131HW5

Scott Shang (8458655)

May 15, 2022

Question1

```
library("tidyverse")
```

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

## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.7     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library("tidymodels")
```

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

## v broom        0.8.0     v rsample      0.1.1
## v dials        0.1.1     v tune         0.2.0
## v infer        1.0.0     v workflows    0.2.6
## v modeldata    0.1.1     v workflowsets 0.2.1
## v parsnip      0.2.1     v yardstick    0.0.9
## v recipes      0.2.0

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

```
library("dplyr")
library("yardstick")
library(readr)
library(pROC)
```

```
## Type 'citation("pROC")' for a citation.

##
## Attaching package: 'pROC'

## The following objects are masked from 'package:stats':
##
##     cov, smooth, var
```

```
library(discrim)
```

```
##
## Attaching package: 'discrim'

## The following object is masked from 'package:dials':
##
##     smoothness
```

```
library(poissonreg)
library(corrr)
library(klaR)
```

```
## Loading required package: MASS

##
## Attaching package: 'MASS'

## The following object is masked from 'package:dplyr':
##
##     select
```

```
library(corrplot)
```

```
## corrplot 0.92 loaded
```

```
library(knitr)
library(MASS)
library(ggplot2)
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
```

```
pkm=read_csv('Pokemon.csv')
```

```
## Rows: 800 Columns: 13

## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (3): Name, Type 1, Type 2
## dbl (9): #, Total, HP, Attack, Defense, Sp. Atk, Sp. Def, Speed, Generation
## lgl (1): Legendary
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
pkm
```

```
## # A tibble: 800 x 13
##      `#` Name   `Type 1` `Type 2` Total    HP Attack Defense `Sp. Atk` `Sp. Def`
##    <dbl> <chr>  <chr>    <chr>    <dbl> <dbl>  <dbl>   <dbl>     <dbl>     <dbl>
##  1     1 Bulba~ Grass    Poison     318    45     49      49        65        65
##  2     2 Ivysa~ Grass    Poison     405    60     62      63        80        80
##  3     3 Venus~ Grass    Poison     525    80     82      83       100       100
##  4     3 Venus~ Grass    Poison     625    80    100     123       122       120
##  5     4 Charm~ Fire     <NA>       309    39     52      43        60        50
##  6     5 Charm~ Fire     <NA>       405    58     64      58        80        65
##  7     6 Chari~ Fire     Flying     534    78     84      78       109        85
##  8     6 Chari~ Fire     Dragon     634    78    130     111       130        85
##  9     6 Chari~ Fire     Flying     634    78    104      78       159       115
## 10     7 Squir~ Water    <NA>       314    44     48      65        50        64
## # ... with 790 more rows, and 3 more variables: Speed <dbl>, Generation <dbl>,
## #   Legendary <lgl>
```

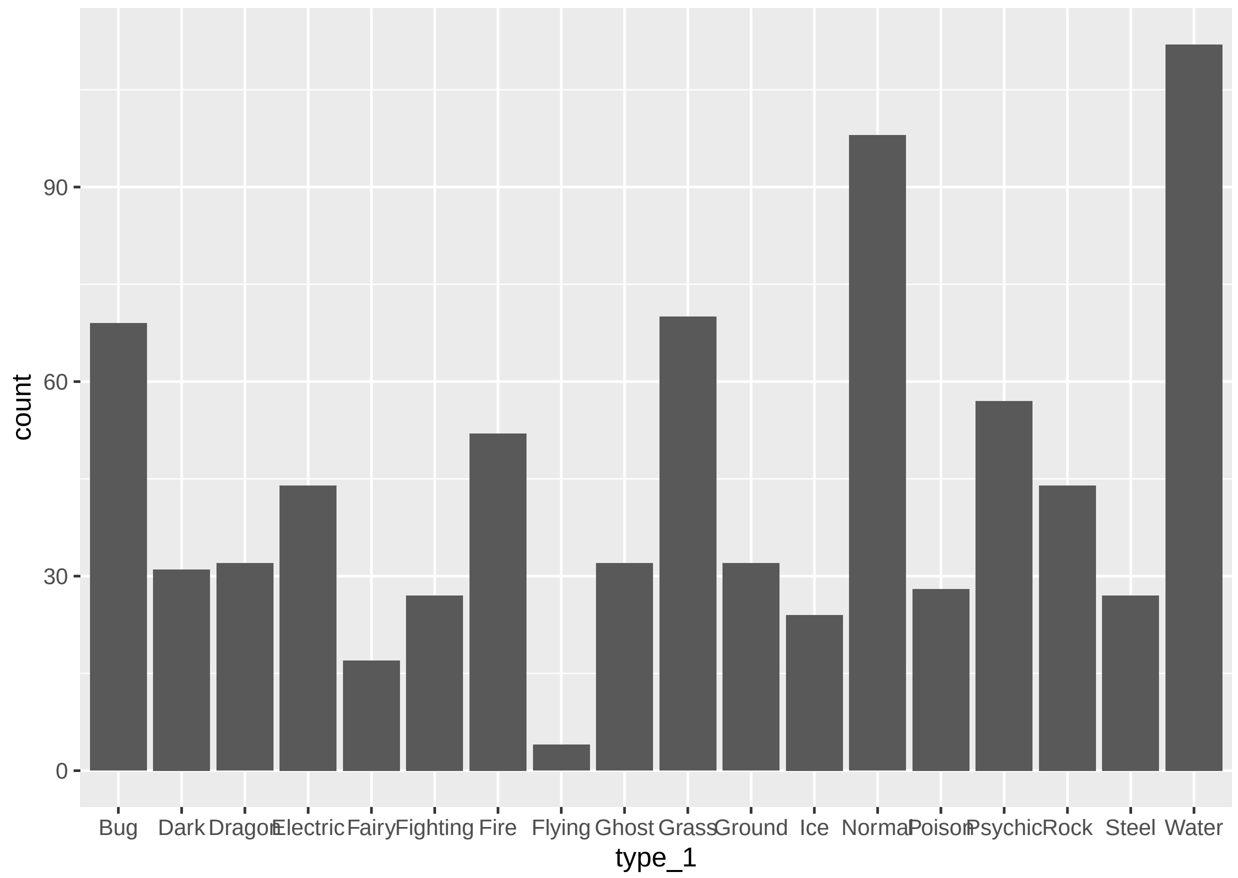
```
pkm=clean_names(pkm)
pkm
```

```
## # A tibble: 800 x 13
##    number name      type_1 type_2 total    hp attack defense sp_atk sp_def speed
##     <dbl> <chr>     <chr>  <chr>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
##  1      1 Bulbasaur Grass  Poison   318    45     49      49     65     65    45
##  2      2 Ivysaur   Grass  Poison   405    60     62      63     80     80    60
##  3      3 Venusaur  Grass  Poison   525    80     82      83    100    100    80
##  4      3 Venusaur~ Grass  Poison   625    80    100     123    122    120    80
##  5      4 Charmand~ Fire   <NA>     309    39     52      43     60     50    65
##  6      5 Charmele~ Fire   <NA>     405    58     64      58     80     65    80
##  7      6 Charizard Fire   Flying   534    78     84      78    109     85   100
##  8      6 Charizar~ Fire   Dragon   634    78    130     111    130     85   100
##  9      6 Charizar~ Fire   Flying   634    78    104      78    159    115   100
## 10      7 Squirtle  Water  <NA>     314    44     48      65     50     64    43
## # ... with 790 more rows, and 2 more variables: generation <dbl>,
## #   legendary <lgl>
```

After using clean_names() on our data, it returns name with only lowercase letter, with _ as a separator, and convert symbol "#" to "number". It is helpful because it cleans up the names of variables.

Question2

```
ggplot(pkm, aes(x = type_1)) +
  geom_bar()
```



There are 18 classes of the outcome. The flying type has the least numbers of Pokemon. Dark, dragon, fairy, fighting, ghost, ground, ice, poison and steel types also have less pokemon comparing to others.

```
pkm=filter(pkm,type_1 %in% c("Bug","Fire","Grass","Normal","Water","Psychic"))
pkm
```

```
## # A tibble: 458 x 13
##    number name      type_1 type_2 total    hp attack defense sp_atk sp_def speed
##     <dbl> <chr>     <chr>  <chr>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
##  1      1 Bulbasaur Grass  Poison   318    45     49      49     65     65    45
##  2      2 Ivysaur   Grass  Poison   405    60     62      63     80     80    60
##  3      3 Venusaur  Grass  Poison   525    80     82      83    100    100    80
##  4      3 Venusaur~ Grass  Poison   625    80    100     123    122    120    80
##  5      4 Charmand~ Fire   <NA>     309    39     52      43     60     50    65
##  6      5 Charmele~ Fire   <NA>     405    58     64      58     80     65    80
##  7      6 Charizard Fire   Flying   534    78     84      78    109     85   100
##  8      6 Charizar~ Fire   Dragon   634    78    130     111    130     85   100
##  9      6 Charizar~ Fire   Flying   634    78    104      78    159    115   100
## 10      7 Squirtle  Water  <NA>     314    44     48      65     50     64    43
## # ... with 448 more rows, and 2 more variables: generation <dbl>,
## #   legendary <lgl>
```

```
pkm$type_1=as.factor(pkm$type_1)
pkm$legendary=as.factor(pkm$legendary)
pkm$generation=as.factor(pkm$generation)
pkm
```

```
## # A tibble: 458 x 13
##    number name      type_1 type_2 total    hp attack defense sp_atk sp_def speed
##     <dbl> <chr>     <fct>  <chr>  <dbl> <dbl>  <dbl>   <dbl>  <dbl>  <dbl> <dbl>
##  1      1 Bulbasaur Grass  Poison   318    45     49      49     65     65    45
##  2      2 Ivysaur   Grass  Poison   405    60     62      63     80     80    60
##  3      3 Venusaur  Grass  Poison   525    80     82      83    100    100    80
##  4      3 Venusaur~ Grass  Poison   625    80    100     123    122    120    80
##  5      4 Charmand~ Fire   <NA>     309    39     52      43     60     50    65
##  6      5 Charmele~ Fire   <NA>     405    58     64      58     80     65    80
##  7      6 Charizard Fire   Flying   534    78     84      78    109     85   100
##  8      6 Charizar~ Fire   Dragon   634    78    130     111    130     85   100
##  9      6 Charizar~ Fire   Flying   634    78    104      78    159    115   100
## 10      7 Squirtle  Water  <NA>     314    44     48      65     50     64    43
## # ... with 448 more rows, and 2 more variables: generation <fct>,
## #   legendary <fct>
```

Question3

```
set.seed(1234)
pkm_split=initial_split(pkm,prop=0.70,strata=type_1)
train=training(pkm_split)
test=testing(pkm_split)
dim(pkm)*0.7
```

```
## [1] 320.6   9.1
```

```
dim(train)
```

```
## [1] 318  13
```

Yes, the training and test sets have the desired number of observations.

```r
folds=vfold_cv(train,v=5,strata=type_1)
folds
```

```
## #  5-fold cross-validation using stratification
## # A tibble: 5 x 2
##   splits           id
##   <list>           <chr>
## 1 <split [252/66]> Fold1
## 2 <split [253/65]> Fold2
## 3 <split [253/65]> Fold3
## 4 <split [256/62]> Fold4
## 5 <split [258/60]> Fold5
```

Stratifying the folds is useful because it makes sure that the folds are representative of the whole data set.

Question4

```r
rcp=recipe(type_1~legendary+generation+sp_atk+attack+speed+defense+hp+sp_def,data=train) %>%
  step_dummy(legendary) %>%
  step_dummy(generation) %>%
  step_normalize(all_predictors())
```

Question5

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

wf=workflow() %>%
  add_recipe(rcp) %>%
  add_model(reg)

grid<-grid_regular(penalty(range = c(-5, 5)),mixture(),levels = 10)
grid
```

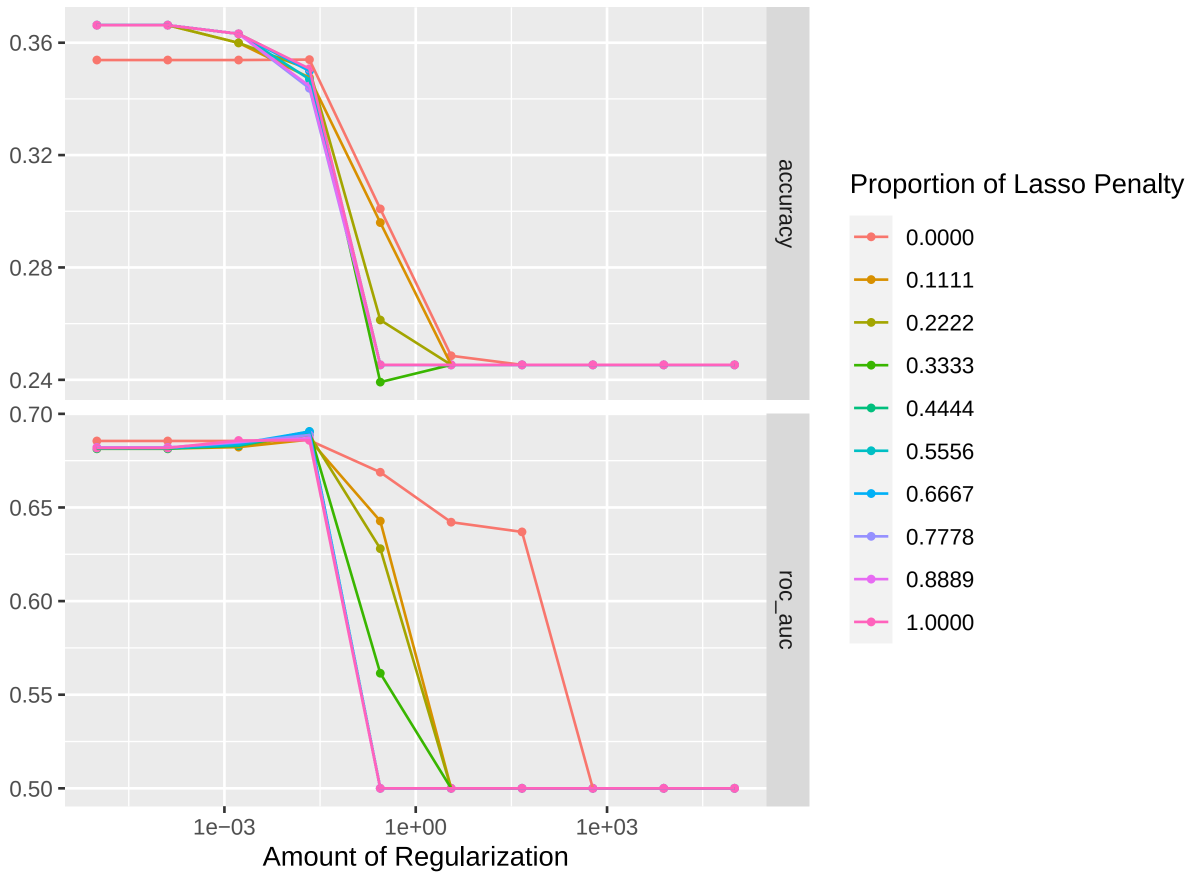
```
## # A tibble: 100 x 2
##          penalty mixture
##            <dbl>   <dbl>
##  1      0.00001        0
##  2      0.000129       0
##  3      0.00167        0
##  4      0.0215         0
##  5      0.278          0
##  6      3.59           0
##  7     46.4            0
##  8    599.             0
##  9   7743.             0
## 10 100000              0
## # ... with 90 more rows
```

We will be fitting 500 models, since we have 10 levels of penalty, 10 levels of mixture, and 5 folds.

Question6

```r
tune_res=tune_grid(wf,resamples=folds,grid=grid)
```

```r
autoplot(tune_res)
```



What do you notice? Do larger or smaller values of penalty and mixture produce better accuracy and ROC AUC?

Smaller values of penalty and mixture produce better accuracy and ROC AUC. Mixture values have no strong impact when penalty is large or small, but do have better accuracy and ROC AUC for mid-range penalty.

Question7

```r
best=select_best(tune_res,metric="roc_auc")
final=finalize_workflow(wf,best)

final_fit=fit(final,data=train)

augment(final_fit,new_data=test) %>%
  accuracy(truth=type_1,estimate=.pred_class)
```

```
## # A tibble: 1 x 3
##   .metric  .estimator .estimate
##   <chr>    <chr>          <dbl>
## 1 accuracy multiclass     0.364
```

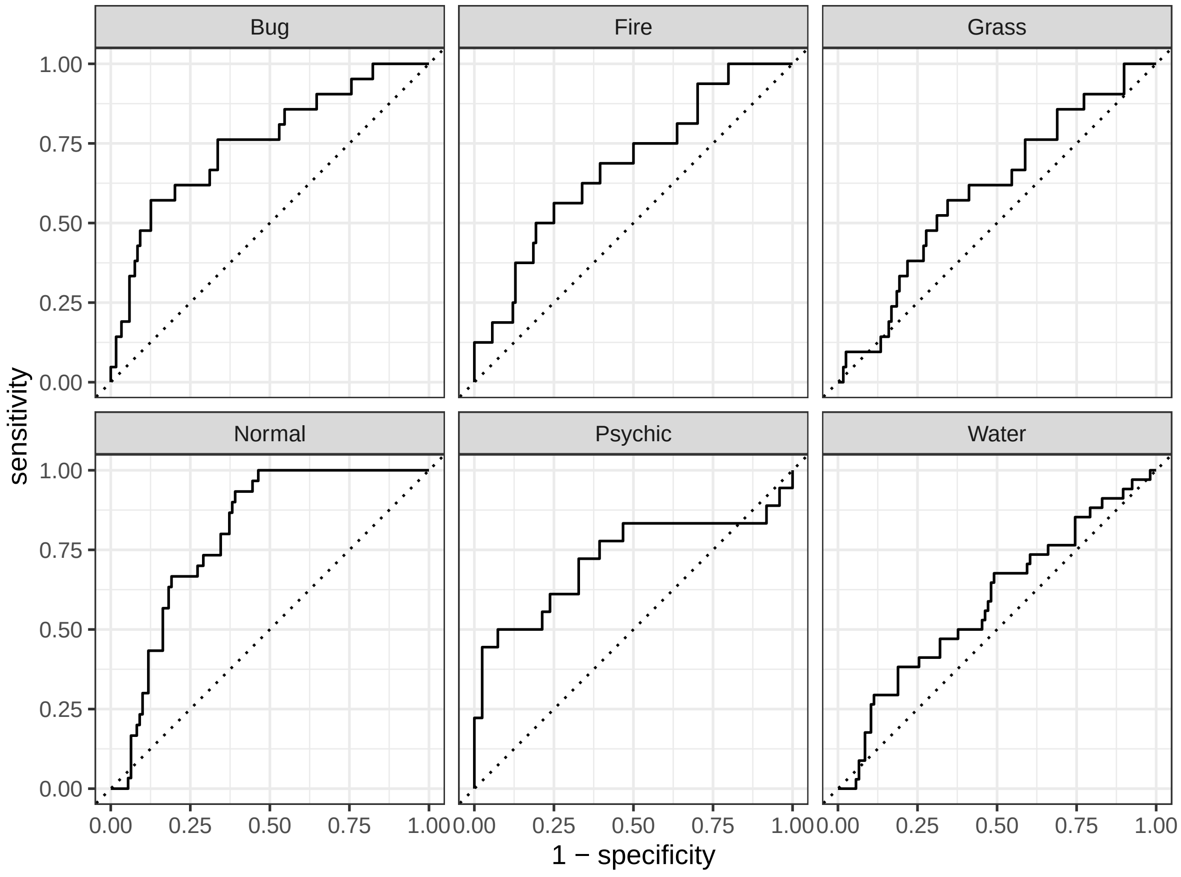
The performance on the testing set is pretty bad.

Question8

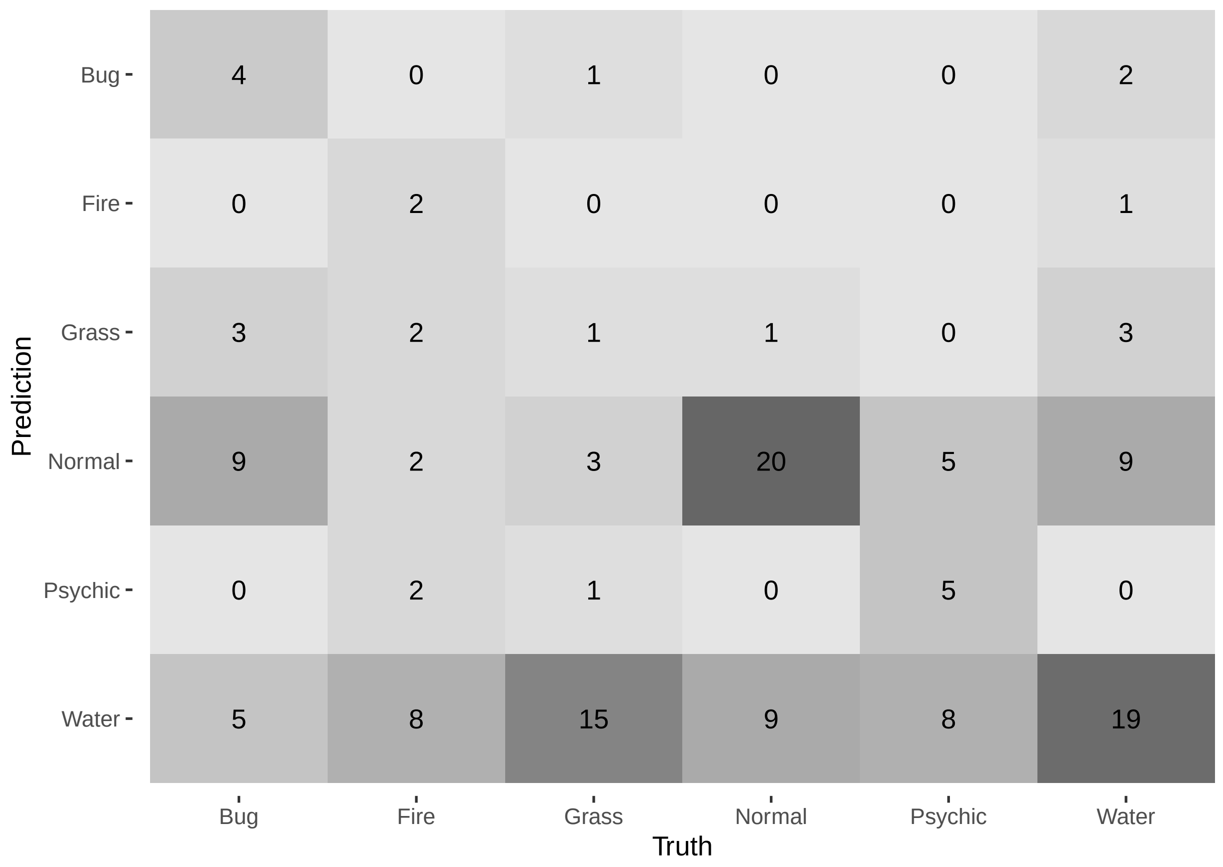
```
augment(final_fit,new_data=test) %>%
  roc_auc(truth=type_1,estimate=.pred_Bug:.pred_Water)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.688
```

```
augment(final_fit,new_data=test) %>%
  roc_curve(truth=type_1,estimate=.pred_Bug:.pred_Water) %>%
  autoplot()
```



```
augment(final_fit,new_data=test) %>%
  conf_mat(truth=type_1,estimate=.pred_class) %>%
  autoplot(type="heatmap")
```



First of all, our model doesn't perform well. From the ROC curves, we can tell that normal type is the model best at predicting, which is also proved by the heatmap of the confusion matrix, followed by bug type and psychic type. Water type is the worst according to ROC curve, but heatmap shows different result. This might because water type has the most observations. The reason behind the poor performance of our model might be not enough observations. With less than 100 observations per type on average, it can be hard to achieve an accurate model.