Homework Assignment 5

Vasiliy Ostapenko (774 970 8)

May 10, 2022

Data

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

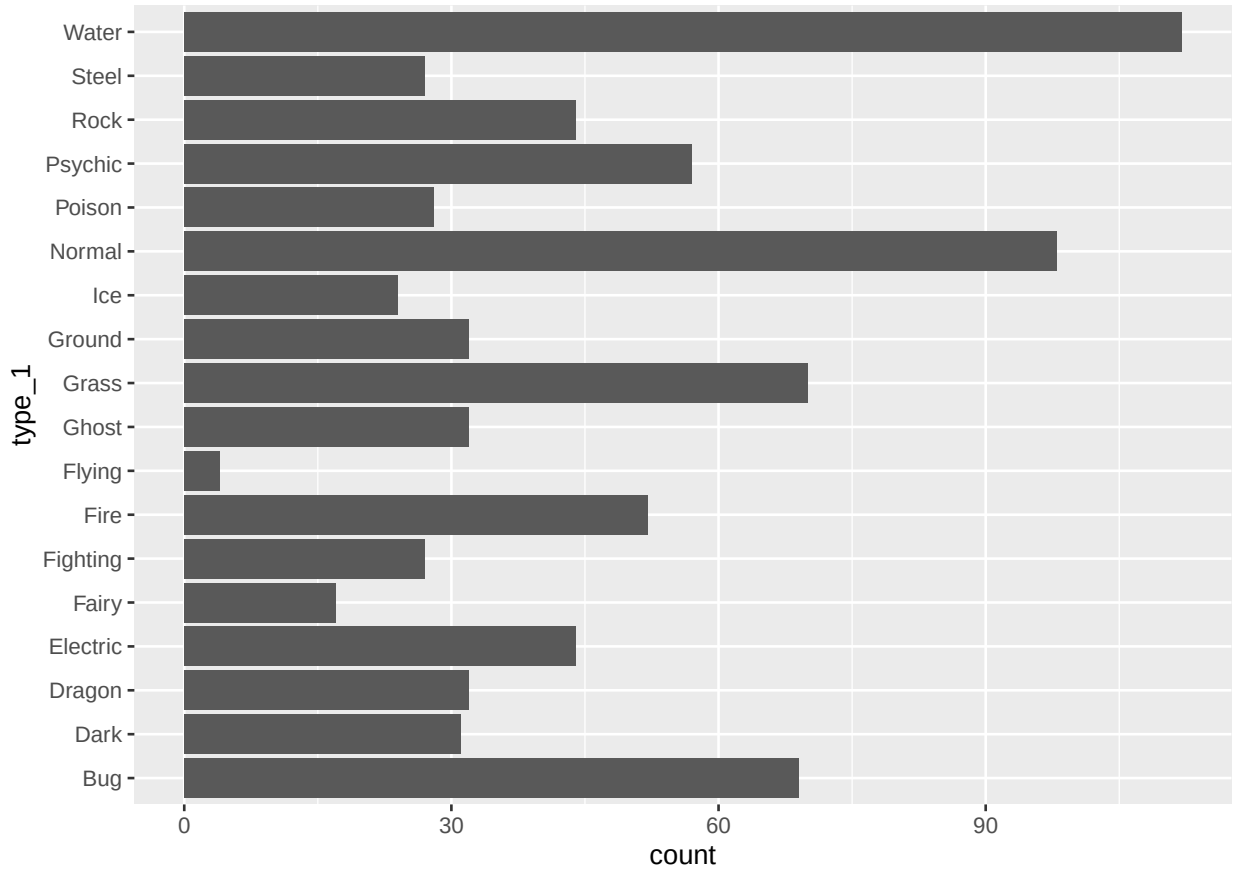
Exercise 1

```
df = df %>% clean_names()
```

Column names are modified to only contain underscores and alphanumeric characters. Spaces and dots, for instance, become underscores. Camel case is used to format names which contain multiple words. This function is useful becomes columns become easier to access by name.

Exercise 2

```
g1 = ggplot(data=df, aes(x=type_1)) +
  geom_bar(stat="count") +
  coord_flip()
g1
```



There are 18 unique classes of type_1. Flying type, for instance, has very few pokemon.

```
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)
```

Exercise 3

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

p_train = training(p_split)
p_test = testing(p_split)
```

```
p_folds = vfold_cv(p_train, v=5, strata="type_1")
```

Stratifying the folds will ensure that we will have a representative number of each class on which to train the model. Thus model performance for training and validation should be similar.

Exercise 4

```
p_rec = recipe(type_1 ~ legendary + generation + sp_atk + attack +
               speed + defense + hp + sp_def, data=p_train) %>%
```

```
  step_dummy(c("legendary", "generation")) %>%
  step_normalize(all_predictors())
```

Exercise 5

```
p_mod = multinom_reg(penalty=tune(), mixture=tune()) %>%
  set_engine("glmnet") %>%
  set_mode("classification")

p_work = workflow() %>%
  add_model(p_mod) %>%
  add_recipe(p_rec)
```

```
p_grid = grid_regular(penalty(range=c(-5, 5)),
                      mixture(range=c(0, 1)), levels=10)
```

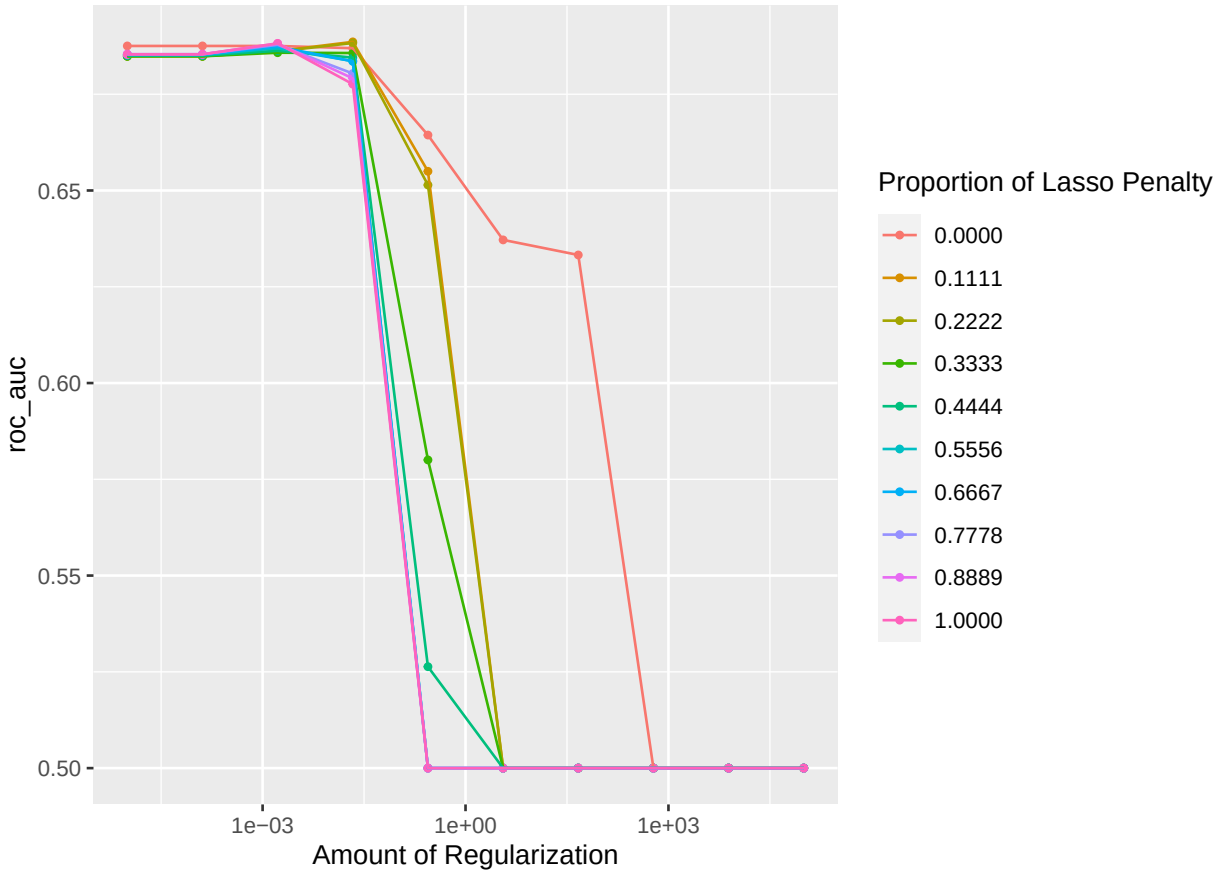
We will be fitting a total of 500 models. 10 levels for penalty times 10 levels for mixture times 5 CV folds.

Exercise 6

```
p_tune = p_work %>%
  tune_grid(resamples=p_folds, grid=p_grid, metrics=metric_set(roc_auc))

save(p_tune, p_work, file="./data/tune.rda")
```

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



Penalty should be very negative and mixture should be close to zero to get as large value for ROC AUC as possible.

Exercise 7

```
p_best = p_tune %>%
  select_best("roc_auc")

p_work_final = p_work %>%
  finalize_workflow(p_best)
```

```
p_fit = p_work_final %>%
  fit(p_train)
```

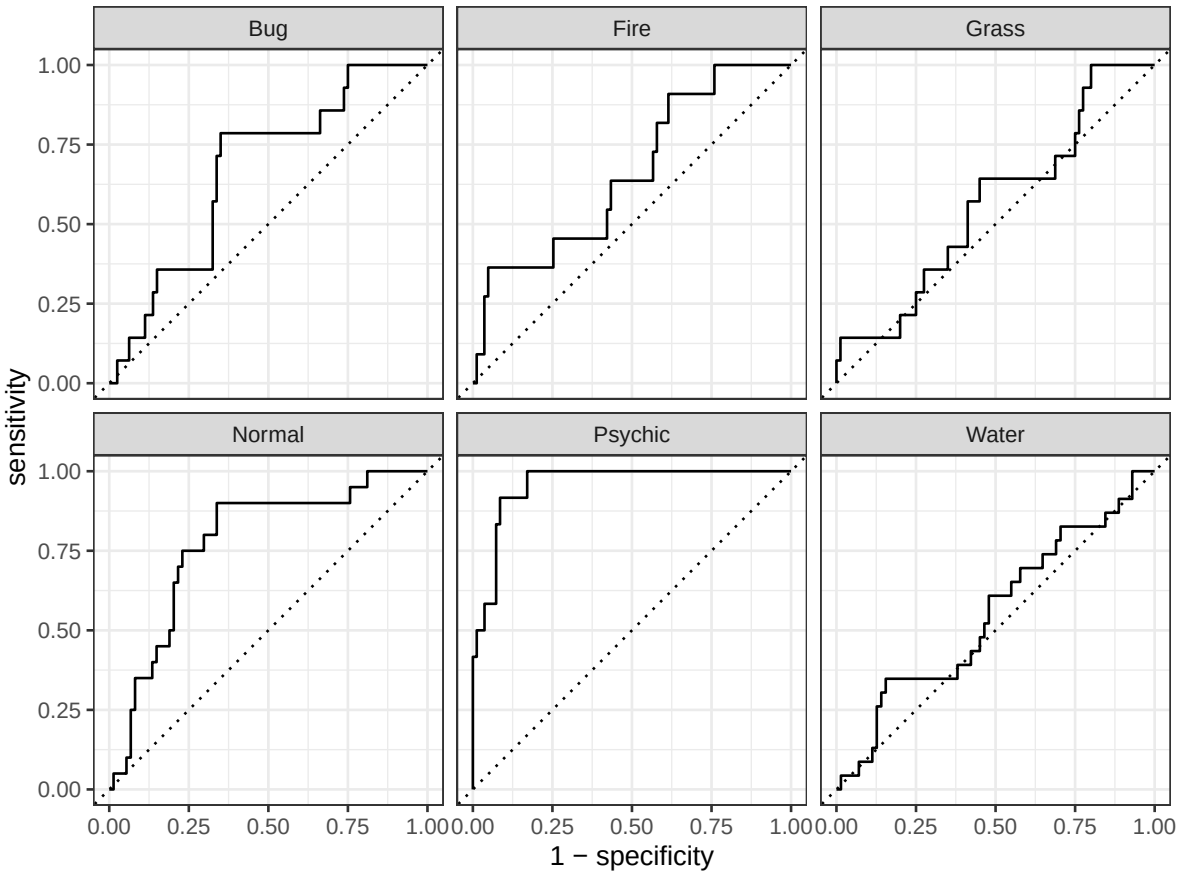
```
p_predict = augment(p_fit, p_test)
```

Exercise 8

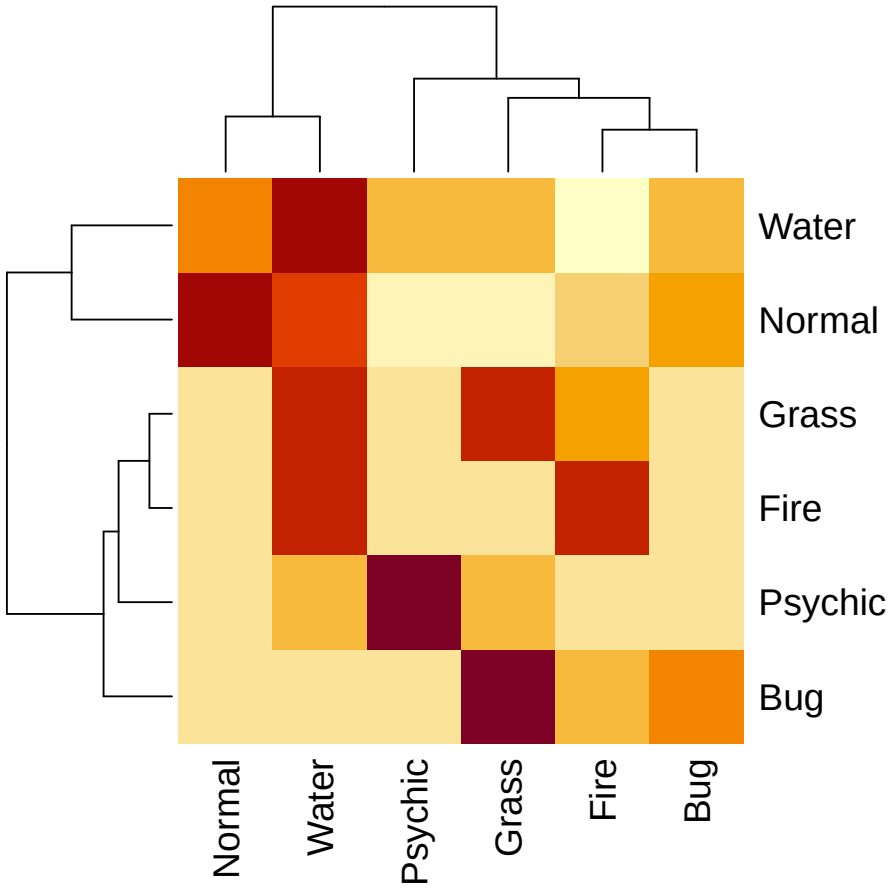
```
test_roc_auc = roc_auc(data=p_predict,
                       truth=type_1,
                       estimate=c(.pred_Bug, .pred_Fire, .pred_Grass,
                                  .pred_Normal, .pred_Psychic, .pred_Water),
                       estimator="macro_weighted")
test_roc_auc$.estimate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

```
## [1] 0.6823
```

```
test_curves = roc_curve(data=p_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=p_predict, truth=type_1, estimate=.pred_class)
heatmap(conf_mtx$table)
```



Given a test-set ROC AUC of about 68%, the model's performance is very poor. Moreover, looking at the individual ROC AUC curves, as well as the confusion matrix, we can state that we predict only psychic, normal and bug classes well. The model evidently has problems distinguishing the other three types, namely water, grass and fire. Some common mistakes include confusing: grass and bug; water and essentially every other class. The reasons for this poor performance, in general, would have to be that we do not have enough useful features per sample, and that we need more samples.