Homework 5

PSTAT 131/231

Contents

Elastic Net Tuning

The goal of this assignment is to build a statistical learning model that can predict the **primary type** of a Pokémon based on its generation, legendary status, and six battle statistics.

Read in the file and familiarize yourself with the variables using `pokemon_codebook.txt`.

Exercise 1

Install and load the `janitor` package. Use its `clean_names()` function on the Pokémon data, and save the results to work with for the rest of the assignment. What happened to the data? Why do you think `clean_names()` is useful?

```
library(tidymodels)
library(tidyverse)
library(ISLR) # For the Smarket data set
library(ISLR2) # For the Bikeshare data set
library(discrim)
library(poissonreg)
library(corrr)
library(klaR) # for naive bayes
library(forcats)
library(corrplot)
library(pROC)
library(recipes)
library(rsample)
library(parsnip)
library(workflows)
library(janitor)
library(glmnet)
tidymodels_prefer()
```

```
pk_data <- read.csv(file = "pokemon.csv")

pk_data <- clean_names(pk_data)
```

the cols in the dataset all changed to lower letter and accented characters are transliterated to ASCII.

Exercise 2

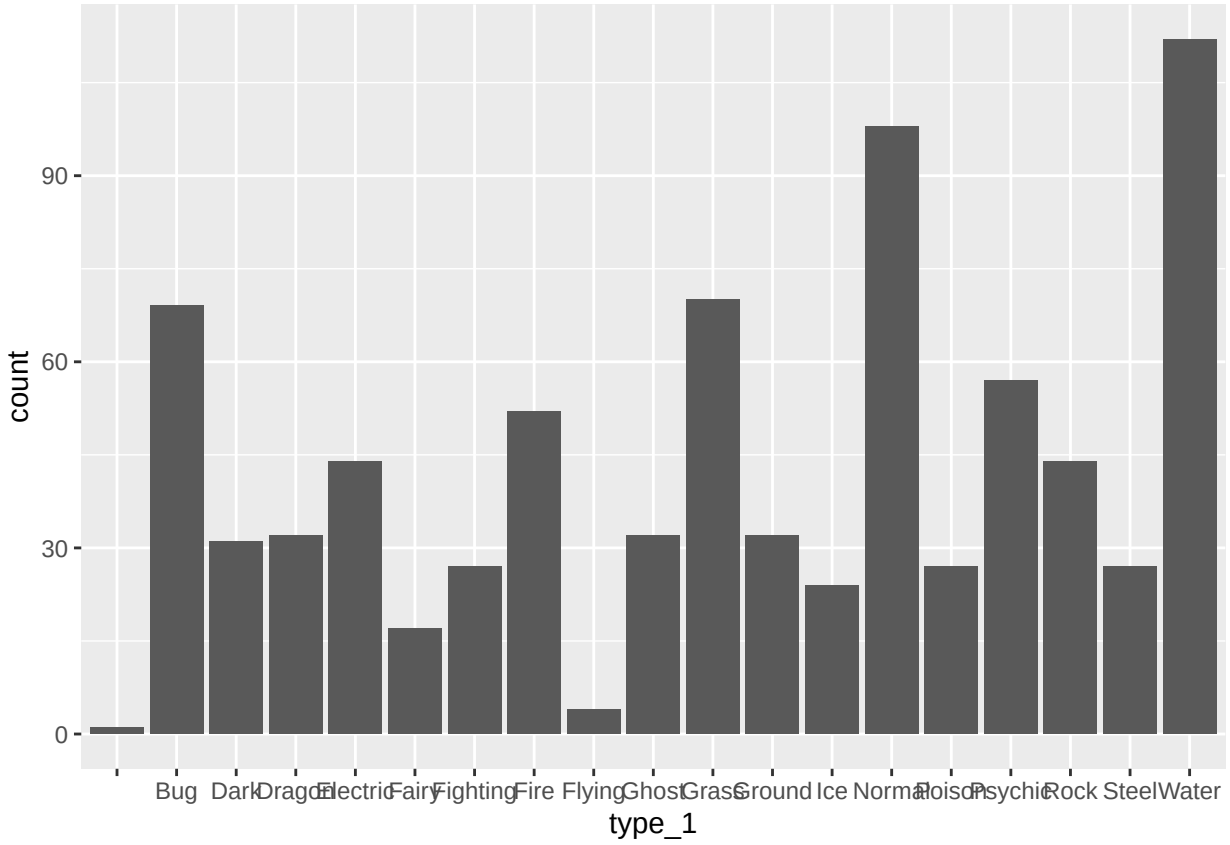
Using the entire data set, create a bar chart of the outcome variable, `type_1`.

How many classes of the outcome are there? Are there any Pokémon types with very few Pokémon? If so, which ones?

For this assignment, we'll handle the rarer classes by simply filtering them out. Filter the entire data set to contain only Pokémon whose `type_1` is Bug, Fire, Grass, Normal, Water, or Psychic.

After filtering, convert `type_1` and `legendary` to factors.

```
pk_data %>%
  ggplot(aes( x = type_1), fill = type_1) +
  geom_bar()
```




```
#filter
ft_data <- filter(pk_data, type_1 == "Bug" | type_1 == "Fire" | type_1 == "Grass" | type_1 == "Normal"
ft_data$type_1 = as.factor(ft_data$type_1)
ft_data$legendary = as.factor(ft_data$legendary)
```

Exercise 3

Perform an initial split of the data. Stratify by the outcome variable. You can choose a proportion to use. Verify that your training and test sets have the desired number of observations.

Next, use *v*-fold cross-validation on the training set. Use 5 folds. Stratify the folds by `type_1` as well. *Hint: Look for a* ***`strata`*** *argument.* Why might stratifying the folds be useful?

```
set.seed(1000)

pk_split <- initial_split(ft_data, prop = 0.75, strata = type_1)
pk_train <- training(pk_split)
pk_test <- testing(pk_split)

dim(pk_train)
```

```
## [1] 341  13
```

```
dim(pk_test)
```

```
## [1] 117  13
```

```
pk_fold <- vfold_cv(pk_train, v = 5)
pk_fold
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits           id
##   <list>           <chr>
## 1 <split [272/69]> Fold1
## 2 <split [273/68]> Fold2
## 3 <split [273/68]> Fold3
## 4 <split [273/68]> Fold4
## 5 <split [273/68]> Fold5
```

Because there's two differ types we specific the first type of the pokemon and want to stratify sampling by it so the fold could have best fit.

Exercise 4

Set up a recipe to predict `type_1` with `legendary`, `generation`, `sp_atk`, `attack`, `speed`, `defense`, `hp`, and `sp_def`.

- Dummy-code `legendary` and `generation`;
- Center and scale all predictors.

```
recipe_pk <- recipe(type_1 ~ legendary + generation + sp_atk + attack + speed + defense + hp + sp_def, d
  step_dummy(legendary) %>%
  step_dummy(generation) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

Exercise 5

We'll be fitting and tuning an elastic net, tuning `penalty` and `mixture` (use `multinom_reg` with the `glmnet` engine).

Set up this model and workflow. Create a regular grid for `penalty` and `mixture` with 10 levels each; `mixture` should range from 0 to 1. For this assignment, we'll let `penalty` range from -5 to 5 (it's log-scaled).

How many total models will you be fitting when you fit these models to your folded data?

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

pk_wkflow <- workflow() %>%
   add_recipe(recipe_pk) %>%
  add_model(pk_spec)

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

will have 500 models

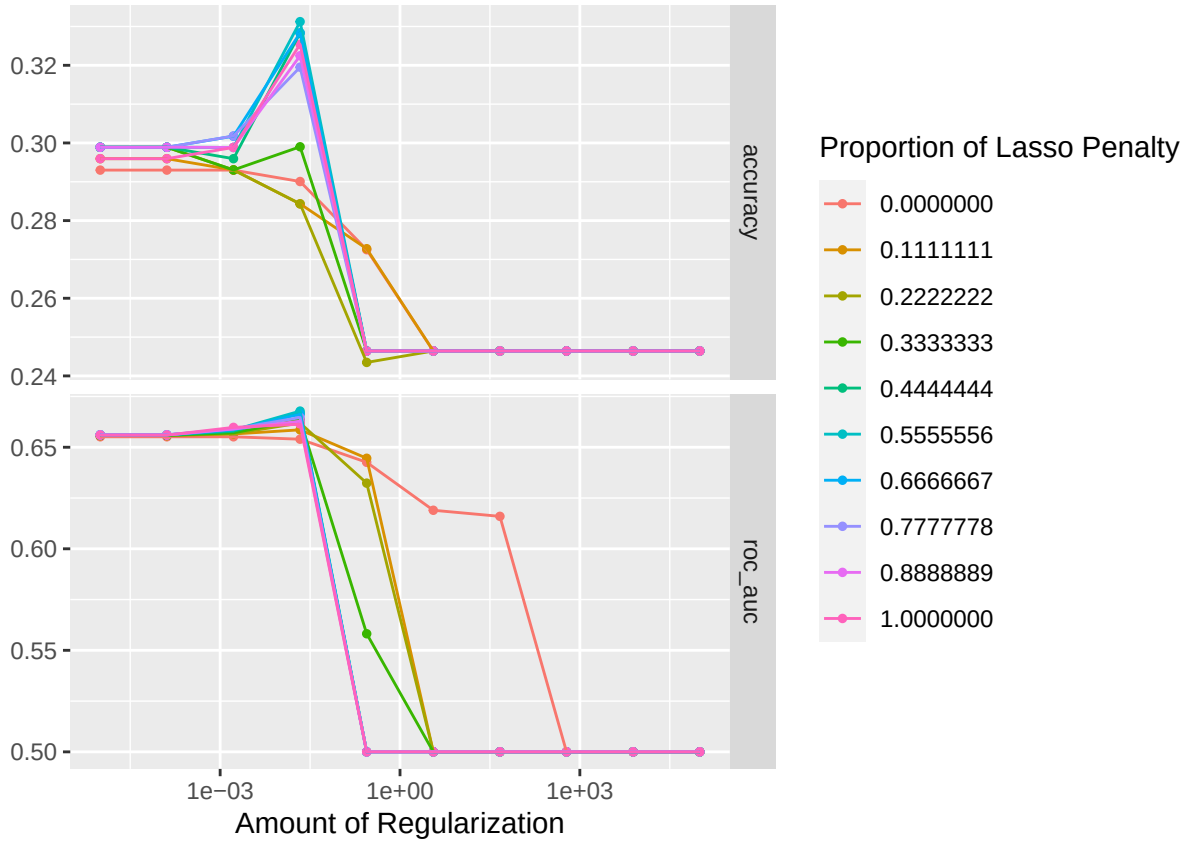
Exercise 6

Fit the models to your folded data using `tune_grid()`.

Use `autoplot()` on the results. What do you notice? Do larger or smaller values of `penalty` and `mixture` produce better accuracy and ROC AUC?

```
tune_res <- tune_grid(
  pk_wkflow,
  resamples = pk_fold,
  grid = penalty_grid
)

autoplot(tune_res)
```



Exercise 7

Use `select_best()` to choose the model that has the optimal `roc_auc`. Then use `finalize_workflow()`, `fit()`, and `augment()` to fit the model to the training set and evaluate its performance on the testing set.

```
best_mod <- select_best(tune_res, metric = "roc_auc")
pk_final <- finalize_workflow(pk_wkflow, best_mod)
pk_final_fit <- fit(pk_final, data= pk_train)
pedicted_data <- augment(pk_final_fit, new_data = pk_test) %>%
  select(type_1, starts_with(".pred"))
```

Exercise 8

Calculate the overall ROC AUC on the testing set.

Then create plots of the different ROC curves, one per level of the outcome. Also make a heat map of the confusion matrix.

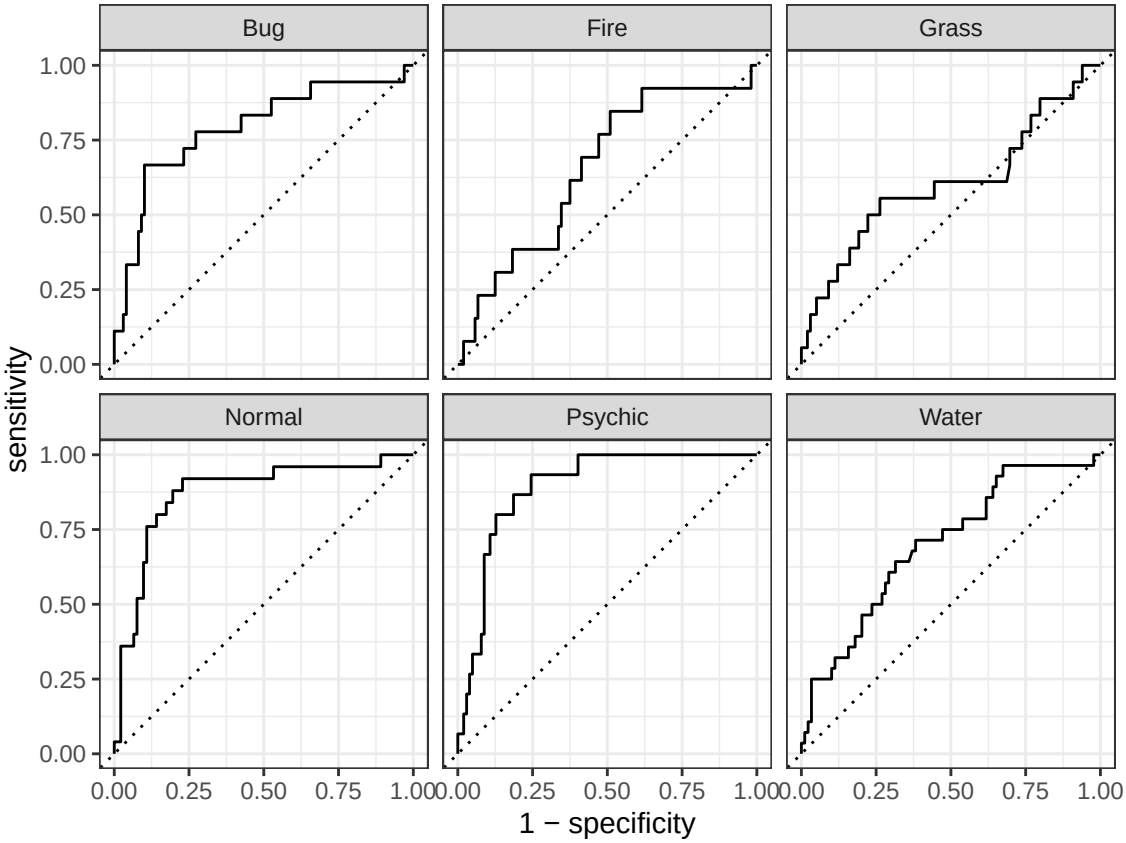
What do you notice? How did your model do? Which Pokemon types is the model best at predicting, and which is it worst at? Do you have any ideas why this might be?

```
pedicted_data %>% roc_auc(type_1, .pred_Bug:.pred_Water)
```

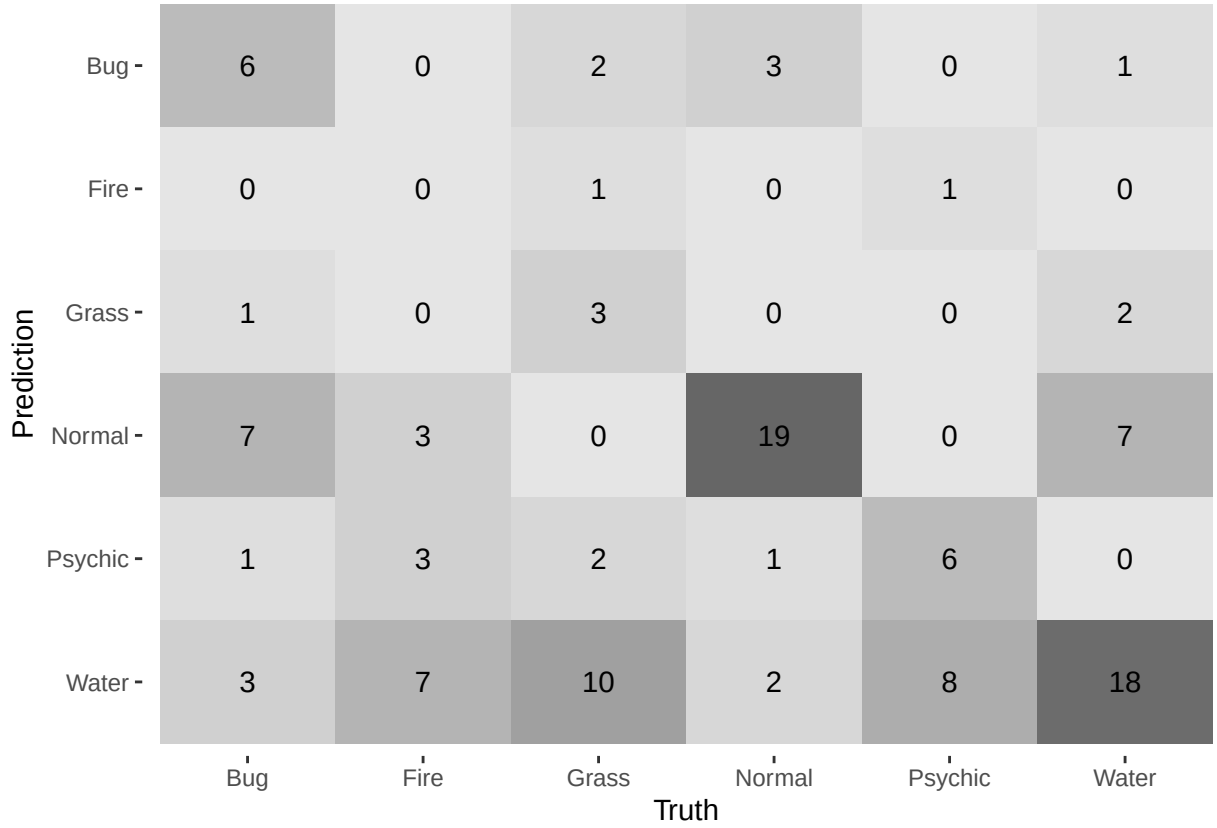
```
## # A tibble: 1 x 3
```

```
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.747
```

```
pedicted_data %>% roc_curve(type_1, .pred_Bug:.pred_Water)%>%
  autoplot()
```




```
augment(pk_final_fit, new_data = pk_test) %>%
  conf_mat(truth = type_1, estimate =.pred_class)%>%
  autoplot("heatmap")
```



Overall roc_auc is 0.75 and is not very high. The model didn't perform well since accuracy is also low. We could see the model predict the normal type pokemon best. I think is maybe the Normal pokemons are similar to others so it had the best prediction.

For 231 Students

Exercise 9

In the 2020-2021 season, Stephen Curry, an NBA basketball player, made 337 out of 801 three point shot attempts (42.1%). Use bootstrap resampling on a sequence of 337 1's (makes) and 464 0's (misses). For each bootstrap sample, compute and save the sample mean (e.g. bootstrap FG% for the player). Use 1000 bootstrap samples to plot a histogram of those values. Compute the 99% bootstrap confidence interval for Stephen Curry's "true" end-of-season FG% using the quantile function in R. Print the endpoints of this interval.