HW5 Xilong Li

Xilong Li (3467966)

2022-05-15

```
library(tidymodels)
```

```
## -- Attaching packages -------------------------------------- tidymodels 0.2.0 --
```

```
## v broom        0.7.12     v recipes      0.2.0
## v dials        0.1.0      v rsample      0.1.1
## v dplyr        1.0.8      v tibble       3.1.6
## v ggplot2      3.3.5      v tidyr        1.2.0
## v infer        1.0.0      v tune         0.2.0
## v modeldata    0.1.1      v workflows    0.2.6
## v parsnip      0.2.1      v workflowsets 0.2.1
## v purrr        0.3.4      v yardstick    0.0.9
```

```
## -- Conflicts ----------------------------------------- tidymodels_conflicts() --
## x purrr::discard() masks scales::discard()
## x dplyr::filter()  masks stats::filter()
## x dplyr::lag()     masks stats::lag()
## x recipes::step()  masks stats::step()
## * Use suppressPackageStartupMessages() to eliminate package startup messages
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v readr   2.1.1     v forcats 0.5.1
## v stringr 1.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-4
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Question 1:

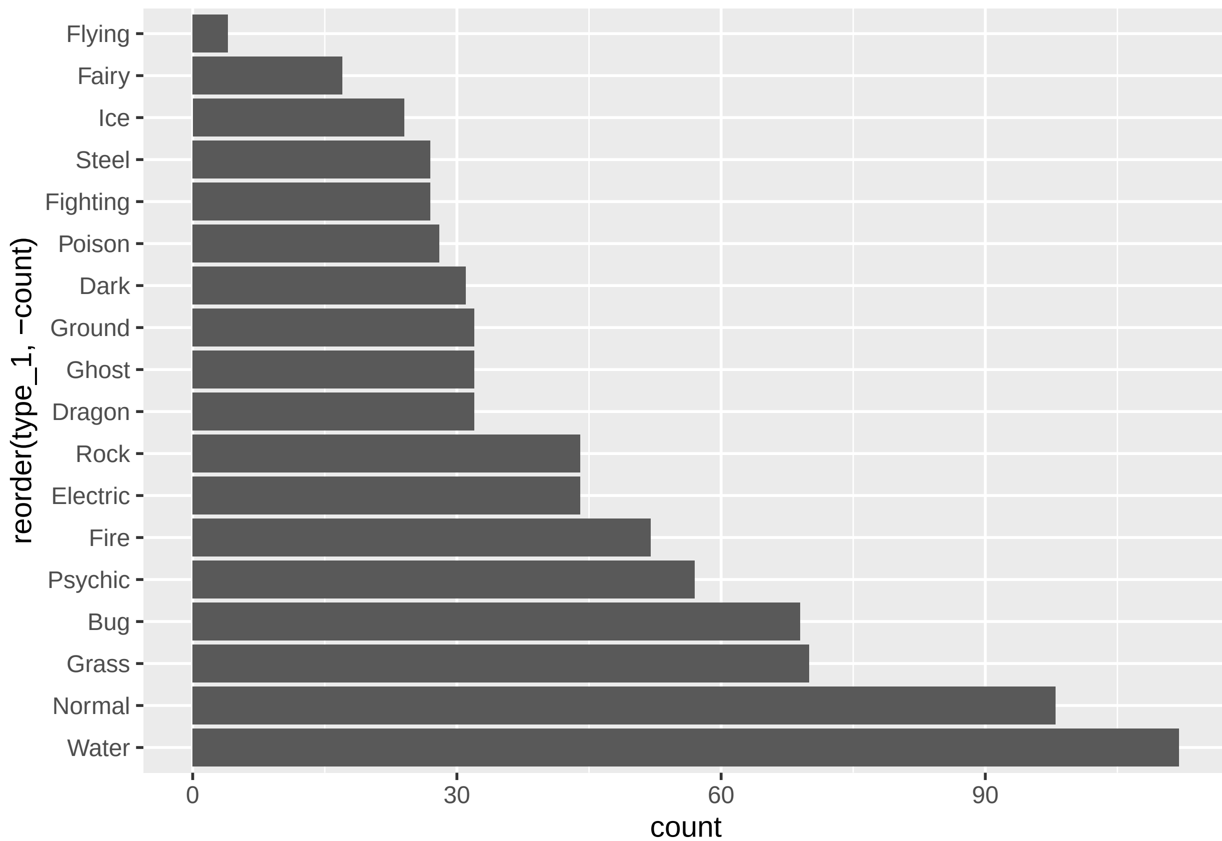
```
pokemon_original <- read.csv("Pokemon.csv")
pokemon <- janitor:: clean_names(dat = pokemon_original)
head(pokemon)
```

```
##   x                  name type_1 type_2 total hp attack defense sp_atk sp_def
## 1 1             Bulbasaur  Grass Poison   318 45     49      49     65     65
## 2 2               Ivysaur  Grass Poison   405 60     62      63     80     80
## 3 3              Venusaur  Grass Poison   525 80     82      83    100    100
## 4 3 VenusaurMega Venusaur  Grass Poison   625 80    100     123    122    120
## 5 4            Charmander   Fire          309 39     52      43     60     50
## 6 5            Charmeleon   Fire          405 58     64      58     80     65
##   speed generation legendary
## 1    45          1     False
## 2    60          1     False
## 3    80          1     False
## 4    80          1     False
## 5    65          1     False
## 6    80          1     False
```

By using the "clean_names" function, the resulting names are unique and consist only of the '_' character, numbers, and letters. Capitalization preferences can be specified using the case parameter. Accented characters are transliterated to ASCII. For example, an "o" with a German umlaut over it becomes "o", and the Spanish character "enye" becomes "n".(This explanation is cited from the website: https://rdrr.io/cran/janitor/man/clean_names.html)
Question 2:

```
copy_pokemon <- pokemon
ordered_data <- copy_pokemon %>%
  group_by(type_1) %>%
```

```
  summarise(count = n()) %>%
  arrange(count)

ggplot(ordered_data, aes(x = count, y = reorder(type_1, -count))) + geom_bar(stat = "identity")
```



As it is shown above, there are 18 classes in total, and classes such as flying, fairy, and ice have fewer pokemons than others,

```
filtered_pokemon <- pokemon %>%
  filter(type_1 %in% c("Bug", "Fire", "Grass", "Normal", "Water", "Psychic"))
final_pokemon <- filtered_pokemon %>%
  mutate(type_1 = factor(type_1),
         legendary = factor(legendary))
dim(final_pokemon)
```

```
## [1] 458  13
```

```
class(final_pokemon$type_1)
```

```
## [1] "factor"
```

```
class(final_pokemon$legendary)
```

```
## [1] "factor"
```

Question 3:

```r
set.seed(2200)

poke_split <- initial_split(final_pokemon, prop = 0.80,
                            strata = type_1)

poke_train <- training(poke_split)
poke_test <- testing(poke_split)

poke_folds <- vfold_cv(poke_train, v = 5, strata = type_1)
class(poke_folds)
```

```
## [1] "vfold_cv"   "rset"       "tbl_df"     "tbl"        "data.frame"
```

By stratifying the folds, we can make sure that the folds are representative of the data, since the splited data is also stratified on type_1. So that the distribution of types in each folds are approximately the same.

Question 4:

```r
poke_recipe <- recipe(type_1 ~
                        legendary +
                        generation +
                        sp_atk +
                        attack +
                        speed +
                        defense +
                        hp +
                        sp_def,
                      data = poke_train) %>%
  step_dummy(legendary,generation) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

class(poke_train$generation)
```

```
## [1] "integer"
```

Question 5:

```r
poke_spec <- multinom_reg (penalty = tune(), mixture = tune()) %>%
  set_engine("glmnet")

poke_workflow <- workflow() %>%
  add_recipe(poke_recipe) %>%
  add_model(poke_spec)
```

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

Thus, there will be 500 models in total, since there are ten levels each for penalty and mixture and 5 folds in the data. ## Question 6:

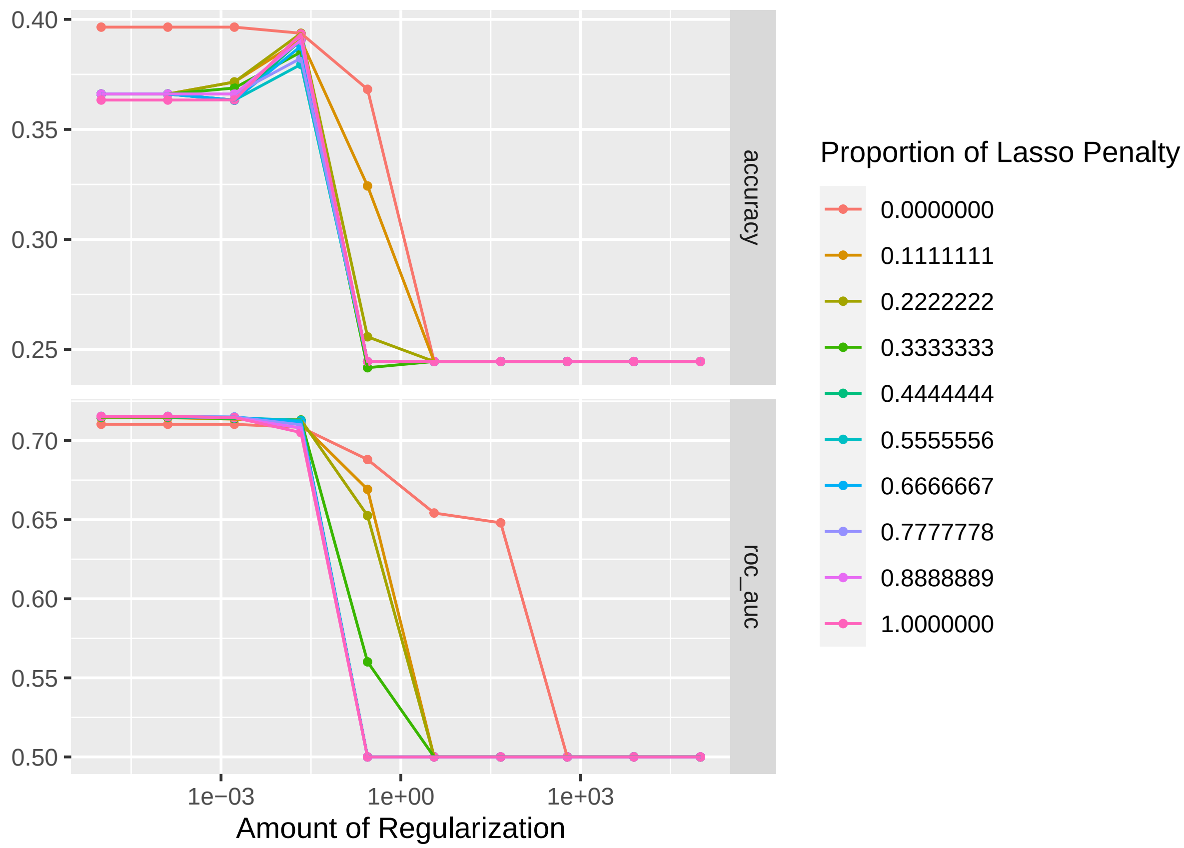
```r
poke_workflow
```

```
## == Workflow ====================================================================
## Preprocessor: Recipe
## Model: multinom_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 3 Recipe Steps
##
## * step_dummy()
## * step_center()
## * step_scale()
##
## -- Model -----------------------------------------------------------------------
## Multinomial Regression Model Specification (classification)
##
## Main Arguments:
##   penalty = tune()
##   mixture = tune()
##
## Computational engine: glmnet
```

```r
tune_res <- tune_grid(
  poke_workflow,
  resamples = poke_folds,
  grid = poke_grid
)
```

```
## ! Fold1: preprocessor 1/1: The following variables are not factor vectors and wil...

## ! Fold2: preprocessor 1/1: The following variables are not factor vectors and wil...

## ! Fold3: preprocessor 1/1: The following variables are not factor vectors and wil...

## ! Fold4: preprocessor 1/1: The following variables are not factor vectors and wil...

## ! Fold5: preprocessor 1/1: The following variables are not factor vectors and wil...
```

```r
autoplot(tune_res)
```



0.40
0.35
0.30
0.25
accuracy
0.70
0.65
0.60
0.55
0.50
roc_auc
1e−03
1e+00
1e+03
Amount of Regularization
Proportion of Lasso Penalty
0.0000000
0.1111111
0.2222222
0.3333333
0.4444444
0.5555556
0.6666667
0.7777778
0.8888889
1.0000000