HW 6

PSTAT 131/231 Arthur Starodynov

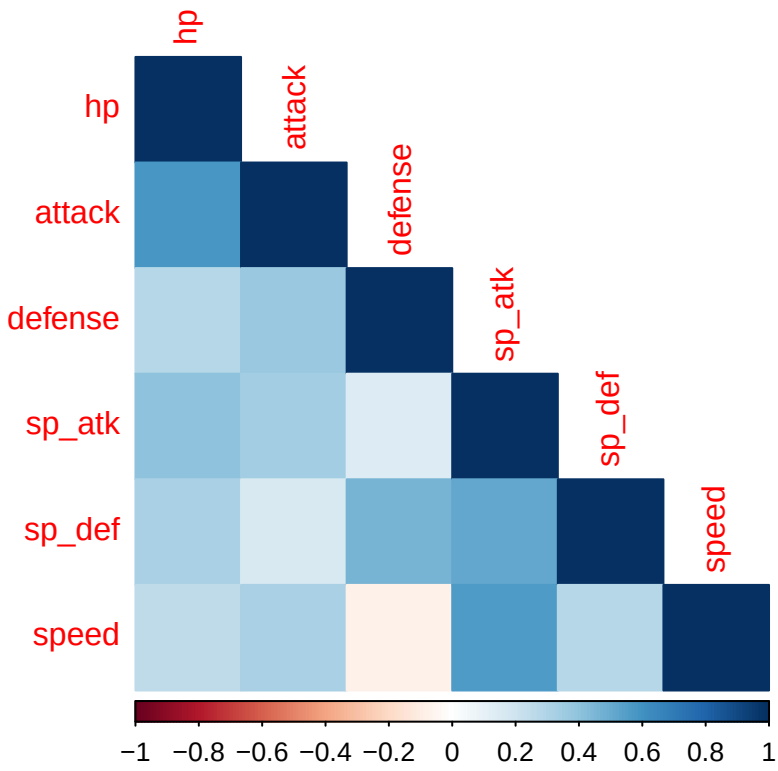
Contents

Exercise 1:

```
cleanpokemon = pokemon.data %>%
  clean_names()
cleanpokemon <- cleanpokemon %>%
  filter(
    !str_detect(type_1, "Bug") &
    !str_detect(type_1, "Fire") &
    !str_detect(type_1, "Grass") &
    !str_detect(type_1, "Normal") &
    !str_detect(type_1, "Water") &
    !str_detect(type_1, "Psychic")
    )
cleanpokemon$type_1 = as.factor(cleanpokemon$type_1)
cleanpokemon$legendary = as.factor(cleanpokemon$legendary)
cleanpokemon$generation = as.factor(cleanpokemon$generation)
split = cleanpokemon %>%
  initial_split(prop=0.7, strata="type_1")
train = training(split)
test = testing(split)
folds = vfold_cv(train, v=5, strata="type_1")
simple_recipe = 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:

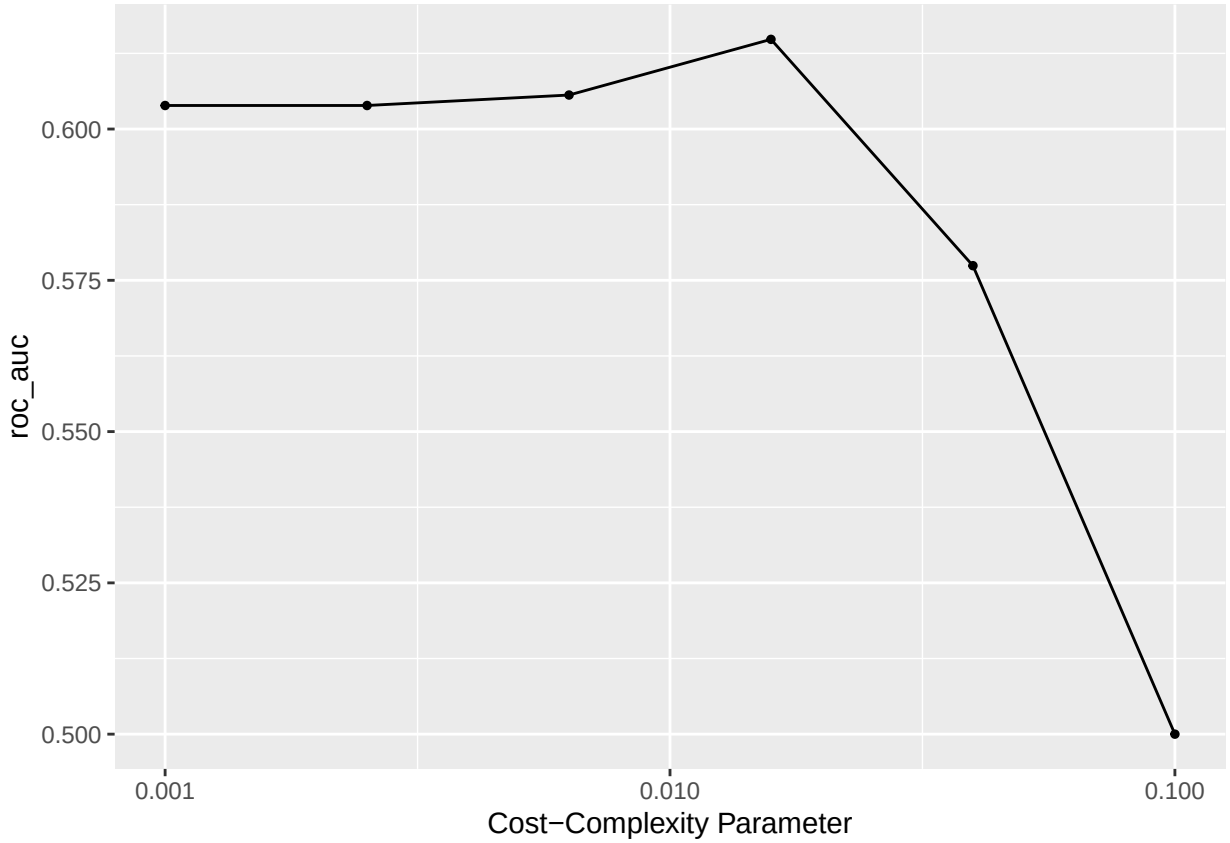
```
corrplot(cor(cleanpokemon[ , names(cleanpokemon) %in% c("sp_atk", "attack", "speed",
                                    "defense", "hp", "sp_def")]),
         method="color", type="lower")
```



What can be seen is that sp_def is correlated with defense and sp_atk is correlated with atack.

Exercise 3:

```
mod = decision_tree(cost_complexity=tune()) %>%
  set_engine("rpart") %>%
  set_mode("classification")
work = workflow() %>%
  add_model(mod) %>%
  add_recipe(simple_recipe)
grid = grid_regular(cost_complexity(range=c(-3, -1)), levels=6)
tune = work %>%
  tune_grid(resamples=folds, grid=grid, metrics=metric_set(roc_auc))
autoplot(tune, metric="roc_auc")
```



Smaller complexity cost means that the decision tree would perform better.

Exercise 4:

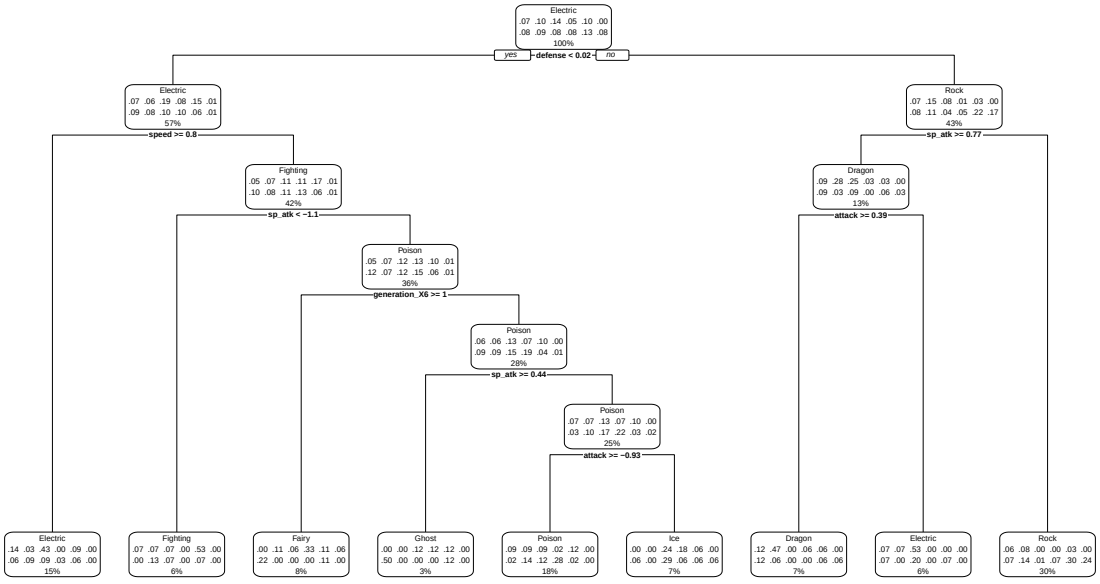
```
tune %>% collect_metrics() %>% arrange(desc(mean))
```

```
## # A tibble: 6 x 7
##   cost_complexity .metric .estimator  mean     n std_err .config
##             <dbl> <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1         0.0158  roc_auc hand_till  0.615     5  0.0186 Preprocessor1_Model4
## 2         0.00631 roc_auc hand_till  0.606     5  0.0193 Preprocessor1_Model3
## 3         0.001   roc_auc hand_till  0.604     5  0.0169 Preprocessor1_Model1
## 4         0.00251 roc_auc hand_till  0.604     5  0.0169 Preprocessor1_Model2
## 5         0.0398  roc_auc hand_till  0.577     5  0.0162 Preprocessor1_Model5
## 6         0.1     roc_auc hand_till  0.5       5  0      Preprocessor1_Model6
```

Best performing tree had 5 folds and had a mean of .6353

Exercise 5:

```
best = tune %>%
  select_best("roc_auc")
final_work = work %>%
  finalize_workflow(best)
best_fit = final_work %>%
  fit(train)
rpart.plot( extract_fit_parsnip(best_fit)$fit )
```



Exercise 6:

```
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(simple_recipe)
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))
```

A random forest model will make independent tress and uses its predicitons. MTRY is limited for the eight predictors given and we have to make sure that they are either non negative and not 0. The tree paramter is total number of trees. With more trees then there are more randomly selected predicters to get a better yield.

```
autoplot(tune2, metric="roc_auc")
```

Exercise 7:

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

Exercise 8:

```
bestft = tune2 %>%
  select_best("roc_auc")
final_forest = work2 %>%
  finalize_workflow(bestft)
best_fit = final_forest %>%
  fit(train)
best_fit %>%
  extract_fit_parsnip() %>%
  vip()
```

Exercise 9:

```
od3 = boost_tree(trees=tune()) %>%
  set_engine("xgboost") %>%
  set_mode("classification")
work3 = workflow() %>%
  add_model(od3) %>%
  add_recipe(simple_recipe)
grid3 = grid_regular(trees(range=c(10, 2000)), levels=10)
tune3 = work3 %>%
  tune_grid(resamples=folds, grid=grid3, metrics=metric_set(roc_auc))
autoplot(tune3, metric="roc_auc")
```

More trees equals more yield.

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

Exercise 10:

```
best3 = tune3 %>%
  select_best("roc_auc")
final_work3 = work3 %>%
  finalize_workflow(best3)
best_fit3 = final_work3 %>%
  fit(train)
predict = augment(best_fit, 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$.estimate
test_curves = roc_curve(data=predict,
                        truth=type_1,
                        estimate=c(.pred_Bug, .pred_Fire, .pred_Grass,
                                   .pred_Normal, .pred_Psychic, .pred_Water))
autoplot(test_curves)
conf_mtx = conf_mat(data=predict, truth=type_1, estimate=.pred_class)
heatmap(conf_mtx$table)
```