# Homework Assignment 6

Vasiliy Ostapenko (774 970 8)

May 18, 2022

## Exercise 1

### Preprocess

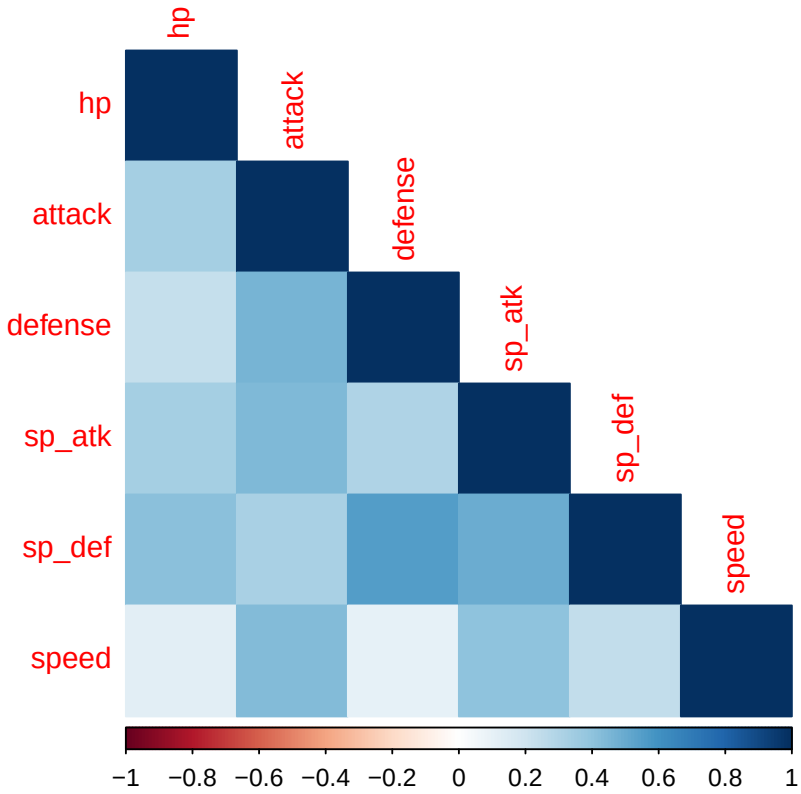
```
DATA_FOLDER = "./data"
POKEMON_FNAME = file.path(DATA_FOLDER, "Pokemon.csv")
df = read.csv(POKEMON_FNAME)

df = df %>% clean_names()

df = df[df$type_1 %in%
          c("Bug", "Fire", "Grass",
            "Normal", "Water", "Psychic"), ] %>% copy()

df$type_1 = as.factor(df$type_1)
df$legendary = as.factor(df$legendary)
df$generation = as.factor(df$generation)
```

### Split, Folds, Recipe

```
split = df %>%
  initial_split(prop=0.8, strata="type_1")

train = training(split)
test = testing(split)

folds = vfold_cv(train, v=5, strata="type_1")

rec = recipe(type_1 ~ legendary + generation + sp_atk + attack +
               speed + defense + hp + sp_def, data=train) %>%
  step_dummy(c("legendary", "generation")) %>%
  step_normalize(all_predictors())
```

## Exercise 2

```
corrplot(cor(df[ , names(df) %in% c("sp_atk", "attack", "speed",
                                    "defense", "hp", "sp_def")]),
         method="color", type="lower")
```

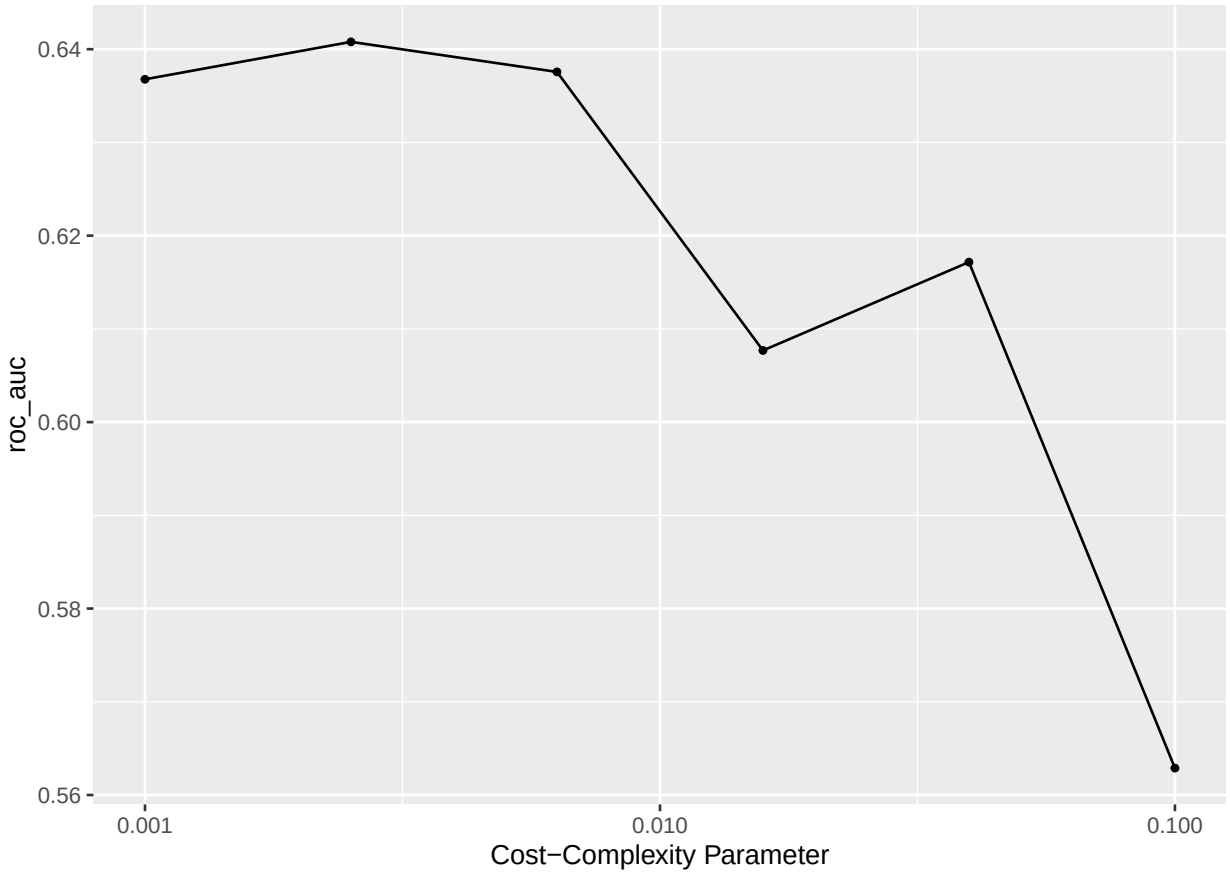



Plotting only numeric columns for the correlation matrix plot. It looks like sp_def is correlated with defense and sp_atk is correlated with attack, whcih makes sense. It would also make sense that higher speed would be granted to instances of high attack and low defense.

## Exercise 3

```
mod = decision_tree(cost_complexity=tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")

work = workflow() %>%
  add_model(mod) %>%
  add_recipe(rec)

grid = grid_regular(cost_complexity(range=c(-3, -1)), levels=6)

tune = work %>%
  tune_grid(resamples=folds, grid=grid, metrics=metric_set(roc_auc))
saveRDS(tune, "./data/tune.rds")
```

```
tune = readRDS("./data/tune.rds")
autoplot(tune, metric="roc_auc")
```




A single decision tree performs better with a smaller complexity penalty.

## Exercise 4

```
tune %>% collect_metrics() %>% arrange(desc(mean))
```

```
## # A tibble: 6 x 7
##   cost_complexity .metric .estimator  mean     n std_err .config
##             <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1         0.00251 roc_auc hand_till  0.641     5  0.0139 Preprocessor1_Model2
## 2         0.00631 roc_auc hand_till  0.638     5  0.0144 Preprocessor1_Model3
## 3         0.001   roc_auc hand_till  0.637     5  0.0164 Preprocessor1_Model1
## 4         0.0398  roc_auc hand_till  0.617     5  0.0106 Preprocessor1_Model5
## 5         0.0158  roc_auc hand_till  0.608     5  0.0169 Preprocessor1_Model4
## 6         0.1     roc_auc hand_till  0.563     5  0.0259 Preprocessor1_Model6
```

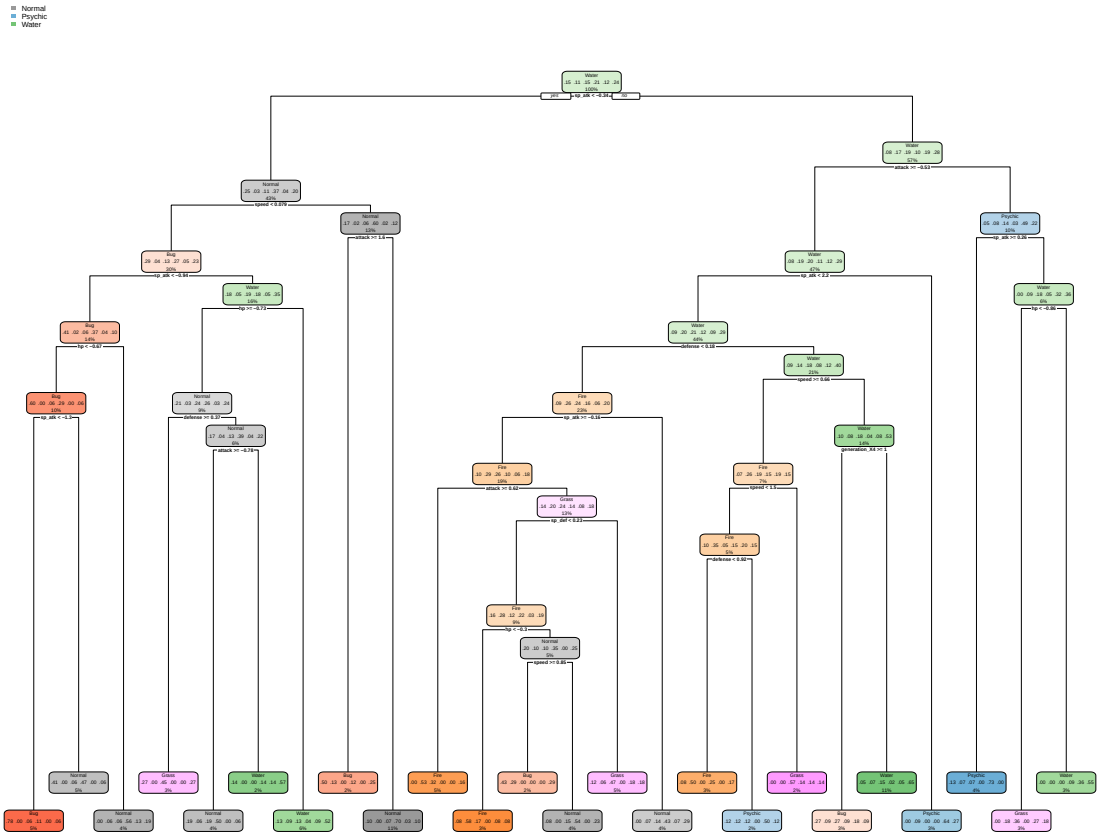
Best-performing decision tree scored a mean roc_auc of 0.6408 over the five folds.

## Exercise 5

```
best = tune %>%
  select_best("roc_auc")

final_work = work %>%
  finalize_workflow(best)
```

```
best_fit = final_work %>%
  fit(train)
saveRDS(best_fit, "./data/best_fit.rds")
```

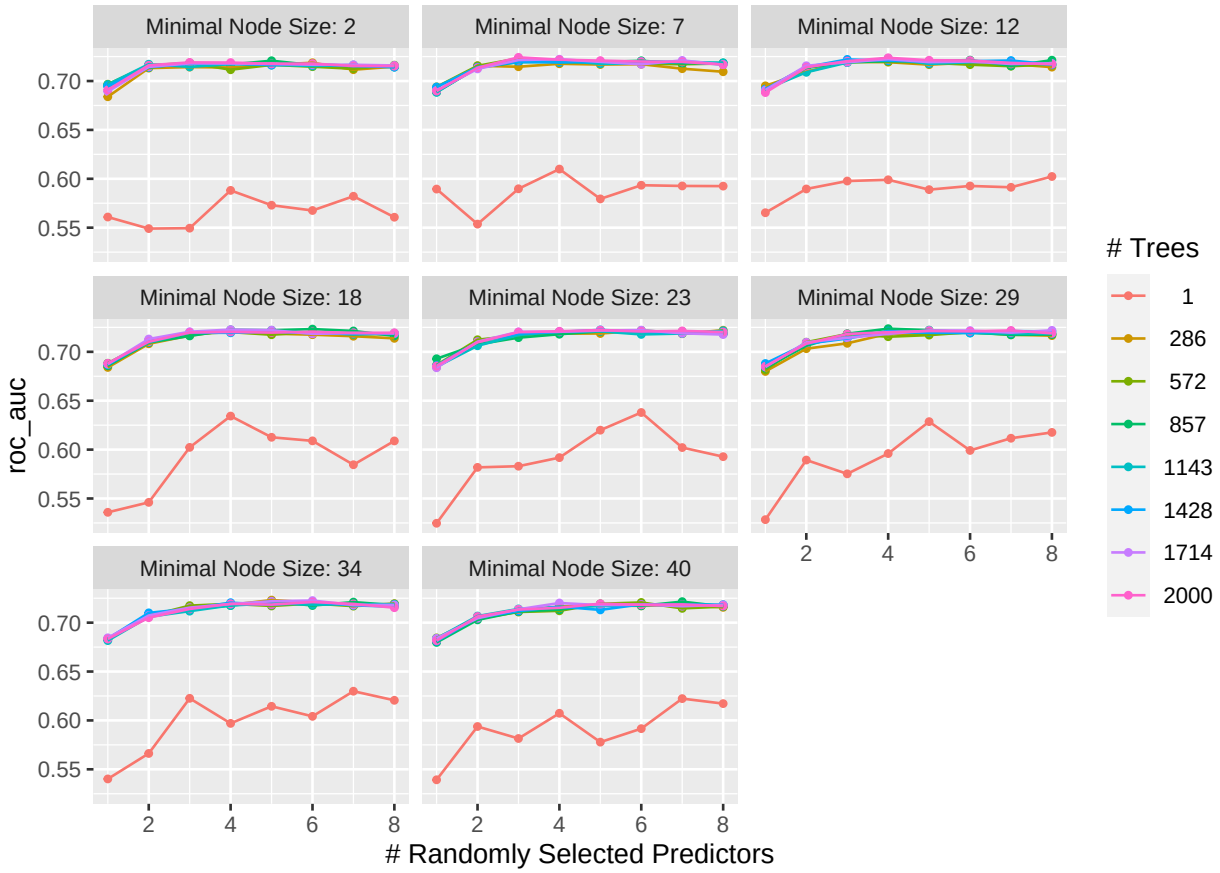
```
best_fit = readRDS("./data/best_fit.rds")
rpart.plot( extract_fit_parsnip(best_fit)$fit )
```




## Exercise 6-7

```
mod2 = rand_forest(mtry=tune(), trees=tune(), min_n=tune()) %>%
  set_engine("ranger", importance="impurity") %>%
  set_mode("classification")

work2 = workflow() %>%
  add_model(mod2) %>%
  add_recipe(rec)

grid2 = grid_regular(mtry(range=c(1, 8)), trees(), min_n(), levels=8)

tune2 = work2 %>%
  tune_grid(resamples=folds, grid=grid2, metrics=metric_set(roc_auc))
saveRDS(tune2, "./data/tune2.rds")
```

```
tune2 = readRDS("./data/tune2.rds")
autoplot(tune2, metric="roc_auc")
```




A random forest model creates many independent decision trees and uses all of their predictions in combination to make a final prediction. The parameter mtry represents the number of predictors (between 1 and all) to be sampled for use in the trees. mtry is limited between using 1 and all eight predictors. We cannot use no or negative predictors. Likewise, we cannot use more predictors than we have. The trees parameter stands for the total number of trees created. Finally, the min_n parameter is an integer that specifies the minimum number of data points needed to split a tree node into further leaves.

Using more trees and more randomly selected predictors seems to yield better performance. Changing minimum node size doesn't seem to make a difference.

## Exercise 8

```
tune2 %>% collect_metrics() %>% arrange(desc(mean))
```

```
## # 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     3  2000     7 roc_auc hand_till  0.724     5  0.0131 Preprocessor1_Model~
## 2     4  2000    12 roc_auc hand_till  0.724     5  0.0138 Preprocessor1_Model~
## 3     4   857    29 roc_auc hand_till  0.724     5  0.0158 Preprocessor1_Model~
## 4     6   857    18 roc_auc hand_till  0.723     5  0.0141 Preprocessor1_Model~
## 5     4  1143    12 roc_auc hand_till  0.723     5  0.0130 Preprocessor1_Model~
```

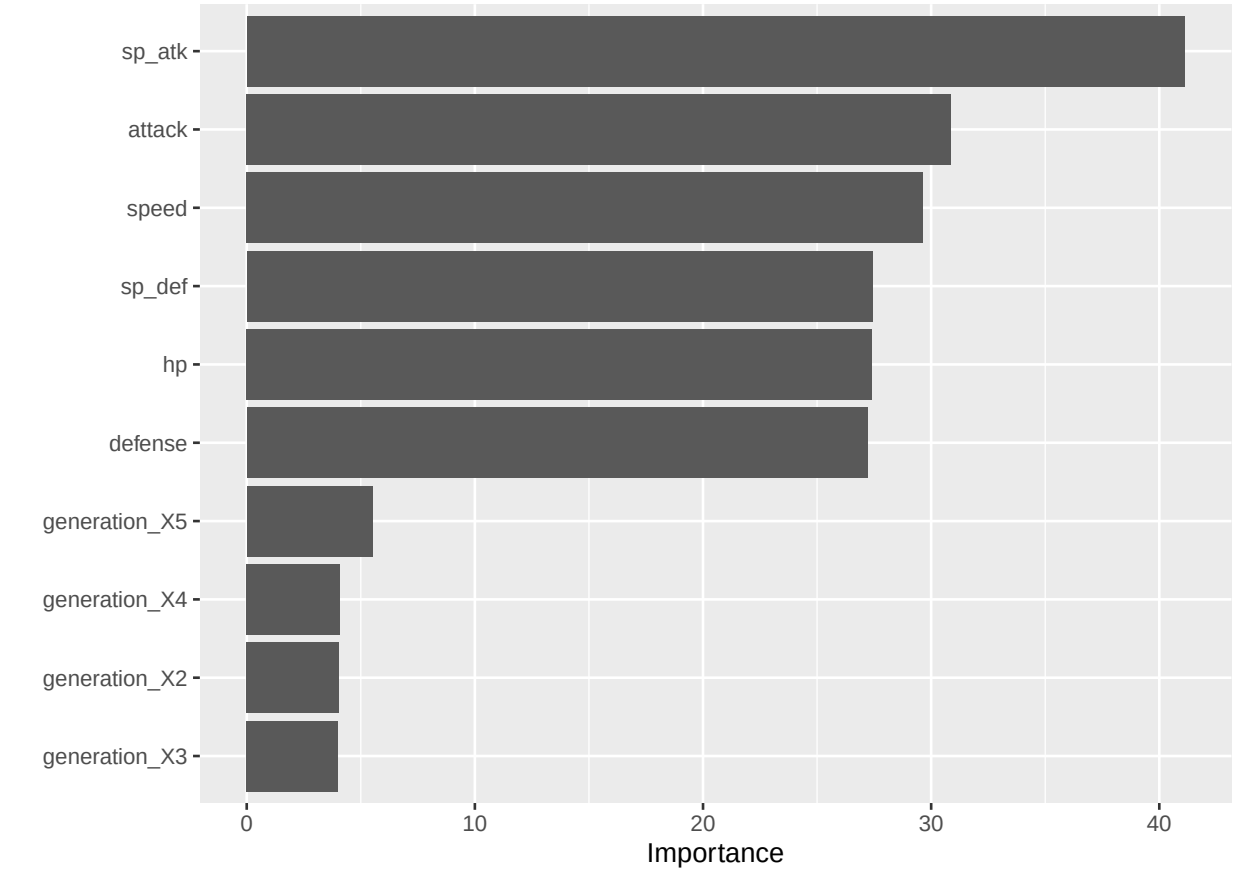
```
##  6     3   572     7 roc_auc hand_till  0.723     5  0.0137 Preprocessor1_Model~
##  7     5   286    34 roc_auc hand_till  0.723     5  0.0164 Preprocessor1_Model~
##  8     4  1714    18 roc_auc hand_till  0.723     5  0.0144 Preprocessor1_Model~
##  9     6  1714    34 roc_auc hand_till  0.723     5  0.0156 Preprocessor1_Model~
## 10     4   857    18 roc_auc hand_till  0.723     5  0.0151 Preprocessor1_Model~
## # ... with 502 more rows
```

Best-performing random forest scored a mean roc_auc of 0.7242 over the five folds.

## Exercise 9

```
best2 = tune2 %>%
  select_best("roc_auc")

final_work2 = work2 %>%
  finalize_workflow(best2)

best_fit2 = final_work2 %>%
  fit(train)
saveRDS(best_fit2, "./data/best_fit2.rds")
```

```
best_fit2 = readRDS("./data/best_fit2.rds")
best_fit2 %>%
  extract_fit_parsnip() %>%
  vip()
```

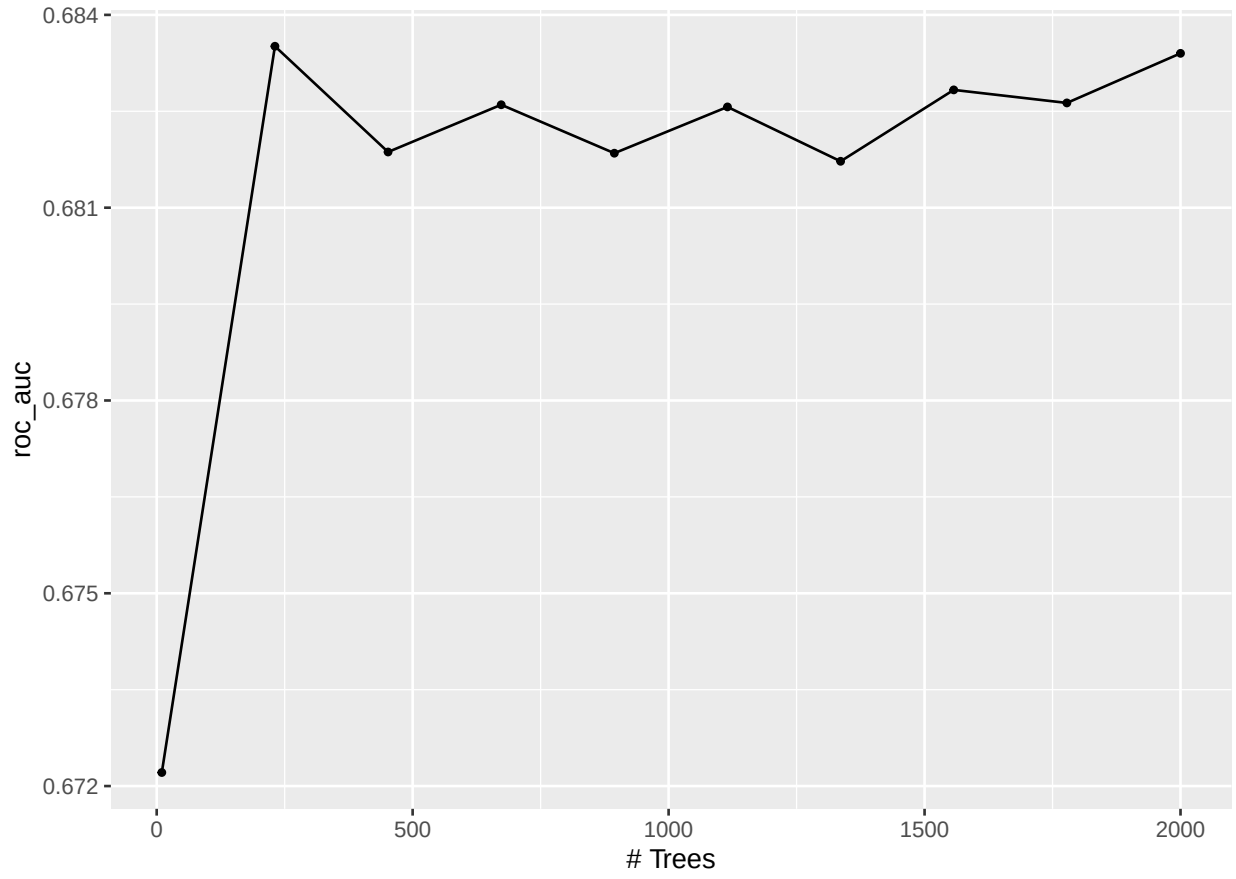


sp_atk, attack, and speed are most useful for prediction, while generation is least useful.

### Exercise 10

```
mod3 = boost_tree(trees=tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

work3 = workflow() %>%
  add_model(mod3) %>%
  add_recipe(rec)

grid3 = grid_regular(trees(range=c(10, 2000)), levels=10)

tune3 = work3 %>%
  tune_grid(resamples=folds, grid=grid3, metrics=metric_set(roc_auc))
saveRDS(tune3, "./data/tune3.rds")
```

```
tune3 = readRDS("./data/tune3.rds")
autoplot(tune3, metric="roc_auc")
```




More trees generally seem to yield higher performance for a boosted trees model.

```
tune3 %>% collect_metrics() %>% arrange(desc(mean))
```

```
## # A tibble: 10 x 7
```

```
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1   231 roc_auc hand_till  0.684     5  0.0110 Preprocessor1_Model02
##  2  2000 roc_auc hand_till  0.683     5  0.0124 Preprocessor1_Model10
##  3  1557 roc_auc hand_till  0.683     5  0.0121 Preprocessor1_Model08
##  4  1778 roc_auc hand_till  0.683     5  0.0125 Preprocessor1_Model09
##  5   673 roc_auc hand_till  0.683     5  0.0114 Preprocessor1_Model04
##  6  1115 roc_auc hand_till  0.683     5  0.0125 Preprocessor1_Model06
##  7   452 roc_auc hand_till  0.682     5  0.0105 Preprocessor1_Model03
##  8   894 roc_auc hand_till  0.682     5  0.0120 Preprocessor1_Model05
##  9  1336 roc_auc hand_till  0.682     5  0.0125 Preprocessor1_Model07
## 10    10 roc_auc hand_till  0.672     5  0.0162 Preprocessor1_Model01
```

The best performing boosted trees model scored a mean roc_auc of 0.6835.

## Exercise 11

```
best3 = tune3 %>%
  select_best("roc_auc")

final_work3 = work3 %>%
  finalize_workflow(best3)

best_fit3 = final_work3 %>%
  fit(train)
saveRDS(best_fit3, "./data/best_fit3.rds")
```

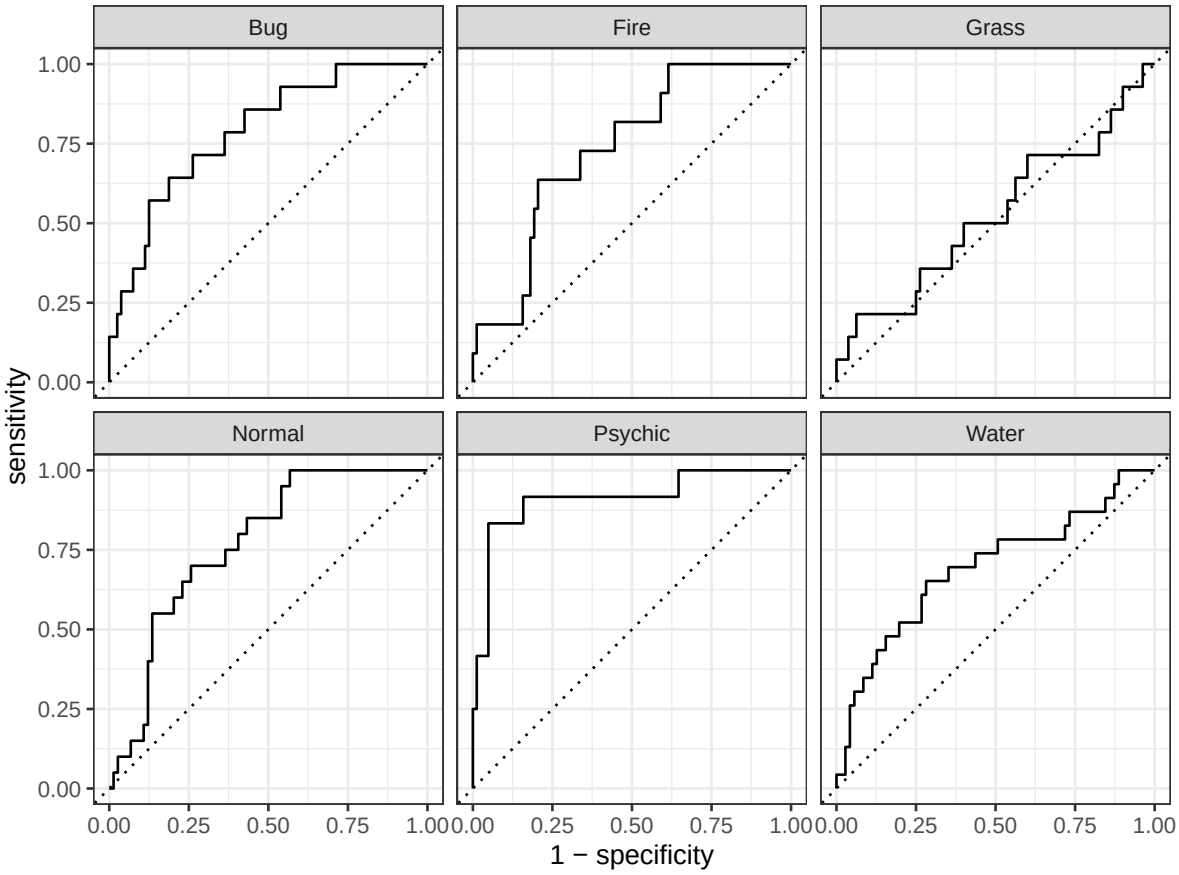
```
best_fit3 = readRDS("./data/best_fit3.rds")
```

We will fit the best-performing random forest model to the test set, as that model achieved higher roc_auc values relative to the best decision tree and boosted tree models.

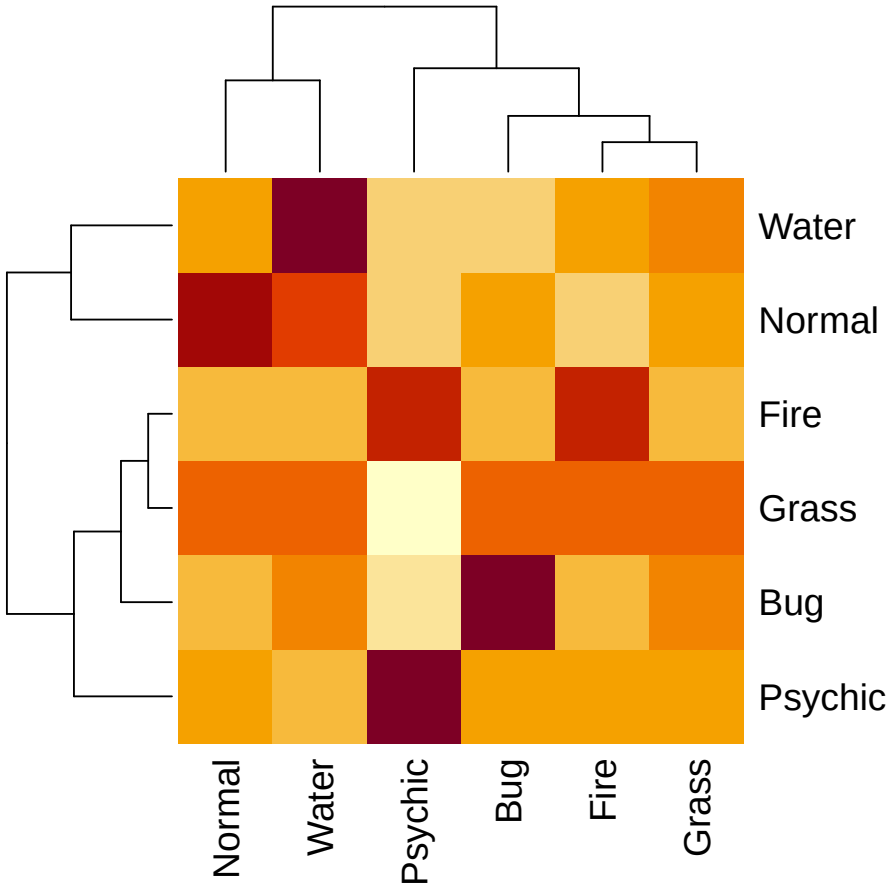
```
predict = augment(best_fit2, test)

test_roc_auc = roc_auc(data=predict,
                       truth=type_1,
                       estimate=c(.pred_Bug, .pred_Fire, .pred_Grass,
                                  .pred_Normal, .pred_Psychic, .pred_Water),
                       estimator="macro_weighted")
test_roc_auc$.estimate
```

```
## [1] 0.7305
```

```
test_curves = roc_curve(data=predict,
                        truth=type_1,
                        estimate=c(.pred_Bug, .pred_Fire, .pred_Grass,
                                   .pred_Normal, .pred_Psychic, .pred_Water))
autoplot(test_curves)
```




```
conf_mtx = conf_mat(data=predict, truth=type_1, estimate=.pred_class)
heatmap(conf_mtx$table)
```




In general, we have the most difficulty predicting the Grass class. Fire, Normal, and Water also look like they are difficult for the model to distinguish.