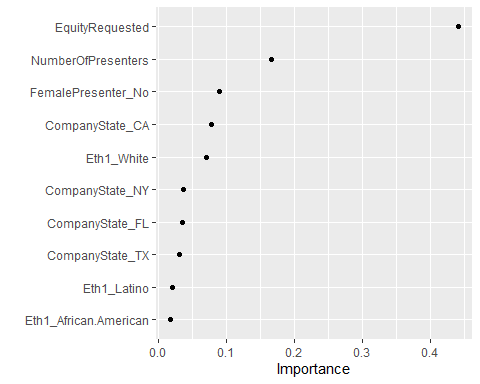
best\_xgb = select\_best(xgboost\_tune, "accuracy")  
  
final\_xgb = finalize\_workflow(  
 xgboost\_workflow,  
 best\_xgb  
)  
  
final\_xgb

## == Workflow ====================================================================  
## Preprocessor: Recipe  
## Model: boost\_tree()  
##   
## -- Preprocessor ----------------------------------------------------------------  
## 2 Recipe Steps  
##   
## \* step\_dummy()  
## \* step\_zv()  
##   
## -- Model -----------------------------------------------------------------------  
## Boosted Tree Model Specification (classification)  
##   
## Main Arguments:  
## trees = 1307  
## min\_n = 3  
## tree\_depth = 5  
## learn\_rate = 0.0133610881803273  
## loss\_reduction = 0.0743723901817429  
## sample\_size = 0.79412292109932  
##   
## Computational engine: xgboost

final\_xgb\_fit = fit(final\_xgb, train)

## [13:21:16] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.

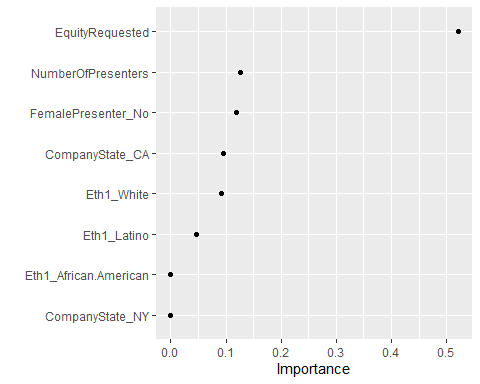
final\_xgb\_fit %>% pull\_workflow\_fit() %>% vip(geom = "point")



final\_xgb\_fittest = fit(final\_xgb, test)

## [13:21:20] WARNING: amalgamation/../src/learner.cc:1095: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval\_metric if you'd like to restore the old behavior.

final\_xgb\_fittest %>% pull\_workflow\_fit() %>% vip(geom = "point")



trainpredxgb = predict(final\_xgb\_fit, train)  
head(trainpredxgb)

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

confusionMatrix(trainpredxgb$.pred\_class, train$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 101 35  
## Yes 51 145  
##   
## Accuracy : 0.741   
## 95% CI : (0.6903, 0.7873)  
## No Information Rate : 0.5422   
## P-Value [Acc > NIR] : 6.678e-14   
##   
## Kappa : 0.4739   
##   
## Mcnemar's Test P-Value : 0.1058   
##   
## Sensitivity : 0.8056   
## Specificity : 0.6645   
## Pos Pred Value : 0.7398   
## Neg Pred Value : 0.7426   
## Prevalence : 0.5422   
## Detection Rate : 0.4367   
## Detection Prevalence : 0.5904   
## Balanced Accuracy : 0.7350   
##   
## 'Positive' Class : Yes   
##

testpredxgb = predict(final\_xgb\_fit, test)

confusionMatrix(testpredxgb$.pred\_class, test$Deal\_Yes,   
 positive = "Yes")

## Confusion Matrix and Statistics  
##   
## Reference  
## Prediction No Yes  
## No 27 30  
## Yes 39 48  
##   
## Accuracy : 0.5208   
## 95% CI : (0.4361, 0.6047)  
## No Information Rate : 0.5417   
## P-Value [Acc > NIR] : 0.7213   
##   
## Kappa : 0.0247   
##   
## Mcnemar's Test P-Value : 0.3355   
##   
## Sensitivity : 0.6154   
## Specificity : 0.4091   
## Pos Pred Value : 0.5517   
## Neg Pred Value : 0.4737   
## Prevalence : 0.5417   
## Detection Rate : 0.3333   
## Detection Prevalence : 0.6042   
## Balanced Accuracy : 0.5122   
##   
## 'Positive' Class : Yes   
##