Hw6

Peter Chu

11/18/2022

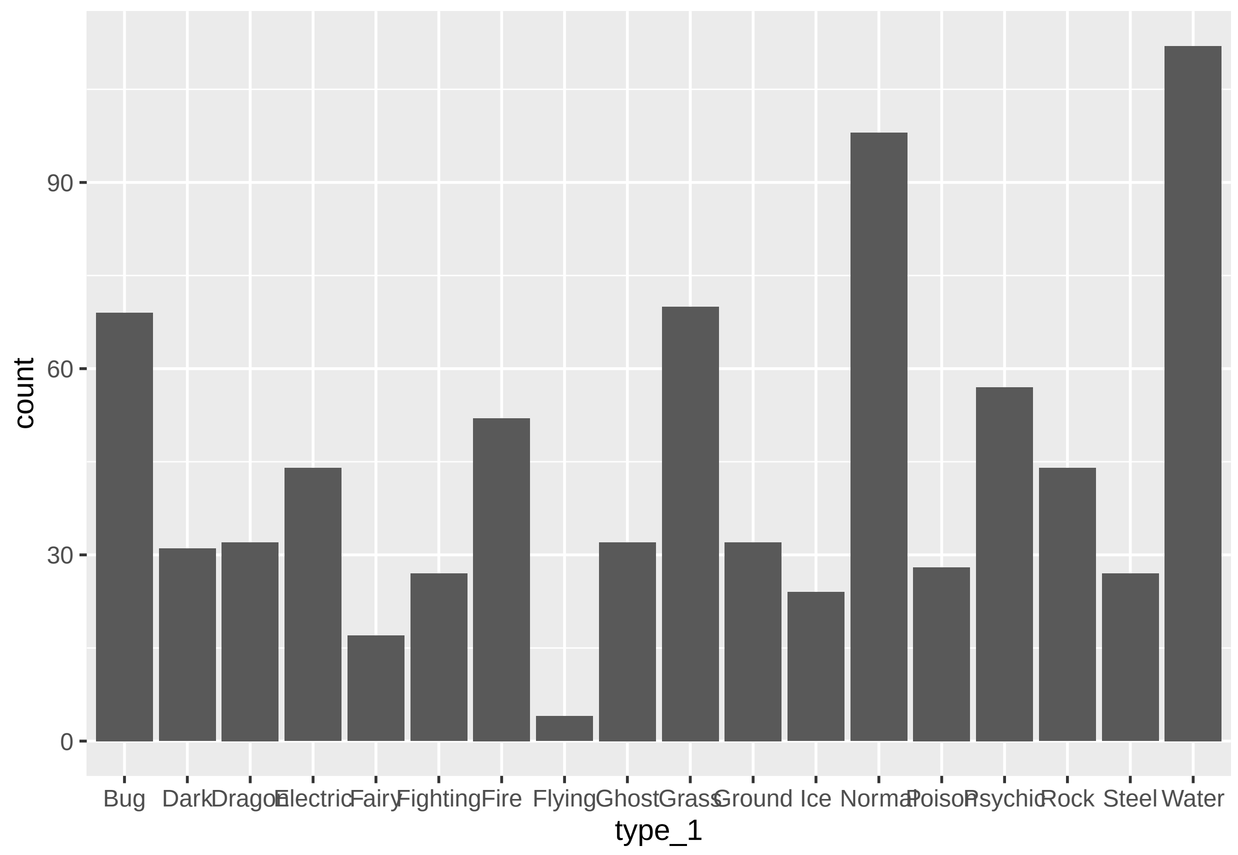
Question 1

```r
data <- read.csv('C:/Users/pchu1/OneDrive/Desktop/231/hw5/Pokemon.csv')

data <- clean_names(data)
```

The clean_names() function made the data variable names more easier and consistent. For example, Sp..Atk became sp_atk. Now all the variables are lower case and have underscores rather than inconsistent numbers of dots. I think it is very useful as it can make code easier to read as well as easier to remember.

Question 2

```r
data %>%
  ggplot((aes(x = type_1))) +
  geom_bar()
```



```
n_distinct(data$type_1)
```

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

```
types <- c('Bug','Fire','Grass','Normal','Water','Psychic')

new_data <- filter(data, data$type_1 %in% types)

data <- new_data

data$type_1 <- as.factor(data$type_1)
data$legendary <- as.factor(data$legendary)
```

There are 18 unique classes of the outcome, type_1. There is a Pokemon type with very few Pokemon like flying. There are less than 10 of them. Some have a lot like Normal and most of the rest have about the same.

Question 3

```
set.seed(102)

data_split <- initial_split(data, strata = type_1, prop = 0.7)
data_train <- training(data_split)
data_test <- testing(data_split)

data_train_fold <- vfold_cv(data_train, v = 5, strata = type_1)
```

Stratisfying the fold is useful so that some folds aren't filled with only a certain type of Pokemon. It allows each fold to have roughly the same amount of Pokemon types which won't skew our results as if we did not do so.

Question 4

```
data_rec <- recipe(type_1 ~ legendary + generation + sp_atk + attack + speed + defense + hp + sp_def, da
  step_dummy(all_nominal_predictors()) %>%
  step_center(all_nominal_predictors()) %>%
  step_scale(all_nominal_predictors())
```

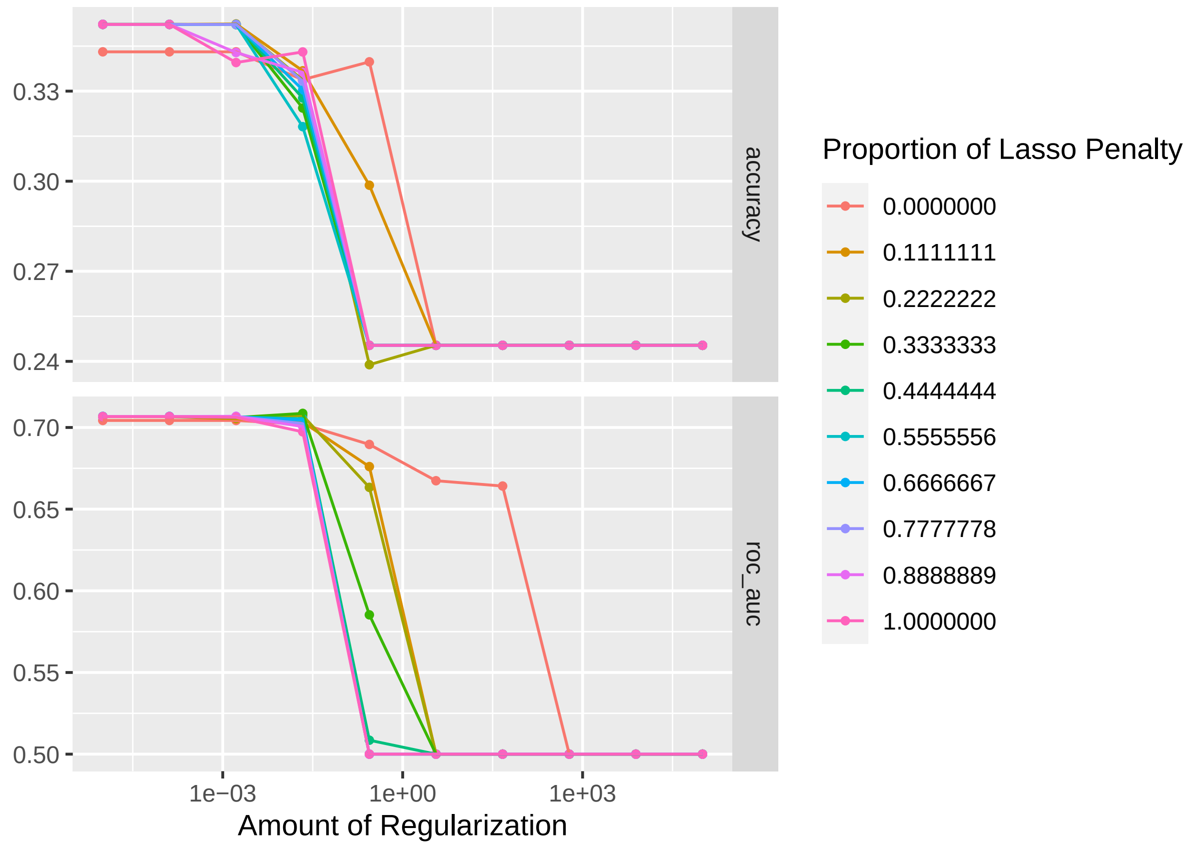
Question 5

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

tuned_wf <- workflow() %>%
  add_recipe(data_rec) %>%
  add_model(mn_model)


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

We will be fitting $10^2 * 10$ which is 1000 models to our folded data.

Question 6

```r
tune_res <- tune_grid(object = tuned_wf, resamples = data_train_fold, grid = degree_grid)

autoplot(tune_res)
```



I am noticing that lower values provide a higher accuracy as well as a higher AUC. A smaller combination of penalty and mixture values provides a better accuracy and ROC AUC

Question 7

```r
best_vals <- select_best(tune_res, metric = 'roc_auc')
best_vals
```

```
## # A tibble: 1 x 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1  0.0215   0.333 Preprocessor1_Model034
```

```r
mn_final <- finalize_workflow(tuned_wf, best_vals)

mn_final_fit <- fit(mn_final, data = data_train)

augment(mn_final_fit, new_data = data_train) %>%
  accuracy(truth = as.factor(type_1) , estimate =.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.377
```

```
augment(mn_final_fit, new_data = data_test) %>%
  accuracy(truth = as.factor(type_1), estimate = .pred_class)
```

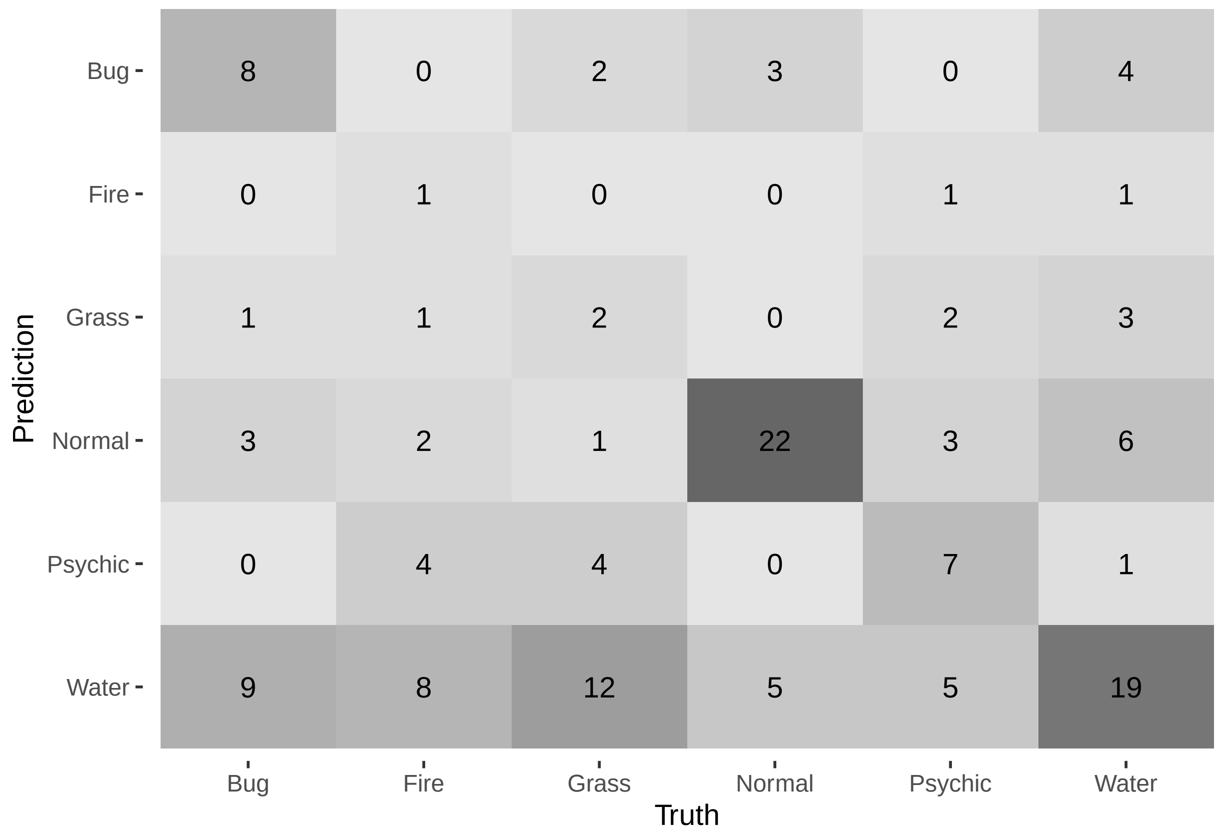
```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.421
```

Question 8

```
augment(mn_final_fit, new_data = data_test) %>%
  roc_auc(type_1, .pred_Bug:.pred_Water)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.704
```

```
augment(mn_final_fit, new_data = data_test) %>%
  conf_mat(type_1, estimate = .pred_class) %>%
  autoplot(type = 'heatmap')
```



Prediction \ Truth	Bug	Fire	Grass	Normal	Psychic	Water
Bug	8	0	2	3	0	4
Fire	0	1	0	0	1	1
Grass	1	1	2	0	2	3
Normal	3	2	1	22	3	6
Psychic	0	4	4	0	7	1
Water	9	8	12	5	5	19

It appears that Normal and Water are most accurately predicted. This may be due to the fact that there are more of these types than any, so there is a larger sample to train and test on. Pokemon types with few count are not as accurately predicted. This model predicts Normal and Water types the best, but also predicted Water poorly. There were 12 mispredictions. This may be because there is such a large number of Water pokemon that it misflags some with similar characteristics.

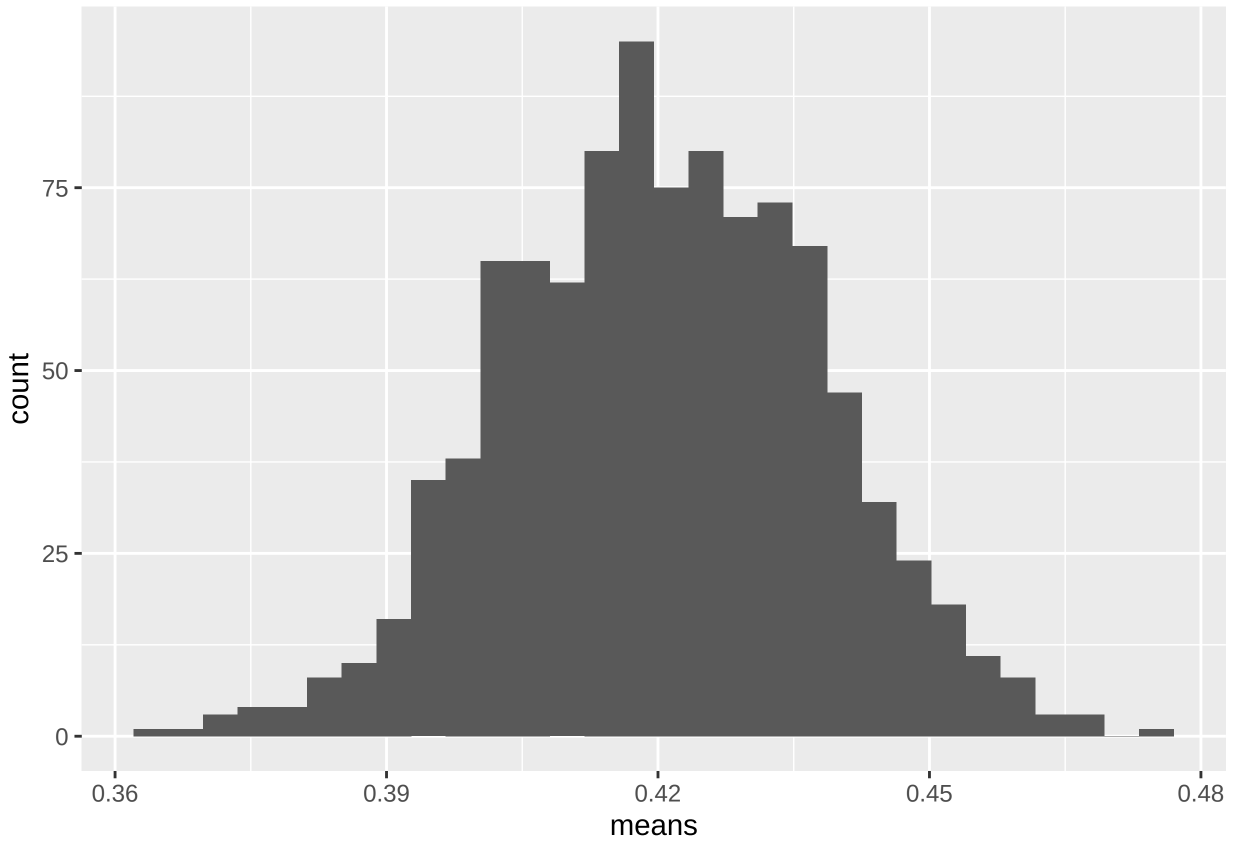
Question 9

```
shots = c(rep(1,337), rep(0,464))
results <- data_frame(num = 1:1000) %>%
  group_by(num) %>%
  dplyr::mutate(means = mean(sample(shots, replace = TRUE)))
```

```
## Warning: ‘data_frame()‘ was deprecated in tibble 1.1.0.
## i Please use ‘tibble()‘ instead.
```

```
results %>%
  ggplot(aes(x = means)) +
  geom_histogram()
```

```
## ‘stat_bin()‘ using ‘bins = 30‘. Pick better value with ‘binwidth‘.
```



```
results <- as.data.frame(results)

quantile(results$means)
```

```
##        0%       25%       50%       75%      100% 
## 0.3645443 0.4079276 0.4207241 0.4332085 0.4756554
```

```
CI(results$means,0.99)
```

```
##     upper      mean     lower 
## 0.4219198 0.4204881 0.4190565
```