Pstat231HW5

Zihao Yang

2022-05-15

Q1

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

```
##   X.                  Name Type.1 Type.2 Total HP Attack Defense Sp..Atk
## 1  1             Bulbasaur  Grass Poison   318 45     49      49      65
## 2  2               Ivysaur  Grass Poison   405 60     62      63      80
## 3  3              Venusaur  Grass Poison   525 80     82      83     100
## 4  3 VenusaurMega Venusaur  Grass Poison   625 80    100     123     122
## 5  4            Charmander   Fire          309 39     52      43      60
## 6  5            Charmeleon   Fire          405 58     64      58      80
##   Sp..Def Speed Generation Legendary
## 1      65    45          1     False
## 2      80    60          1     False
## 3     100    80          1     False
## 4     120    80          1     False
## 5      50    65          1     False
## 6      65    80          1     False
```

```
#view(pokemon)
pk <- pokemon %>% clean_names()
head(pk)
```

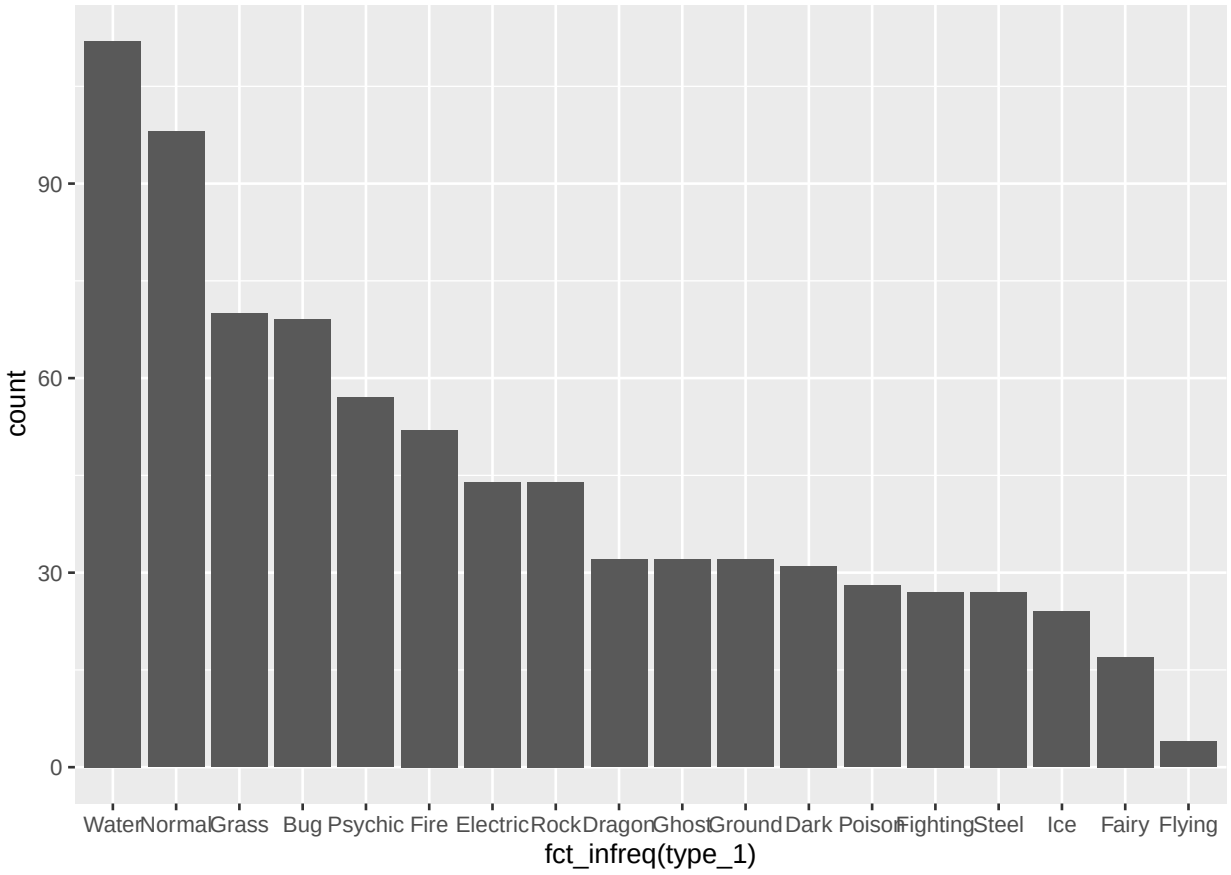
```
##   x                   name type_1 type_2 total hp attack defense sp_atk sp_def
## 1 1             Bulbasaur  Grass Poison   318 45     49      49     65     65
## 2 2               Ivysaur  Grass Poison   405 60     62      63     80     80
## 3 3              Venusaur  Grass Poison   525 80     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
```

```
#view(pk)
```

The variable names of the data change into a clean format. The names are unique and consist only of the space character, numbers, and letters. All the names are in lower-case by default. It is useful because it becomes easier for user to understand and access the variables name.

Q2

```
pk %>% ggplot(aes(x=fct_infreq(type_1))) +
  geom_bar()# Plot the barplot in descending order
```




```
nlevels(factor(pk$type_1))# Count the number of classes
```

```
## [1] 18
```

```
table(fct_infreq(pk$type_1))# Count obersvations in each level in descending order
```

```
##
##    Water   Normal    Grass      Bug  Psychic     Fire Electric     Rock
##      112       98       70       69       57       52       44       44
```

```
##   Dragon    Ghost   Ground     Dark   Poison Fighting    Steel      Ice
##       32       32       32       31       28       27       27       24
##    Fairy   Flying
##       17        4
```

There are 18 classes of pokemons, and the flying type has only 4 pokemons, which with very few pokemon. Besides, Poison, Fighting, Steel, Ice, and Fairy are less than 30.

```
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)
```

Q3

```
#initial split
pk_split <- initial_split(pk2, prop = 0.80,  strata = "type_1")
pk_train <- training(pk_split)
pk_test <- testing(pk_split)
#verify the number of observations
dim(pk_train)
```

```
## [1] 364  13
```

```
dim(pk_test)
```

```
## [1] 94 13
```

```
364/(364+94)
```

```
## [1] 0.79475983
```

The number of observations is correct.

```
pk_folds <- vfold_cv(pk_train,v=5,strata = "type_1")
```

Stratifying is useful because it ensures that there is a representative number in each class. In other words, it ensures the same proportion of each class in both training set and the whole dataset. Then the model will have the same performance on training and testing.

Q4

```
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())
```

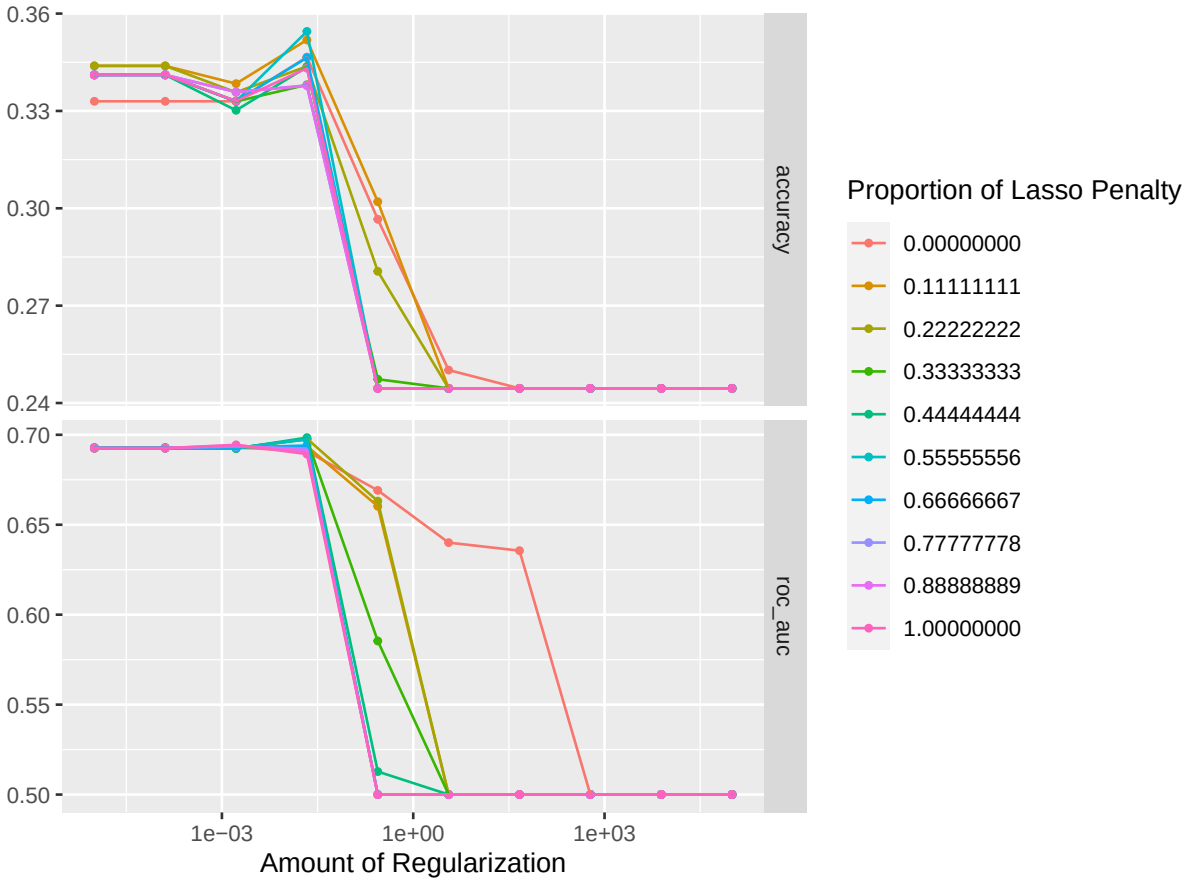
Q5

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

pk_workflow <- workflow() %>%
  add_recipe(pk_recipe) %>%
  add_model(pk_spec)

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

We will fit 500 models. 10 penalty levels times 10 mixture levels then times 5 folds.

Q6

```
tune_res <- tune_grid(pk_workflow,resamples = pk_folds,
                      grid = penalty_grid)
autoplot(tune_res)
```



According to the graphs, the smaller values of penalty and mixture lead to better accuracy and ROC AUC.

Q7

```
pk_best<-select_best(tune_res,metrix="roc_auc")
```

```
## Warning: No value of ‘metric‘ was given; metric ’roc_auc’ will be used.
```

```
pk_final<-finalize_workflow(pk_workflow,pk_best)
pk_fit <- fit(pk_final, data = pk_train)
predict(pk_fit,new_data=pk_test,type="class")
```

```
## # A tibble: 94 x 1
##    .pred_class
##    <fct>
##  1 Water
##  2 Water
##  3 Water
##  4 Water
##  5 Water
##  6 Water
##  7 Normal
##  8 Normal
```

```
##  9 Water
## 10 Water
## # ... with 84 more rows
```

```
test_acc<-augment(pk_fit,new_data=pk_test) %>%
  accuracy(truth=type_1,estimate=.pred_class)
test_acc
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.426
```

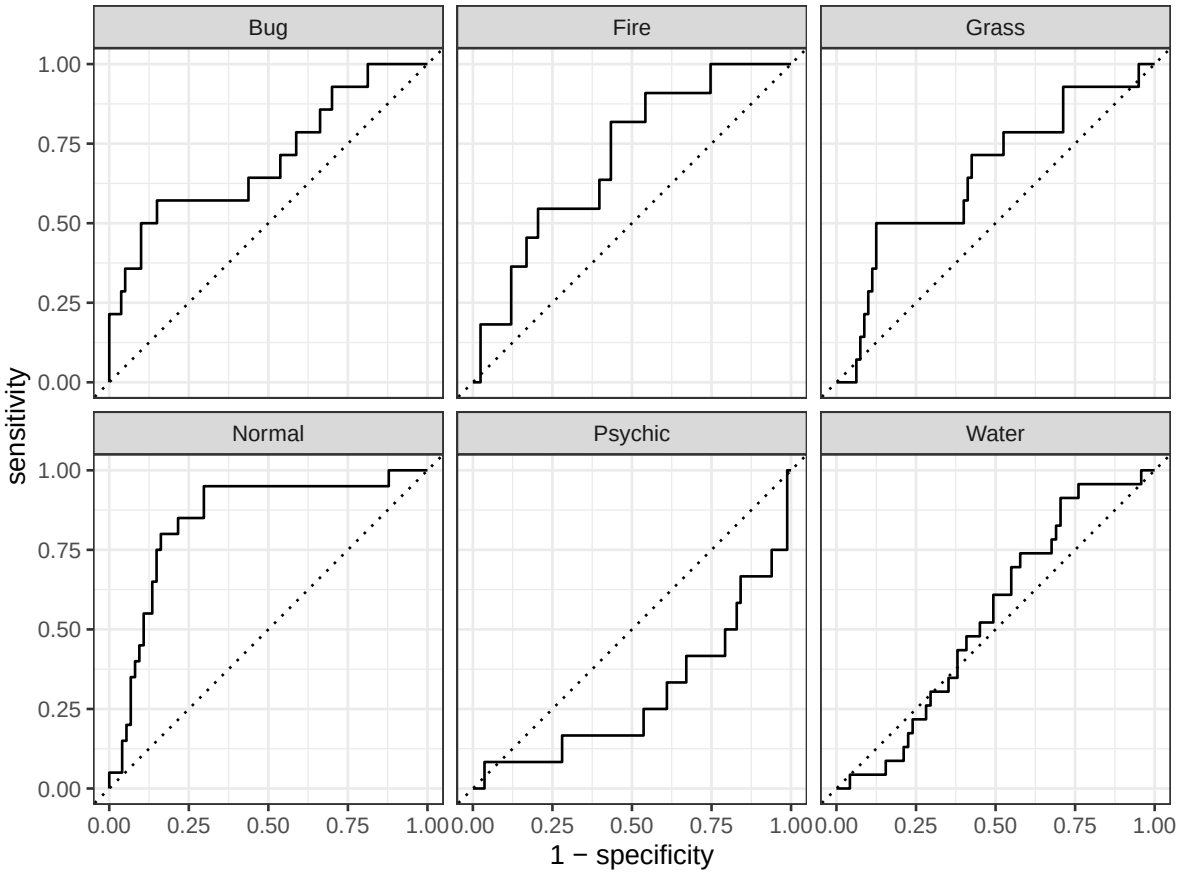
According to the output, the model doesn't perform well on the testing set, because the accuracy is only 0.4255.

Q8

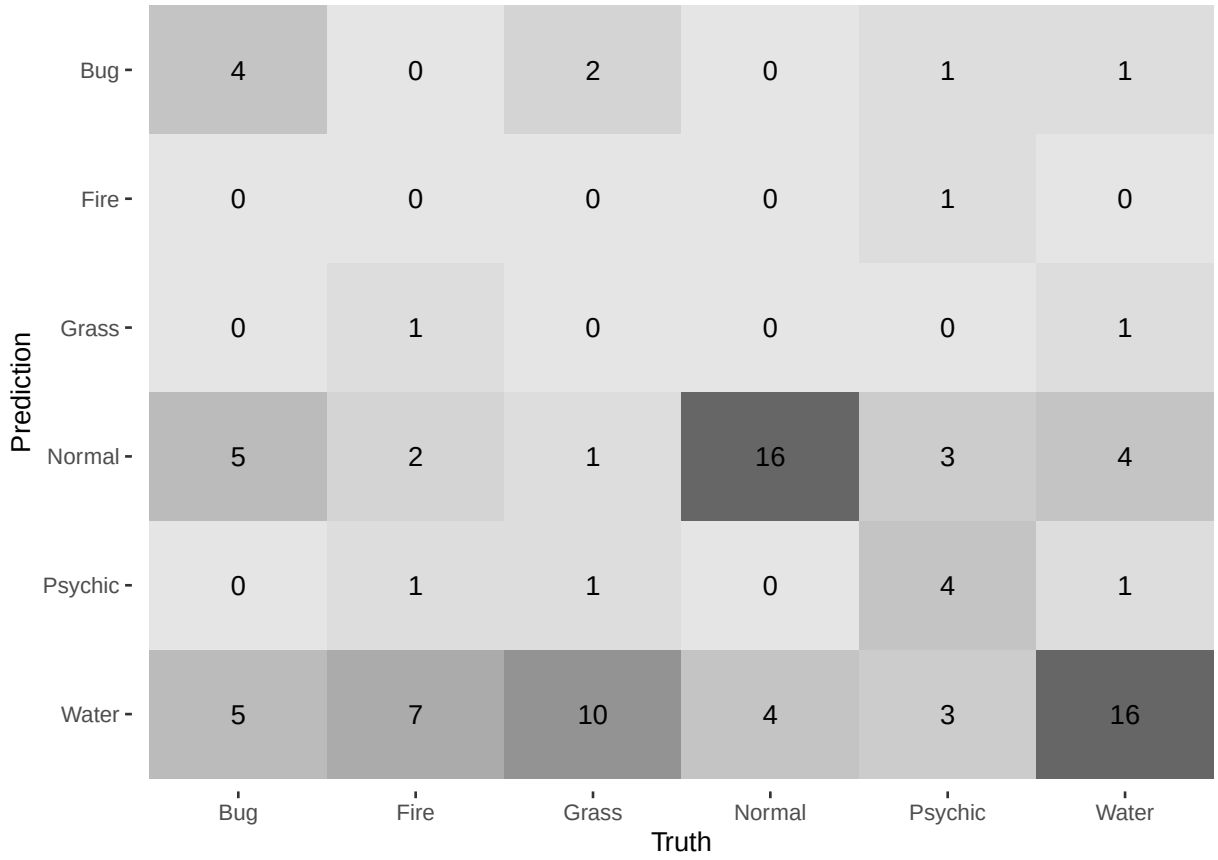
```
roc_auc(augment(pk_fit,new_data=pk_test),
        type_1,.pred_Bug,.pred_Fire,.pred_Grass,.pred_Normal,
        .pred_Water,.pred_Psychic)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.619
```

```
roc_curve(augment(pk_fit,new_data=pk_test),
        type_1,.pred_Bug,.pred_Fire,.pred_Grass,.pred_Normal,
        .pred_Water,.pred_Psychic) %>%
  autoplot()
```




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

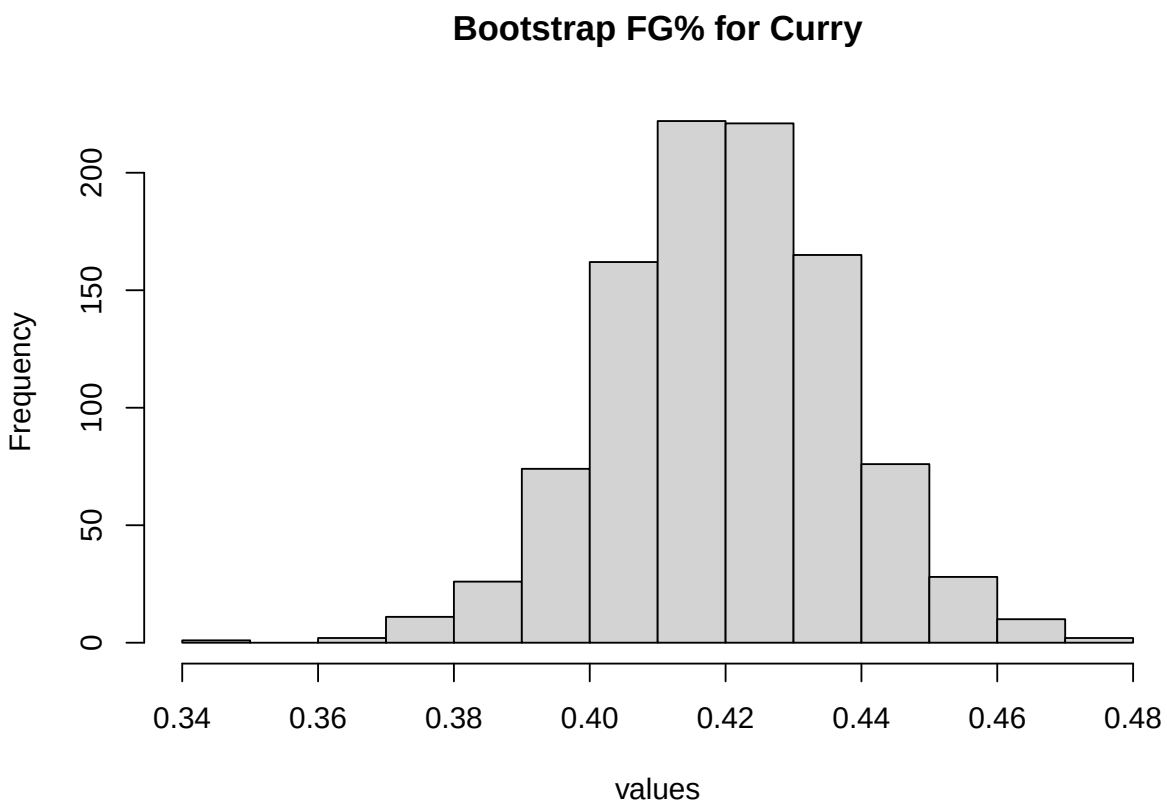


The overall rocauc is not really high with 0.6194, and the model is not performing well because the accuracy is is only 0.4255. According to the area under the roc curve, the model is performing best on Normal type with the largest area, while performing worst on the Psychic type with the smaller area.

It might be caused by the sample size of the Psychic is not big enough, so we don't have enough features to be used for prediction.

Q9

```
SC_shots = rep(1:0, c(337, 464))
PNB <- function(n){
  x = list()
  for (i in 1:n){
    boost = sample(SC_shots, length(SC_shots), replace = T)
    x = append(x, sum(boost)/length(boost))
  }
  return (unlist(x))
}
values = PNB(1000)
hist(values, main = "Bootstrap FG% for Curry")
```




```
quantile(values, probs = seq(0.005, 0.995, 0.99))
```

```
##       0.5%      99.5%
## 0.37202871 0.46442572
```

The 99% CI is [0.3720, 0.4644] with the endpoints rounded from the result above.