Homework 6

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.2.0 --

## v broom        0.8.0     v recipes      0.2.0
## v dials        0.1.1     v rsample      0.1.1
## v dplyr        1.0.9     v tibble       3.1.7
## v ggplot2      3.3.6     v tidyr        1.2.0
## v infer        1.0.0     v tune         0.2.0
## v modeldata    0.1.1     v workflows    0.2.6
## v parsnip      0.2.1     v workflowsets 0.2.1
## v purrr        0.3.4     v yardstick    0.0.9

## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Search for functions across packages at https://www.tidymodels.org/find/
```

```
library(ISLR)
library(ISLR2)
```

```
##
## Attaching package: 'ISLR2'

## The following objects are masked from 'package:ISLR':
##
##     Auto, Credit
```

```
library(tidyverse)
```

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

## v readr   2.1.2     v forcats 0.5.1
## v stringr 1.4.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'
```

```
## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack
```

```
## Loaded glmnet 4.1-4
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'
```

```
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(rpart.plot)
```

```
## Loading required package: rpart
```

```
##
## Attaching package: 'rpart'
```

```
## The following object is masked from 'package:dials':
##
##     prune
```

```
library(ranger)
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(vip)
```

```
##
## Attaching package: 'vip'
```

```
## The following object is masked from 'package:utils':
##
##     vi
```

```
library(xgboost)
```

```
##
## Attaching package: 'xgboost'

## The following object is masked from 'package:dplyr':
##
##     slice
```

```
tidymodels_prefer()
```

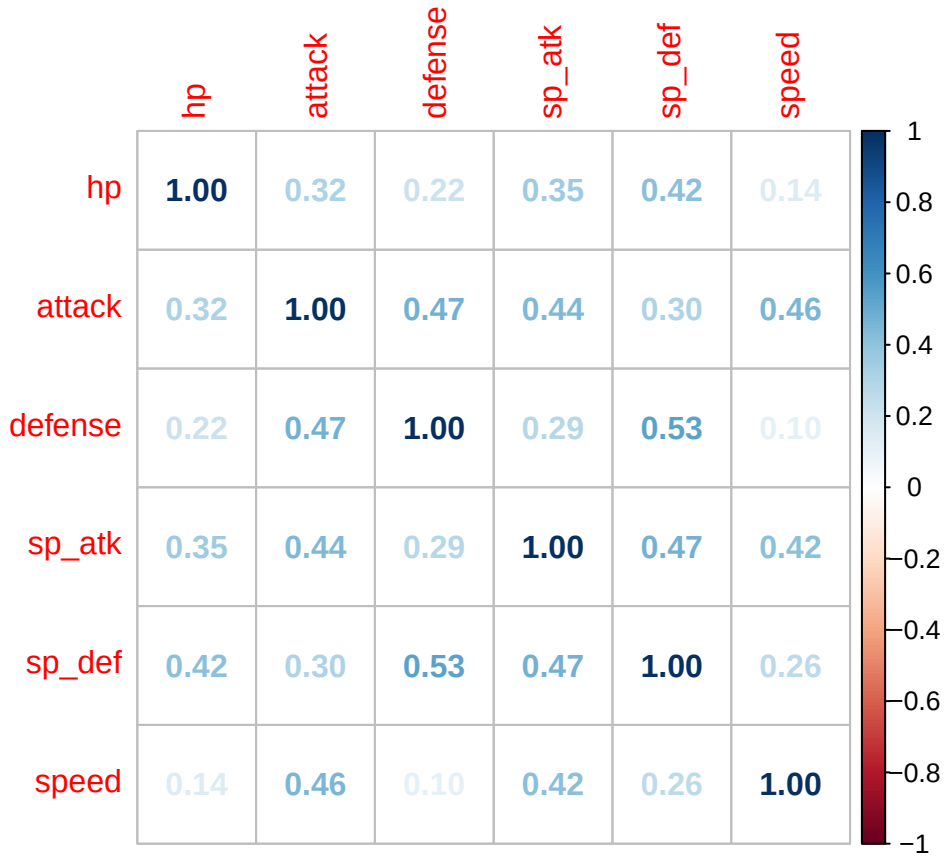
```
poke <- read.csv('Pokemon.csv')
pokemon <- clean_names(poke)
```

```
pokemon <- pokemon%>%filter(type_1 == 'Bug' | type_1== 'Fire' | type_1== 'Grass' | type_1== 'Normal' | t

pokemon$legendary <- as.factor(pokemon$legendary)

pokemon$type_1 <- as.factor(pokemon$type_1)
pokemon$generation <- as.factor(pokemon$generation)
```

```
set.seed(608)
pokeSplit <- initial_split(pokemon, prop = 0.80,
                                strata = type_1)
poke_train <- training(pokeSplit)
poke_test <- testing(pokeSplit)
```

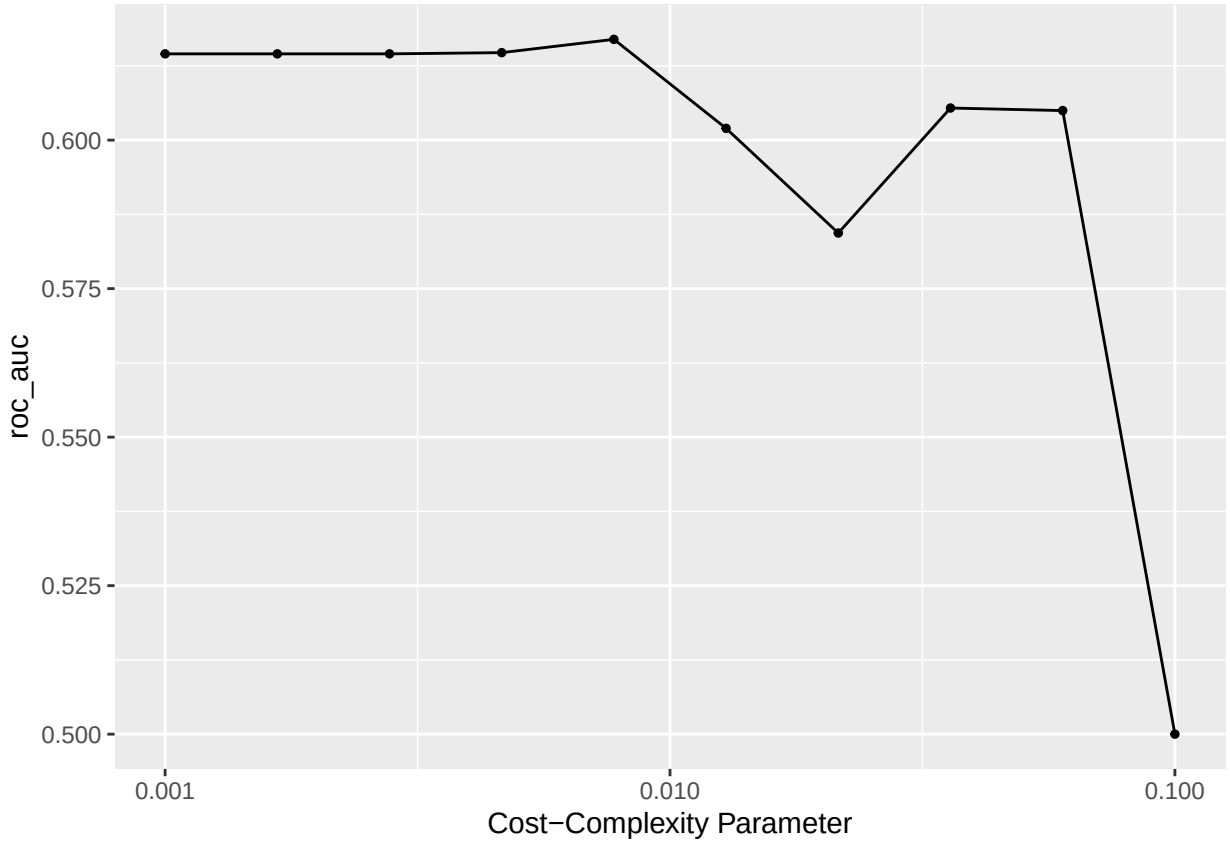
```
poke_fold <- vfold_cv(poke_train, v = 5, strata = type_1)
pokeRecipe <- recipe(type_1 ~ legendary + generation + sp_atk + attack + speed  + defense + hp + sp_def
```

```
poke_train %>% select(where(is.numeric)) %>% select(-x, -total) %>% cor() %>% corrplot(method = 'number
```



None of the relationships are very high, however, defense and hp, speed and defense, speed and hp, sp-def and speed, all have especially low correlations

```
tree_spec <- decision_tree() %>%
  set_engine("rpart")

class_tree_spec <- tree_spec %>%
  set_mode("classification")

class_tree_wf <- workflow() %>%
  add_model(class_tree_spec %>% set_args(cost_complexity = tune())) %>%
  add_recipe(pokeRecipe)

param_grid <- grid_regular(cost_complexity(range = c(-3, -1)), levels = 10)

tune_res <- tune_grid(
  class_tree_wf,
  resamples = poke_fold,
  grid = param_grid,
  metrics = metric_set(roc_auc)
)

autoplot(tune_res)
```



It performs better with a smaller cost-complexity parameter

```
collect_metrics(tune_res)
```

```
## # A tibble: 10 x 7
##    cost_complexity .metric .estimator  mean     n std_err .config
##              <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1         0.001   roc_auc hand_till  0.615     5  0.0184 Preprocessor1_Model01
##  2         0.00167 roc_auc hand_till  0.615     5  0.0184 Preprocessor1_Model02
##  3         0.00278 roc_auc hand_till  0.615     5  0.0184 Preprocessor1_Model03
##  4         0.00464 roc_auc hand_till  0.615     5  0.0201 Preprocessor1_Model04
##  5         0.00774 roc_auc hand_till  0.617     5  0.0184 Preprocessor1_Model05
##  6         0.0129  roc_auc hand_till  0.602     5  0.0226 Preprocessor1_Model06
##  7         0.0215  roc_auc hand_till  0.584     5  0.0204 Preprocessor1_Model07
##  8         0.0359  roc_auc hand_till  0.605     5  0.0132 Preprocessor1_Model08
##  9         0.0599  roc_auc hand_till  0.605     5  0.0126 Preprocessor1_Model09
## 10         0.1     roc_auc hand_till  0.5       5  0      Preprocessor1_Model10
```

```
arrange(tune_res)
```

```
## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 x 4
##   splits           id    .metrics          .notes
##   <list>           <chr> <list>            <list>
## 1 <split [289/75]> Fold1 <tibble [10 x 5]> <tibble [0 x 3]>
```

```
## 2 <split [291/73]> Fold2 <tibble [10 x 5]> <tibble [0 x 3]>
## 3 <split [291/73]> Fold3 <tibble [10 x 5]> <tibble [0 x 3]>
## 4 <split [292/72]> Fold4 <tibble [10 x 5]> <tibble [0 x 3]>
## 5 <split [293/71]> Fold5 <tibble [10 x 5]> <tibble [0 x 3]>
```

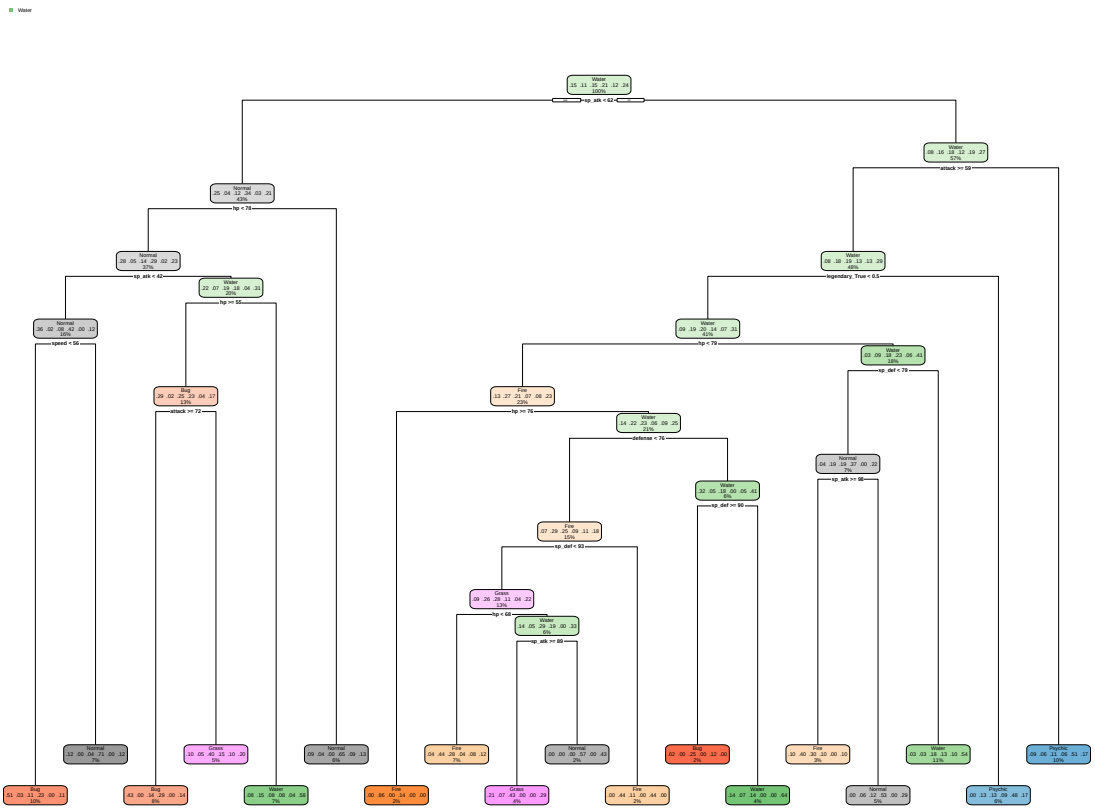
```
best_pruned <- select_best(tune_res)
class_tree_final <- finalize_workflow(class_tree_wf, best_pruned)
class_final_fit <- fit(class_tree_final, data = poke_train)
augment(class_final_fit, new_data = poke_test) %>% roc_auc(type_1, estimate = c(.pred_Bug, .pred_Fire,
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.628
```

auc = 0.6277206

```
class_final_fit %>%
  extract_fit_engine() %>%
  rpart.plot(roundint = FALSE)
```

```
## Warning: labs do not fit even at cex 0.15, there may be some overplotting
```

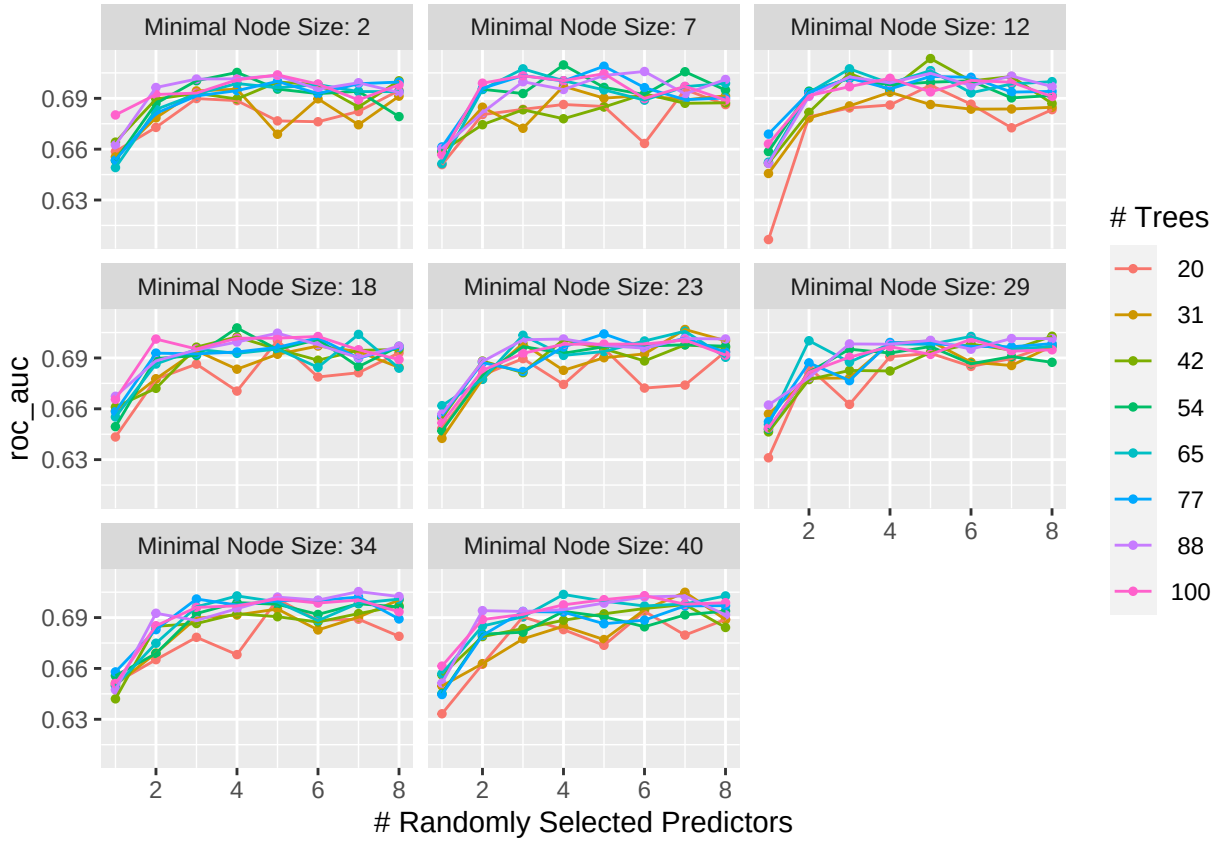


```
rf_spec <- rand_forest(mtry = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("classification")

rf_wf <- workflow() %>%
  add_model(rf_spec %>% set_args(trees = tune(), min_n = tune())) %>%
  add_recipe(pokeRecipe)

rf_grid <-  grid_regular(mtry(range = c(1, 8)), trees(range = c(20, 100)), min_n(range = c(2, 40)), leve
```

mtry = number of predictors sampled at each split trees = number of trees min_n = min number of observations required for another split

at mtry =8 we are using all the predictors in each split. Therefore we can not go higher than 8, similarly, less than 1 is also not plausible because we won't have a useful model if 0 predictors are sampled at each split.

```
tune_res_rf <- tune_grid(
  rf_wf,
  resamples = poke_fold,
  grid = rf_grid,
  metrics = metric_set(roc_auc)
)

autoplot(tune_res_rf)
```



```
collect_metrics(tune_res_rf)
```

```
## # A tibble: 512 x 9
##     mtry trees min_n .metric .estimator  mean     n std_err .config
##    <int> <int> <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1     1    20     2 roc_auc hand_till  0.659     5 0.0203  Preprocessor1_Model~
##  2     2    20     2 roc_auc hand_till  0.673     5 0.0211  Preprocessor1_Model~
##  3     3    20     2 roc_auc hand_till  0.690     5 0.00448 Preprocessor1_Model~
##  4     4    20     2 roc_auc hand_till  0.689     5 0.0181  Preprocessor1_Model~
##  5     5    20     2 roc_auc hand_till  0.677     5 0.0202  Preprocessor1_Model~
##  6     6    20     2 roc_auc hand_till  0.676     5 0.0112  Preprocessor1_Model~
##  7     7    20     2 roc_auc hand_till  0.682     5 0.0163  Preprocessor1_Model~
##  8     8    20     2 roc_auc hand_till  0.695     5 0.0190  Preprocessor1_Model~
##  9     1    31     2 roc_auc hand_till  0.655     5 0.0170  Preprocessor1_Model~
## 10     2    31     2 roc_auc hand_till  0.679     5 0.0194  Preprocessor1_Model~
## # ... with 502 more rows
```

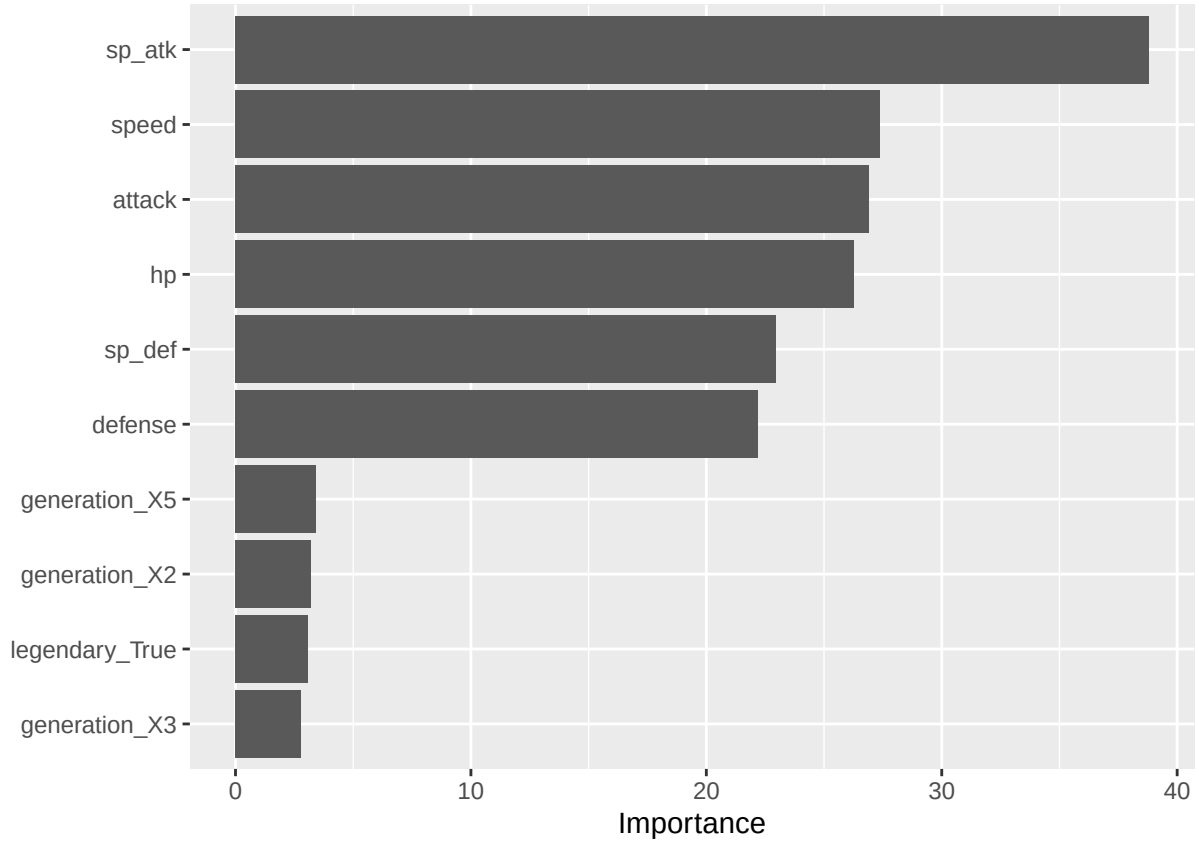
```
arrange(tune_res_rf)
```

```
## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 x 4
##   splits           id    .metrics           .notes
##   <list>           <chr> <list>             <list>
## 1 <split [289/75]> Fold1 <tibble [512 x 7]> <tibble [0 x 3]>
## 2 <split [291/73]> Fold2 <tibble [512 x 7]> <tibble [0 x 3]>
## 3 <split [291/73]> Fold3 <tibble [512 x 7]> <tibble [0 x 3]>
## 4 <split [292/72]> Fold4 <tibble [512 x 7]> <tibble [0 x 3]>
## 5 <split [293/71]> Fold5 <tibble [512 x 7]> <tibble [0 x 3]>
```

```
best_rf<- select_best(tune_res_rf)
rf_final <- finalize_workflow(rf_wf, best_rf)
rf_final_fit <- fit(rf_final, data = poke_train)
augment(rf_final_fit, new_data = poke_test) %>% roc_auc(type_1, estimate = c(.pred_Bug, .pred_Fire, .pr
```

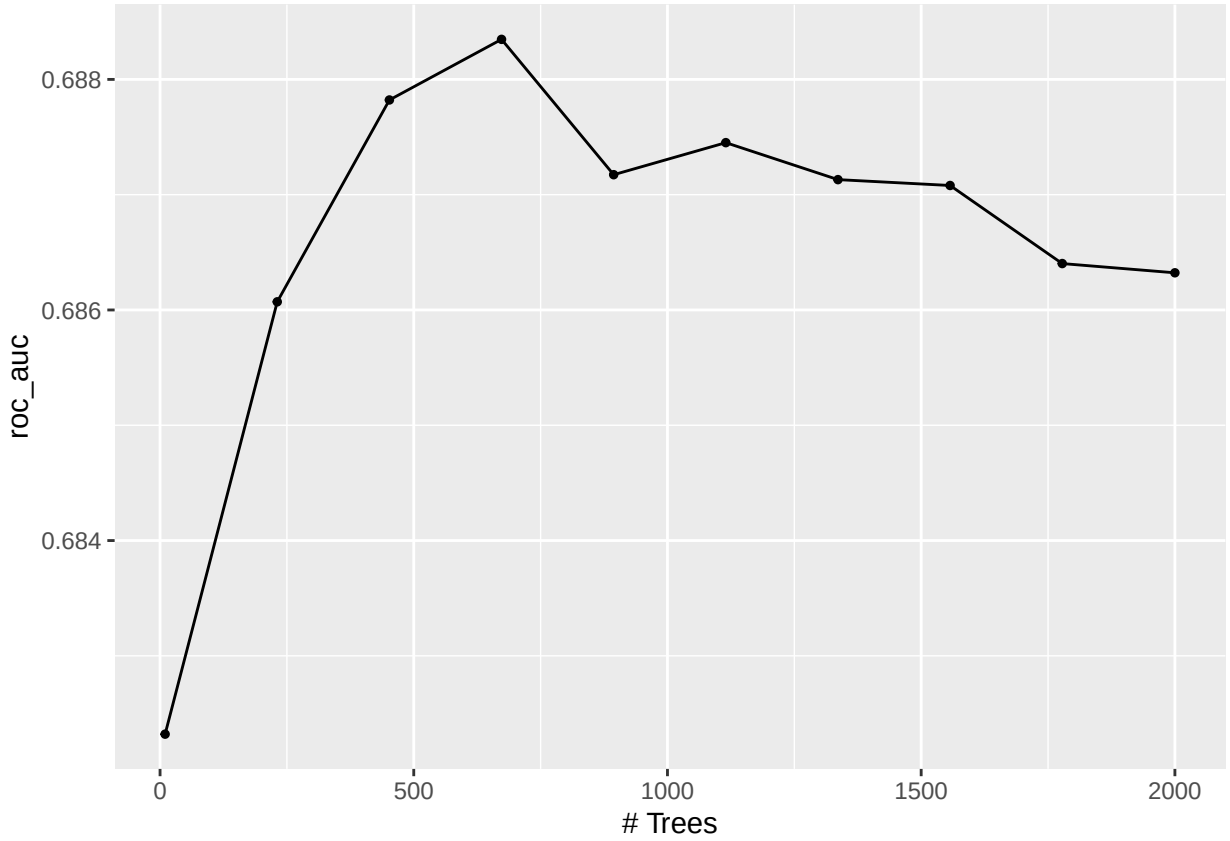
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.678
```

```
rf_final_fit %>%
  extract_fit_parsnip()%>%
  vip()
```



special attack and attack were the most useful, generation was the leasr useful. This is expected since type heavily influences attack and special attack stats.

```
boost_spec <- boost_tree(trees = tune(), tree_depth = 4) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

boost_wf <- workflow() %>%
  add_model(boost_spec %>% set_args(trees = tune())) %>%
  add_recipe(pokeRecipe)

boost_grid <-  grid_regular(trees(range = c(10, 2000)), levels = 10)

boost_tune <- tune_grid(
  boost_wf,
  resamples = poke_fold,
  grid = boost_grid,
  metrics = metric_set(roc_auc)
)
autoplot(boost_tune)
```




```
collect_metrics(boost_tune)
```

```
## # A tibble: 10 x 7
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1    10 roc_auc hand_till  0.682     5  0.0132 Preprocessor1_Model01
##  2   231 roc_auc hand_till  0.686     5  0.0157 Preprocessor1_Model02
##  3   452 roc_auc hand_till  0.688     5  0.0170 Preprocessor1_Model03
##  4   673 roc_auc hand_till  0.688     5  0.0179 Preprocessor1_Model04
##  5   894 roc_auc hand_till  0.687     5  0.0186 Preprocessor1_Model05
##  6  1115 roc_auc hand_till  0.687     5  0.0184 Preprocessor1_Model06
##  7  1336 roc_auc hand_till  0.687     5  0.0189 Preprocessor1_Model07
##  8  1557 roc_auc hand_till  0.687     5  0.0193 Preprocessor1_Model08
##  9  1778 roc_auc hand_till  0.686     5  0.0196 Preprocessor1_Model09
## 10  2000 roc_auc hand_till  0.686     5  0.0195 Preprocessor1_Model10
```

```
arrange(boost_tune)
```

```
## # Tuning results
## # 5-fold cross-validation using stratification
## # A tibble: 5 x 4
##   splits           id    .metrics          .notes
##   <list>           <chr> <list>            <list>
## 1 <split [289/75]> Fold1 <tibble [10 x 5]> <tibble [0 x 3]>
## 2 <split [291/73]> Fold2 <tibble [10 x 5]> <tibble [0 x 3]>
```

```
## 3 <split [291/73]> Fold3 <tibble [10 x 5]> <tibble [0 x 3]>
## 4 <split [292/72]> Fold4 <tibble [10 x 5]> <tibble [0 x 3]>
## 5 <split [293/71]> Fold5 <tibble [10 x 5]> <tibble [0 x 3]>
```

```
best_boost<- select_best(boost_tune)
boost_final <- finalize_workflow(boost_wf, best_boost)
boost_final_fit <- fit(boost_final, data = poke_train)
augment(boost_final_fit, new_data = poke_test) %>% roc_auc(type_1, estimate = c(.pred_Bug, .pred_Fire,
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.677
```

0.6771615