Pstat231HW6

Zihao Yang

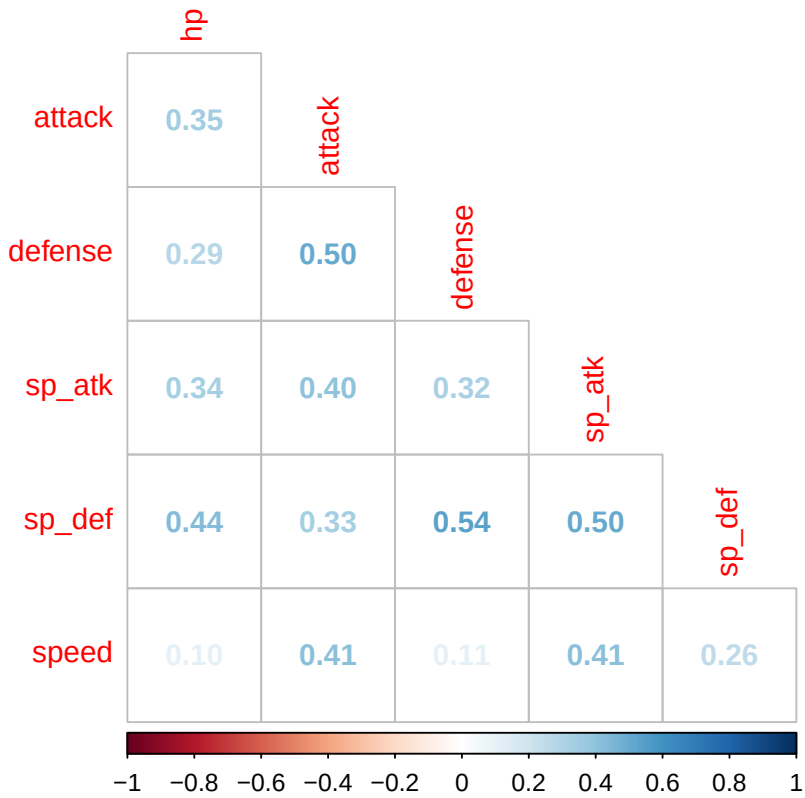
2022-05-24

Q1

```
# load the data
pokemon <- read.csv("Pokemon.csv")

#view(pokemon)
#clean names
pk <- pokemon %>% clean_names()

#Filter out the rarer Pokémon types and Convert type_1 and legendary to factors
pk2 <- pk %>%
  filter(type_1 %in%
           c("Bug","Fire","Grass","Normal","Water","Psychic"))
pk2$type_1 <- factor(pk2$type_1)
pk2$legendary <- factor(pk2$legendary)
pk2$generation <- factor(pk2$generation)

#Do a initial split
pk_split <- initial_split(pk2, prop = 0.80,  strata = "type_1")
pk_train <- training(pk_split)
pk_test <- testing(pk_split)

#Fold the training set using v-fold cv with v=5
pk_folds <- vfold_cv(pk_train,v=5,strata = "type_1")

#Set up the recipe
#Dummy-code legendary and generation;
#Center and scale all predictors.
pk_recipe <- recipe(type_1 ~ legendary + generation +
                      sp_atk + attack + speed +
                      defense + hp + sp_def,
                    data = pk_train) %>%
  step_dummy(c("legendary","generation")) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

Q2

```
pk_train %>%
  select(where(is.numeric)) %>%
  select(-x,-total) %>%
  cor() %>%
  corrplot(type = "lower",method = "number",diag = FALSE)
```

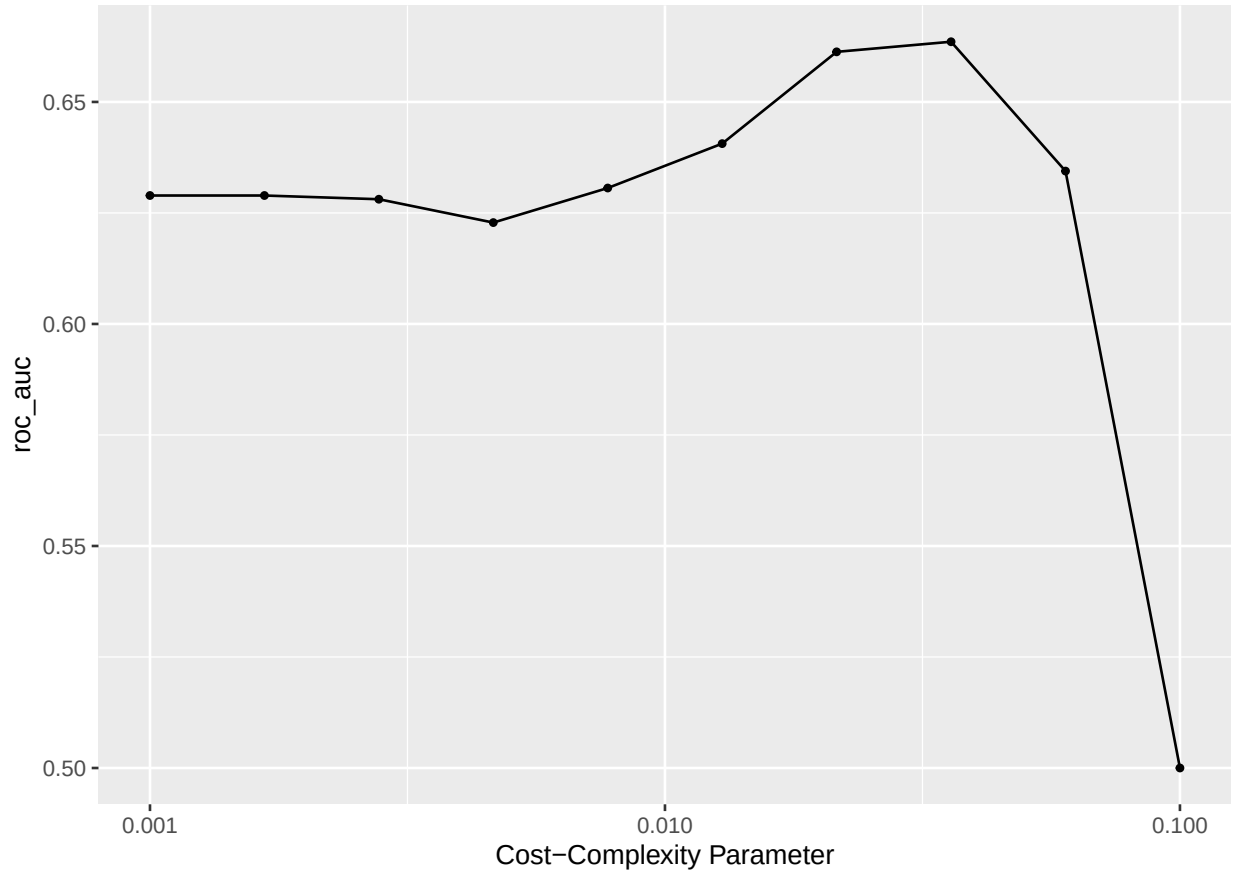


According to the correlation matrix, we can see that the sp_def is correlated with defense. The defense and attack are also correlated. The sp_atk and sp_def are also correlated. Speed are correlated with both attack and sp_atk. They are all make sense to me.

Q3

```
tree_spec <- decision_tree(cost_complexity = tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")

tree_wkflow <- workflow() %>%
  add_recipe(pk_recipe) %>%
  add_model(tree_spec)

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

tune_res <- tune_grid(tree_wkflow,
```

```
                    resamples = pk_folds,
                    grid = param_grid,
                    metrics = metric_set(roc_auc))
autoplot(tune_res)
```



The decision tree preforms better with smaller complexity penalties. It has a peak at 0.05, then drop significantly.

Q4

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

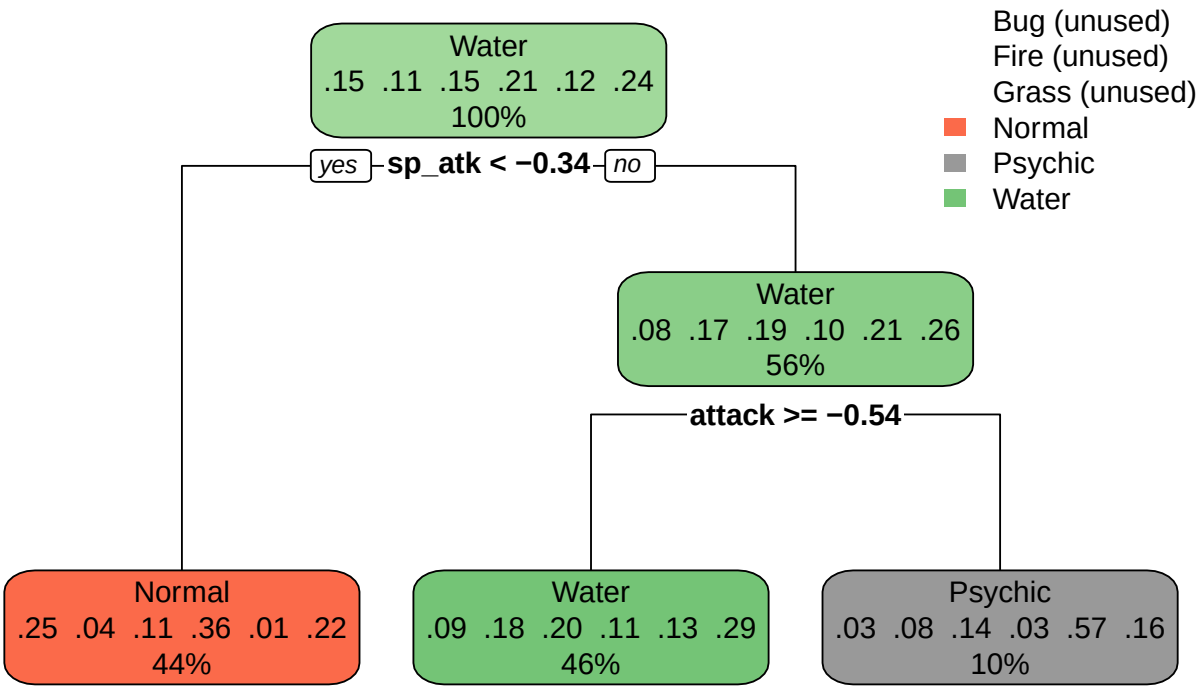
```
## # A tibble: 10 x 7
##    cost_complexity .metric .estimator  mean     n std_err .config
##              <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1         0.0359  roc_auc hand_till  0.664     5 0.00440 Preprocessor1_Model08
##  2         0.0215  roc_auc hand_till  0.661     5 0.00415 Preprocessor1_Model07
##  3         0.0129  roc_auc hand_till  0.641     5 0.0131  Preprocessor1_Model06
##  4         0.0599  roc_auc hand_till  0.634     5 0.00890 Preprocessor1_Model09
##  5         0.00774 roc_auc hand_till  0.631     5 0.0242  Preprocessor1_Model05
##  6         0.001   roc_auc hand_till  0.629     5 0.0202  Preprocessor1_Model01
##  7         0.00167 roc_auc hand_till  0.629     5 0.0202  Preprocessor1_Model02
##  8         0.00278 roc_auc hand_till  0.628     5 0.0199  Preprocessor1_Model03
```

```
##  9         0.00464 roc_auc hand_till  0.623     5 0.0214  Preprocessor1_Model04
## 10         0.1     roc_auc hand_till  0.5       5 0       Preprocessor1_Model10
```

The roc_auc of the best_performing pruned decision tree was 0.66355282.

Q5

```
best <- select_best(tune_res)

tree_final <- finalize_workflow(tree_wkflow, best)

tree_fit <- fit(tree_final,pk_train)

tree_fit %>%
  extract_fit_engine() %>%
  rpart.plot()
```



Q5

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

```
rf_wkflow <- workflow() %>%
  add_recipe(pk_recipe) %>%
  add_model(rf_model)
```

mtry: the number of predictors that will be randomly sampled when creating the tree models.

trees: the number of trees contained in the ensemble.

min_n: the minimum number of data points in a node that are required for the node to be split further.

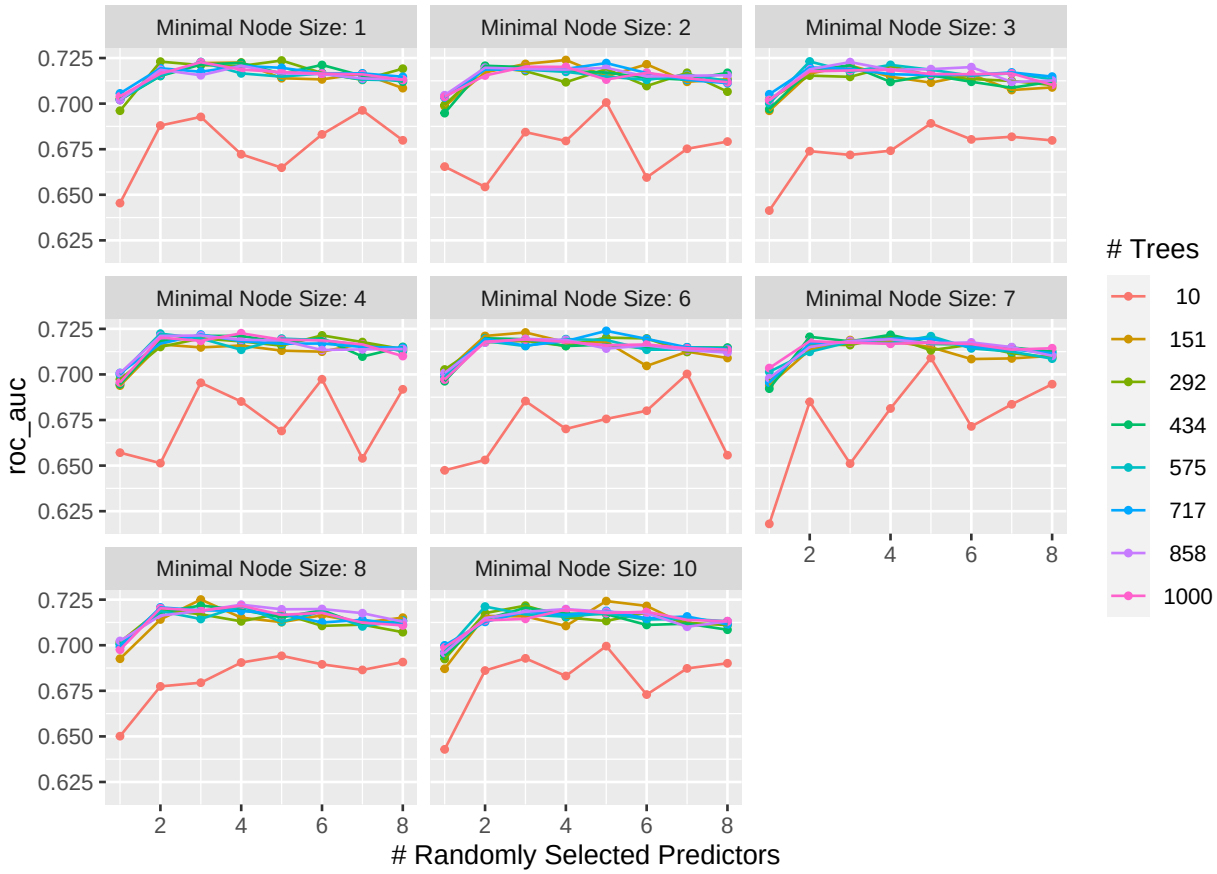
```
rf_grid <- grid_regular(
  mtry(range = c(1, 8)),
  trees(range = c(10,1000)),
  min_n(range = c(1, 10)),
  levels = 8)
```

mtry can't be greater than 8 because it represents the number of predictors and we only have 8 predictors. If mtry = 8, then the model uses all 8 predictors.

Q6

```
rf_tune <- tune_grid(
  rf_wkflow,
  resamples = pk_folds,
  grid = rf_grid,
  metrics = metric_set(roc_auc)
)

autoplot(rf_tune)
```

```
autoplot(rf_tune)
```



Minimal node size seems to have little effect on the accuracy. In general, more trees yield to high and stable accuracy, especially wher it's larger than 10. And as the number of predictors increasing, the accuracy increase on a large scale.

Q7

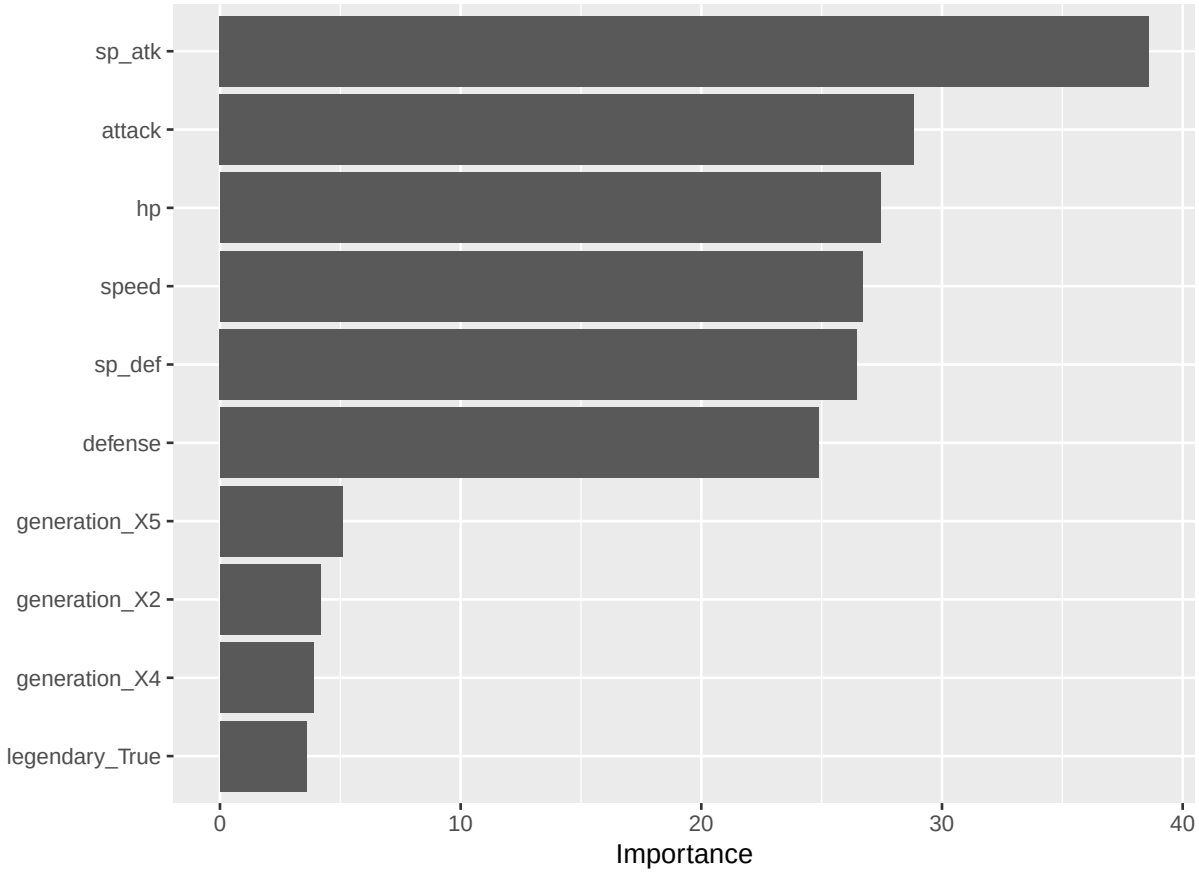
```
collect_metrics(rf_tune) %>% arrange(desc(mean))
```

```
## # 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   151     8 roc_auc hand_till  0.725     5  0.0158 Preprocessor1_Model~
##  2     5   151    10 roc_auc hand_till  0.724     5  0.0162 Preprocessor1_Model~
##  3     4   151     2 roc_auc hand_till  0.724     5  0.0170 Preprocessor1_Model~
##  4     5   717     6 roc_auc hand_till  0.724     5  0.0161 Preprocessor1_Model~
##  5     5   292     1 roc_auc hand_till  0.724     5  0.0159 Preprocessor1_Model~
##  6     2   575     3 roc_auc hand_till  0.723     5  0.0162 Preprocessor1_Model~
##  7     2   292     1 roc_auc hand_till  0.723     5  0.0208 Preprocessor1_Model~
##  8     3   151     6 roc_auc hand_till  0.723     5  0.0172 Preprocessor1_Model~
##  9     3   575     1 roc_auc hand_till  0.723     5  0.0176 Preprocessor1_Model~
## 10     3   858     3 roc_auc hand_till  0.723     5  0.0168 Preprocessor1_Model~
## # ... with 502 more rows
```

The best model's roc_auc is 0.72505186, with mtry=3, trees=151, and min_n=8.

Q8

```
rf_final <- finalize_workflow(rf_wkflow,select_best(rf_tune,"roc_auc"))

rf_fit <- fit(rf_final,pk_train)

rf_fit %>%
  extract_fit_engine() %>%
  vip()
```

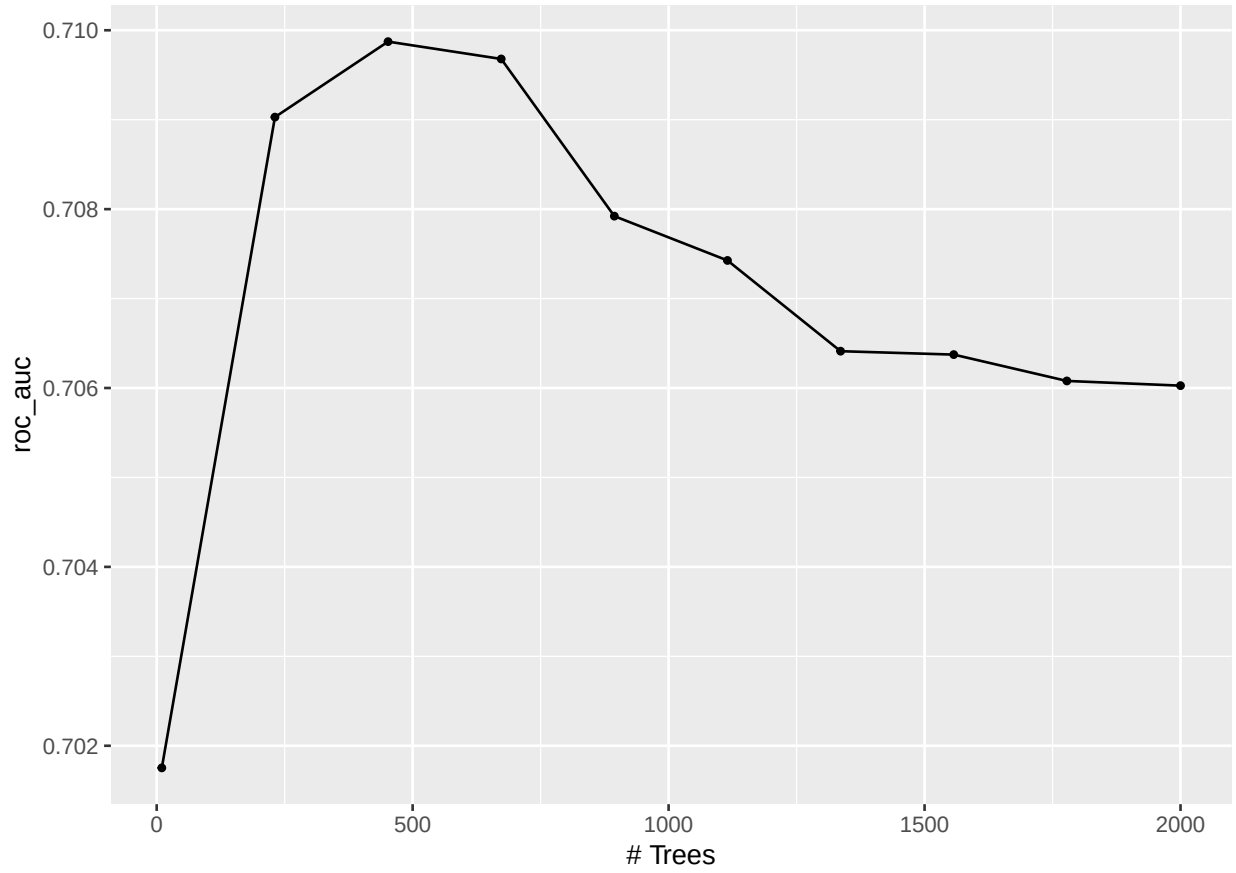


The most useful variable is sp_atk, and the least useful varible is generation and legendary. Besides, the attack, hp, sp_def, speed, and defense are also quite useful.

Q9

```
bt_spec <- boost_tree(trees = tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")

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

bt_wkflow <- workflow() %>%
  add_model(bt_spec) %>%
```

```
  add_recipe(pk_recipe)

bt_tune_res <- tune_grid(
  bt_wkflow,
  resamples = pk_folds,
  grid = bt_grid,
  metrics = metric_set(roc_auc)
)

autoplot(bt_tune_res)
```



The roc_auc increases when number of trees is increasing, and reaches the peak at around 450 trees. Then the roc_auc keeps decreasing as the trees increasing.

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

```
## # A tibble: 10 x 7
##    trees .metric .estimator  mean     n std_err .config
##    <int> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
##  1   452 roc_auc hand_till  0.710     5  0.0138 Preprocessor1_Model03
##  2   673 roc_auc hand_till  0.710     5  0.0134 Preprocessor1_Model04
##  3   231 roc_auc hand_till  0.709     5  0.0146 Preprocessor1_Model02
##  4   894 roc_auc hand_till  0.708     5  0.0124 Preprocessor1_Model05
##  5  1115 roc_auc hand_till  0.707     5  0.0126 Preprocessor1_Model06
##  6  1336 roc_auc hand_till  0.706     5  0.0125 Preprocessor1_Model07
```

```
##  7  1557 roc_auc hand_till  0.706     5  0.0124 Preprocessor1_Model08
##  8  1778 roc_auc hand_till  0.706     5  0.0124 Preprocessor1_Model09
##  9  2000 roc_auc hand_till  0.706     5  0.0124 Preprocessor1_Model10
## 10    10 roc_auc hand_till  0.702     5  0.0210 Preprocessor1_Model01
```

The best_performing boosted tree model's roc_auc is 0.70987274 with 452 trees.

Q10

```
bt_final <- finalize_workflow(bt_wkflow,select_best(bt_tune_res,"roc_auc"))
bt_fit <- fit(bt_final,pk_train)

final_class_model <- augment(tree_fit, new_data = pk_train)
final_random_forest <- augment(rf_fit, new_data = pk_train)
final_boosted_tree <- augment(bt_fit, new_data = pk_train)

bind_rows(
  roc_auc(final_class_model, truth = type_1, .pred_Bug,
          .pred_Fire, .pred_Grass, .pred_Normal, .pred_Water, .pred_Psychic),
  roc_auc(final_random_forest, truth = type_1, .pred_Bug,
          .pred_Fire, .pred_Grass, .pred_Normal, .pred_Water, .pred_Psychic),
  roc_auc(final_boosted_tree, truth = type_1, .pred_Bug,
          .pred_Fire, .pred_Grass, .pred_Normal, .pred_Water, .pred_Psychic))
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.589
## 2 roc_auc hand_till      0.780
## 3 roc_auc hand_till      0.800
```

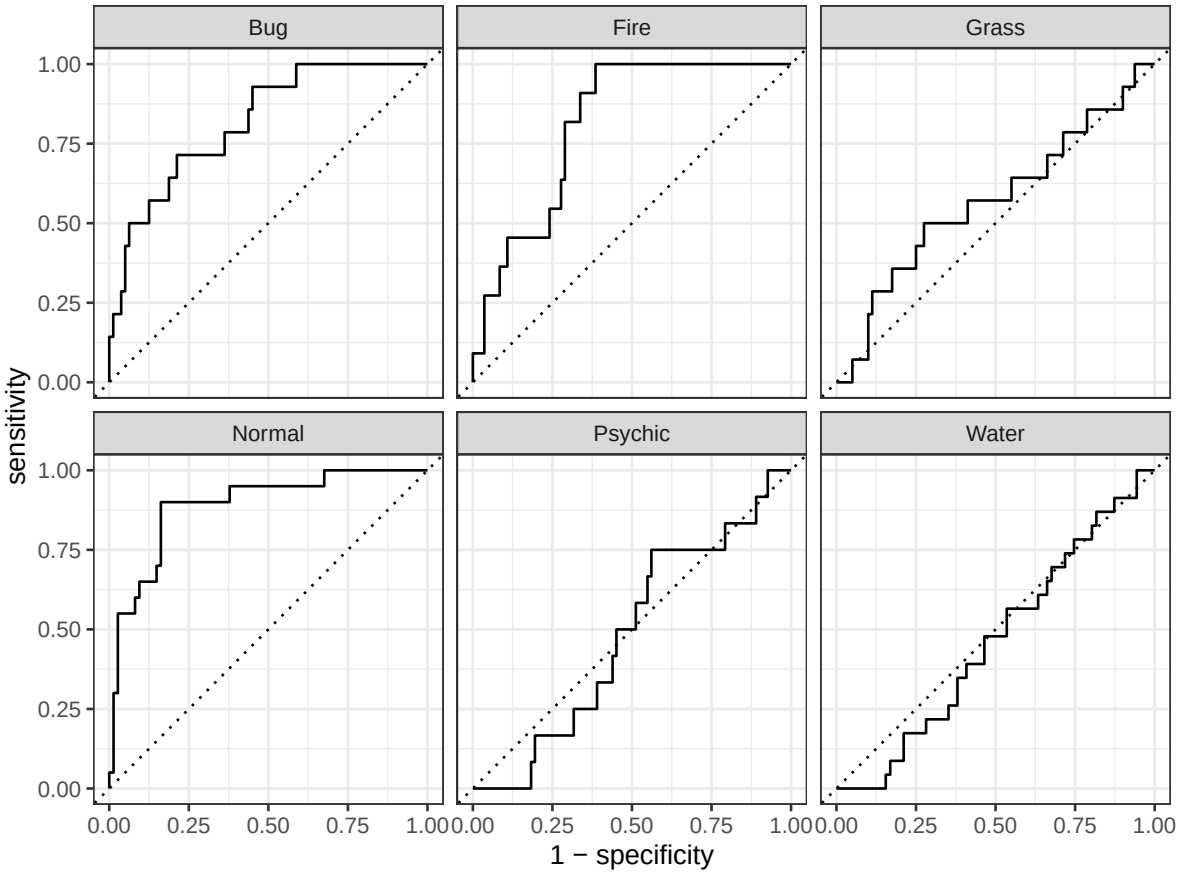
From the output, we can see that the best model is boosted tree model with roc_auc of 0.79992047.

```
final_boosted_tree_test <- augment(bt_fit, new_data = pk_test)
roc_auc(final_boosted_tree_test, truth = type_1, .pred_Bug,
        .pred_Fire, .pred_Grass, .pred_Normal, .pred_Water, .pred_Psychic)
```

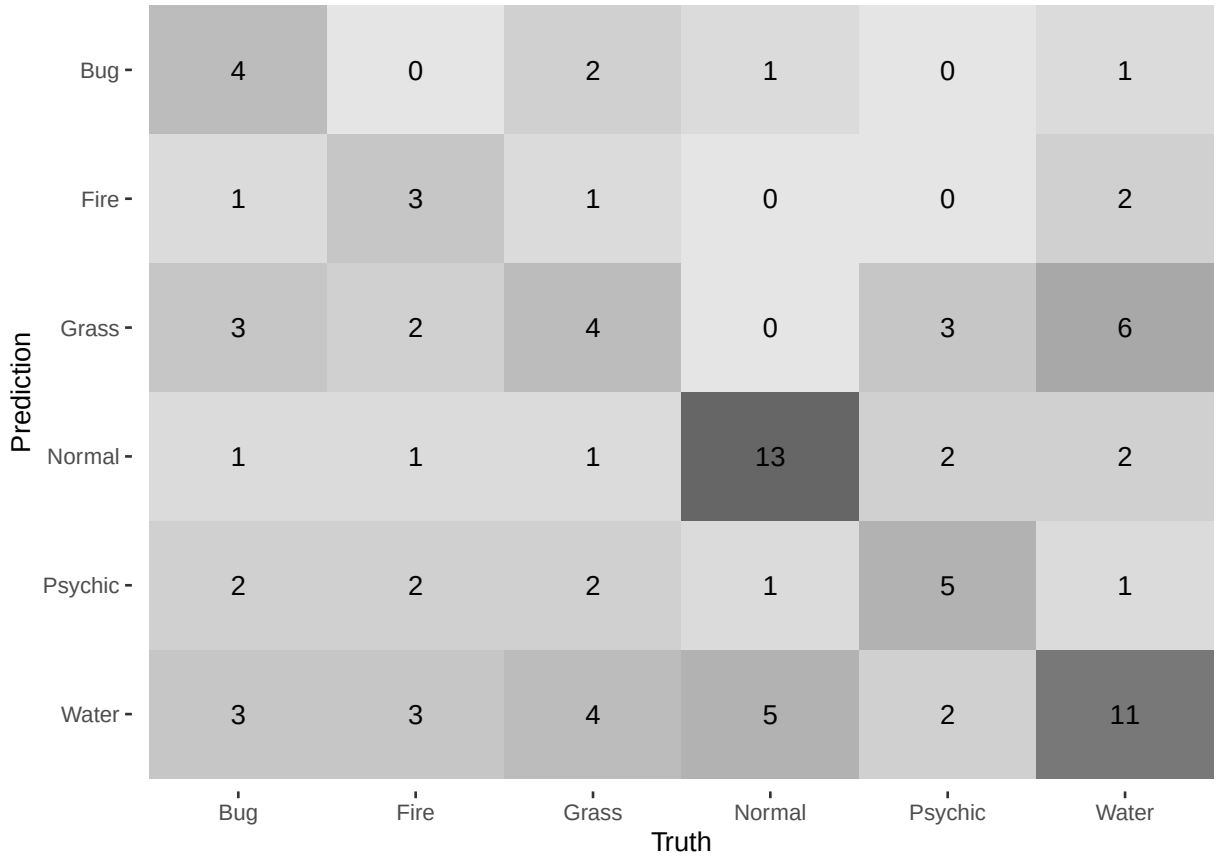
```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.670
```

The roc_auc on the test data set is 0.66969006.

```
autoplot(roc_curve(final_boosted_tree_test, truth = type_1, .pred_Bug,
        .pred_Fire, .pred_Grass, .pred_Normal, .pred_Water, .pred_Psychic))
```




```
conf_mat(final_boosted_tree_test, truth = type_1, estimate = .pred_class) %>%
  autoplot(type = "heatmap")
```



The model is most accurate at predicting Normal, Fire, and Bug, and worst at predicting Water, Psychic, and Grass.

Q11

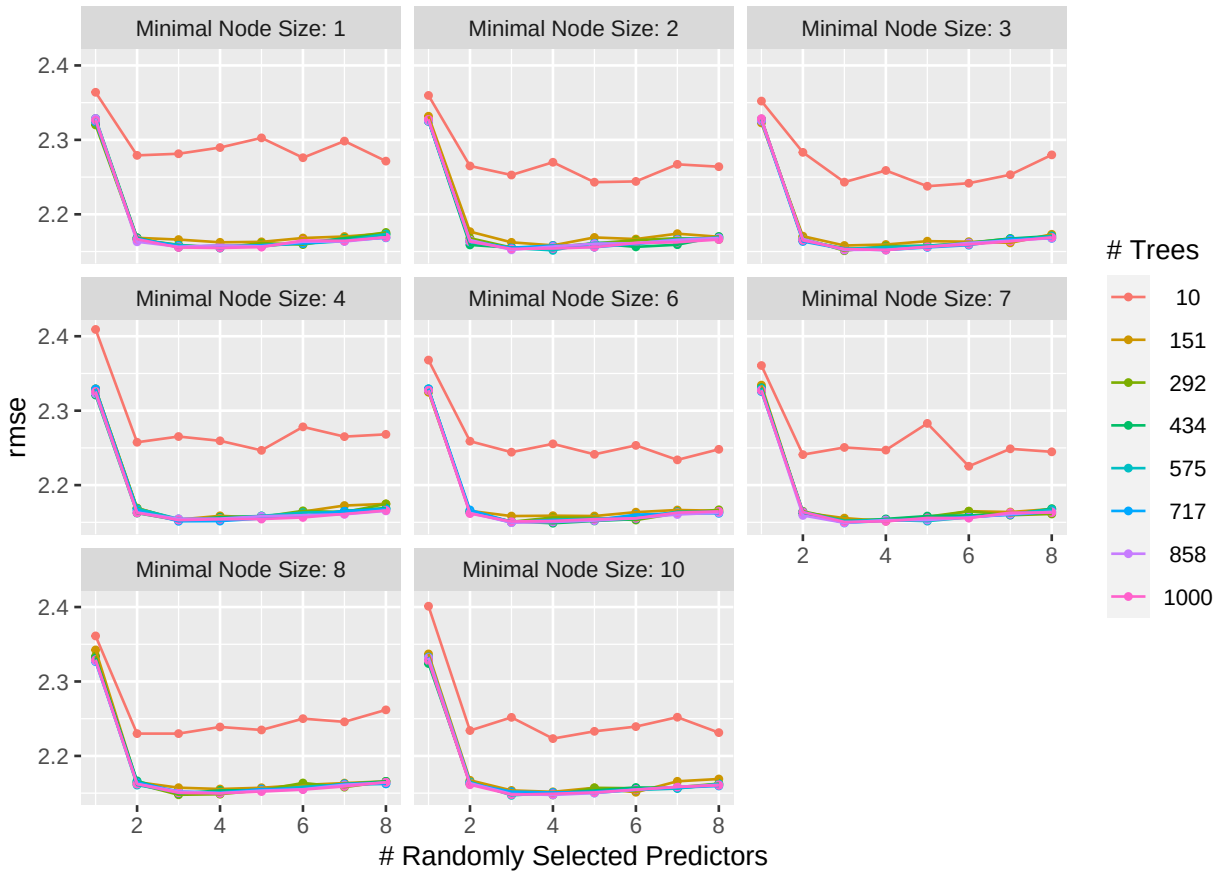
```
abalone <- read.csv("abalone.csv")
abalone["age"] <- abalone["rings"]+1.5
abalone_split <- initial_split(abalone,prop=0.80,strata = age)
abalone_train <- training(abalone_split)
abalone_test <- testing(abalone_split)
abalone_folds <- vfold_cv(abalone_train, v = 5, strata = age)
abtrain_wo_rings <- abalone_train %>% select(-rings)
abalone_recipe <- recipe(age ~ ., data = abtrain_wo_rings) %>%
  step_dummy(all_nominal_predictors()) %>%
  step_interact(terms= ~ starts_with("type"):shucked_weight+
                  longest_shell:diameter+
                  shucked_weight:shell_weight) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

```
abalone_rf <- rand_forest(mtry = tune(), trees = tune(), min_n = tune()) %>%
  set_engine("ranger", importance = "impurity") %>%
  set_mode("regression")
```

```
abalone_wkflow <- workflow() %>%
  add_recipe(abalone_recipe) %>%
  add_model(abalone_rf)

abalone_grid <- grid_regular(
  mtry(range = c(1,8)),
  trees(range = c(10,1000)),
  min_n(range = c(1,10)),
  levels = 8
)
```

```
abalone_tune <- tune_grid(
  abalone_wkflow,
  resamples = abalone_folds,
  grid = abalone_grid,
  metrics = metric_set(rmse)
)
```

```
autoplot(abalone_tune)
```




```
abalone_final <- finalize_workflow(abalone_wkflow,select_best(abalone_tune))
abalone_fit <- fit(abalone_final, abalone_train)
```

```r
augment(abalone_fit, new_data = abalone_test) %>%
  rmse(truth = age, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        2.13
```

The model's RMSE on the testing set is 2.1321725.