hw5

2022-11-17

Set up

```
library(tidymodels)
```

```
## ── Attaching packages ─────────────────────────────────────── tidymodels 1.0.0 ──
```

```
## ✓ broom        1.0.1      ✓ recipes      1.0.1
## ✓ dials        1.1.0      ✓ rsample      1.1.0
## ✓ dplyr        1.0.10     ✓ tibble       3.1.8
## ✓ ggplot2      3.4.0      ✓ tidyr        1.2.1
## ✓ infer        1.0.3      ✓ tune         1.0.1
## ✓ modeldata    1.0.1      ✓ workflows    1.1.0
## ✓ parsnip      1.0.2      ✓ workflowsets 1.0.0
## ✓ purrr        0.3.4      ✓ yardstick    1.1.0
```

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

```
library(tidyverse)
```

```
## ── Attaching packages
## ──────────────────────────────────────
## tidyverse 1.3.2 ──
```

```
## ✓ readr   2.1.2     ✓ forcats 0.5.2
## ✓ stringr 1.4.1
## ── Conflicts ──────────────────────────────────────── tidyverse_conflicts() ──
## ✖ readr::col_factor() masks scales::col_factor()
## ✖ purrr::discard()    masks scales::discard()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ stringr::fixed()    masks recipes::fixed()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ readr::spec()       masks yardstick::spec()
```

```
library(dplyr)
#library(corrr)
library(ggplot2)
library(discrim)
```

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

```
library(klaR)
```

```
## Loading required package: MASS
##
## Attaching package: 'MASS'
##
## The following object is masked from 'package:dplyr':
##
##     select
```

```
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
```

```
tidymodels_prefer()
pokemon <- read.csv("data/Pokemon.csv")
set.seed(1234)
head(pokemon)
```

```
##   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
```

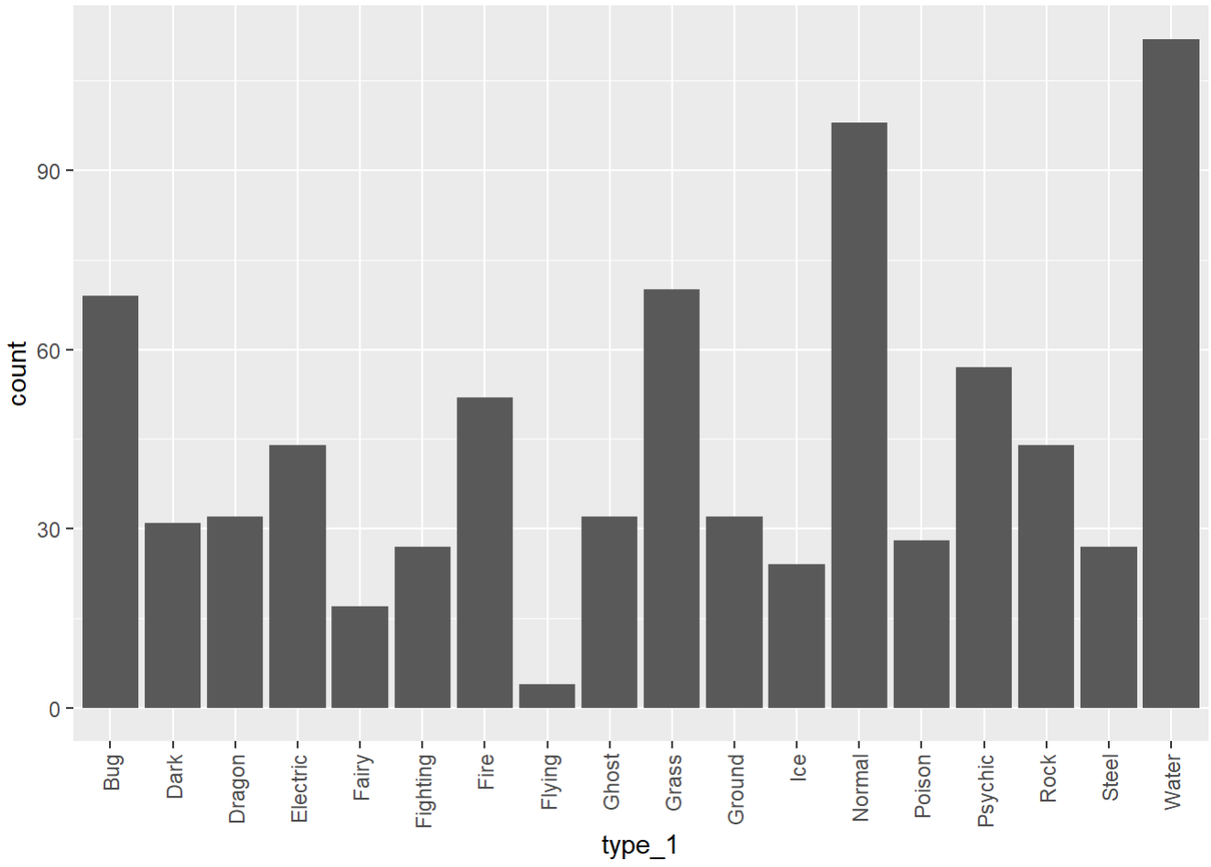
Question 1

```
library(janitor)
pokemon <- pokemon %>%
  clean_names() #names lowercase without periods/commas
#now underscore
```

With clean_names(), we see that the variables how are all lowercase and don't have periods or commas anymore and are replaced with underscore. This is useful so that the variables have a consistent variable format.

Question 2

```
barchart <- ggplot(data = pokemon, aes(x = type_1)) +
  geom_bar() +
  theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))
barchart
```




```
#filter out rarer classes
pokemon_filter <- pokemon %>%
  filter(type_1 == c("Bug", "Fire", "Grass", "Normal", "Water", "Psychic"))
```

```
## Warning in type_1 == c("Bug", "Fire", "Grass", "Normal", "Water", "Psychic"):
## longer object length is not a multiple of shorter object length
```

```
pokemon_factor <- pokemon_filter %>%
  mutate(type_1 = factor(type_1),
         legendary = factor(legendary),
         generation = factor(generation))
```

There are 18 classes of type_1. There are a few Pokemon that are Flying and Fairy. There are also around 35 or less Pokemon that are Dark, Dragon, Fighting, Ghost, Ground, Ice, Poison, and Steel.

Exercise 3

```
pokemon_split <- initial_split(pokemon_factor,
                               prop = 0.80,
                               strata = type_1)
pokemon_split
```

```
## <Training/Testing/Total>
## <63/19/82>
```

```
pokemon_train <- training(pokemon_split)
pokemon_test <- testing(pokemon_split)

pokemon_folds <- vfold_cv(pokemon_train, v = 5,
                          strata = type_1)
pokemon_folds
```

```
## #  5-fold cross-validation using stratification
## # A tibble: 5 × 2
##   splits          id
##   <list>          <chr>
## 1 <split [48/15]> Fold1
## 2 <split [49/14]> Fold2
## 3 <split [50/13]> Fold3
## 4 <split [52/11]> Fold4
## 5 <split [53/10]> Fold5
```

Stratifying the folds might be useful as we could divide also by the type of Pokemon to analyze.

Exercise 4

```
pokemon_recipe <-
  recipe(type_1 ~ legendary + generation + sp_atk + attack + speed + defense +
           hp + sp_def, data = pokemon_train) %>%
  step_dummy(c("legendary", "generation")) %>% #all_nominal_predictors?
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
pokemon_recipe
```

```
## Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          8
##
## Operations:
##
## Dummy variables from c("legendary", "generation")
## Centering for all_predictors()
## Scaling for all_predictors()
```

Exercise 5

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

```
## Multinomial Regression Model Specification (classification)
##
## Main Arguments:
##   penalty = tune()
##   mixture = tune()
##
## Computational engine: glmnet
```

```
multi_wrkflow <- workflow() %>%
  add_model(multi_model) %>%
  add_recipe(pokemon_recipe)

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

```
## # A tibble: 100 × 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
```

I'll be fitting 100 models.

Exercise 6

```
#use autoplot()
tune_res <- tune_grid(
  multi_wrkflow,
  resamples = pokemon_folds,
  grid = pokemon_grid
)
```

```
## ! Fold1: preprocessor 1/1, model 1/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 2/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 3/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 4/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 5/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 6/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 7/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 8/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 9/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold1: preprocessor 1/1, model 10/10: one multinomial or binomial class has fewer than 8  o
bservations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 1/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 2/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 3/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 4/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 5/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 6/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 7/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 8/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 9/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold2: preprocessor 1/1, model 10/10: one multinomial or binomial class has fewer than 8  o
bservations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 1/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 2/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 3/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 4/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 5/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 6/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 7/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 8/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 9/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold3: preprocessor 1/1, model 10/10: one multinomial or binomial class has fewer than 8  o
bservations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 1/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 2/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 3/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 4/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 5/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 6/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 7/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 8/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 9/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold4: preprocessor 1/1, model 10/10: one multinomial or binomial class has fewer than 8  o
bservations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 1/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 2/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 3/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 4/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 5/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 6/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

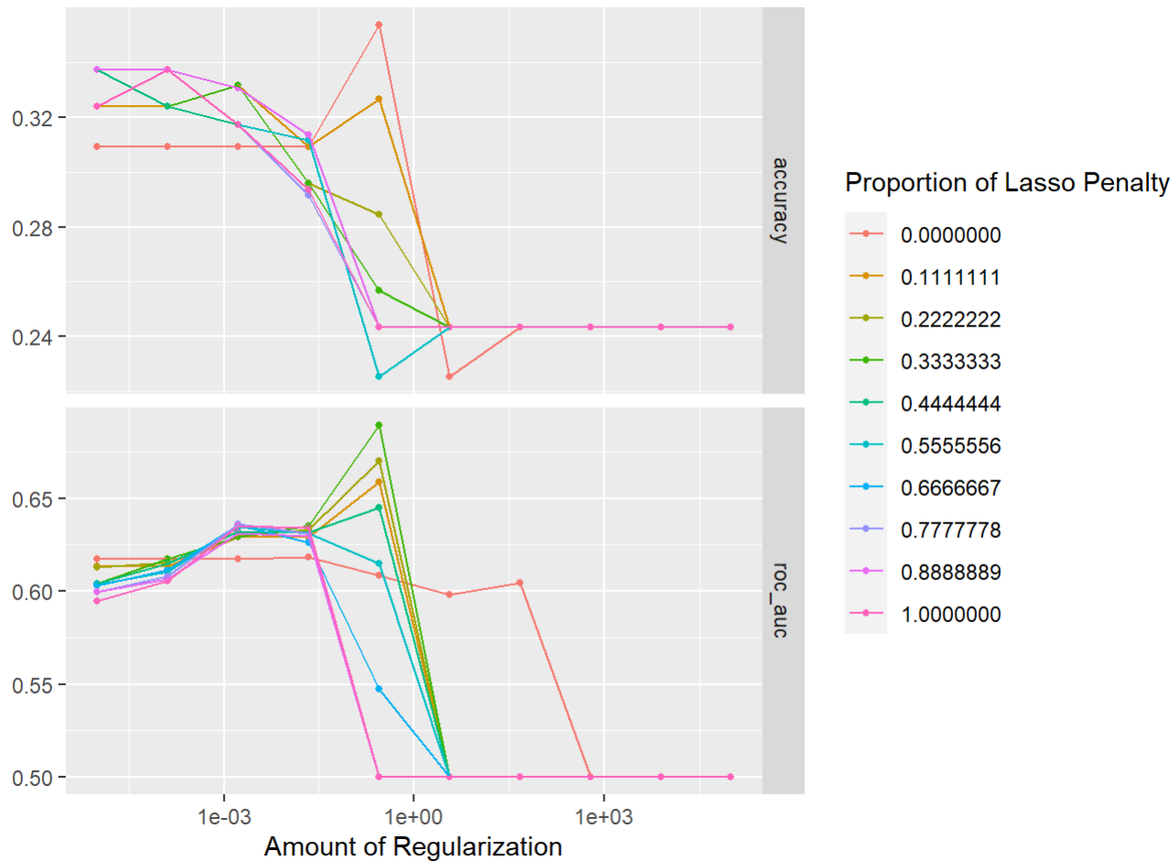
```
## ! Fold5: preprocessor 1/1, model 7/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 8/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 9/10: one multinomial or binomial class has fewer than 8  ob
servations; danger...
```

```
## ! Fold5: preprocessor 1/1, model 10/10: one multinomial or binomial class has fewer than 8  o
bservations; danger...
```

```
autoplot(tune_res)
```



I notice that a lot a lot of proportions of lasso penalty go down once it approaches 1e+00 amount of regularization. Smaller values of penalty and mixture produce better accuracy and ROC AUC.

Exercise 7

```
best_penalty <- select_best(tune_res, metric = "roc_auc") #rsq??
best_penalty
```

```
## # A tibble: 1 × 3
##   penalty mixture .config
##     <dbl>   <dbl> <chr>
## 1   0.278   0.333 Preprocessor1_Model035
```

```
ridge_final <- finalize_workflow(multi_wrkflow, best_penalty)

ridge_final_fit <- fit(ridge_final, data = pokemon_train)
```

```
## Warning in lognet(xd, is.sparse, ix, jx, y, weights, offset, alpha, nobs, : one
## multinomial or binomial class has fewer than 8 observations; dangerous ground
```

```
roc_auc1 <- augment(ridge_final_fit, new_data = pokemon_test) %>%
  select(type_1, starts_with(".pred"))
roc_auc1#rsq??? #not roc_auc or multi_metric
```

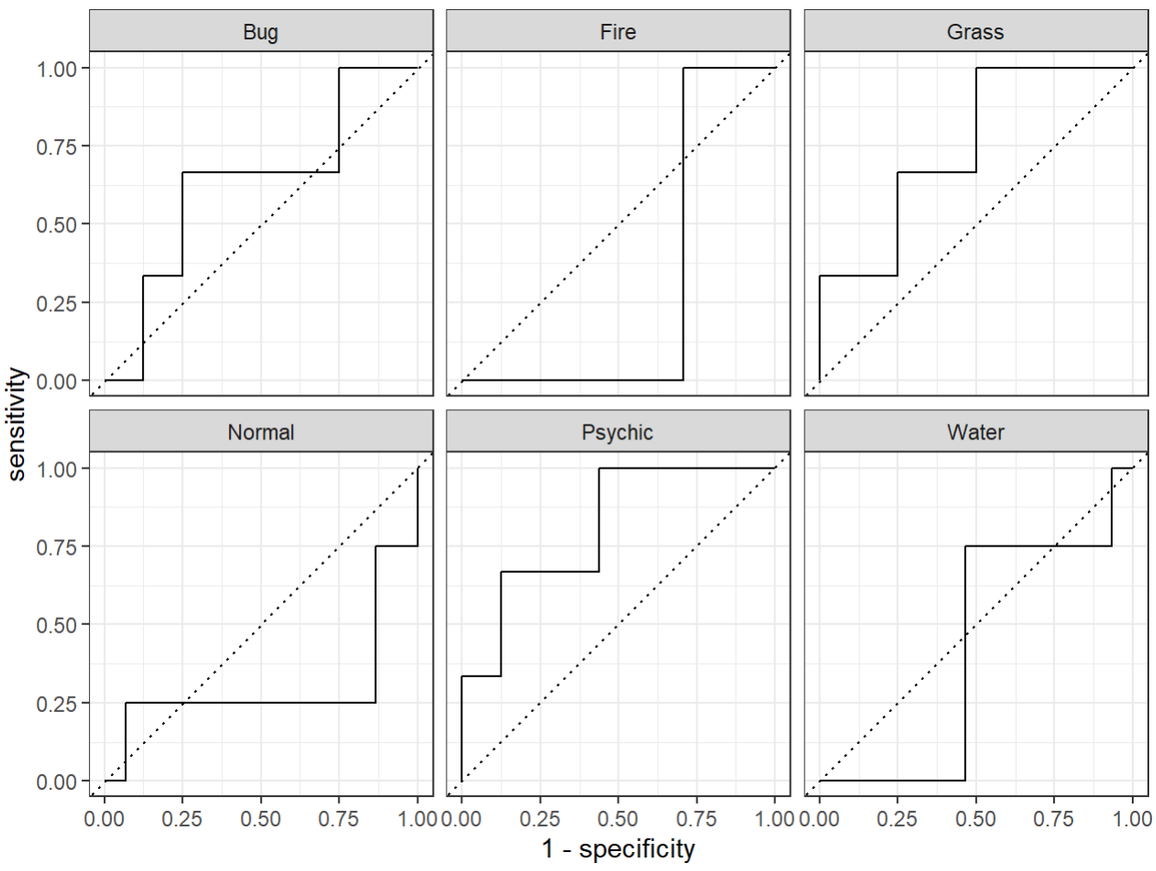
```
## # A tibble: 19 × 8
##    type_1  .pred_class .pred_Bug .pred_Fire .pred_Grass .pred_…¹ .pred…² .pred…³
##    <fct>   <fct>           <dbl>      <dbl>       <dbl>    <dbl>   <dbl>   <dbl>
##  1 Grass   Normal          0.135     0.108        0.170    0.240   0.145   0.201
##  2 Fire    Normal          0.128     0.105        0.165    0.229   0.178   0.195
##  3 Normal  Normal          0.150     0.115        0.181    0.223   0.118   0.213
##  4 Grass   Normal          0.138     0.111        0.175    0.237   0.132   0.207
##  5 Water   Normal          0.139     0.109        0.171    0.259   0.121   0.202
##  6 Water   Normal          0.136     0.109        0.171    0.207   0.176   0.202
##  7 Bug     Normal          0.144     0.111        0.174    0.227   0.140   0.205
##  8 Water   Normal          0.126     0.0994       0.156    0.322   0.112   0.185
##  9 Psychic Normal          0.123     0.0974       0.153    0.263   0.183   0.181
## 10 Grass   Water           0.151     0.119        0.188    0.203   0.117   0.222
## 11 Bug     Normal          0.152     0.115        0.180    0.233   0.108   0.213
## 12 Psychic Normal          0.132     0.102        0.161    0.277   0.138   0.190
## 13 Normal  Normal          0.136     0.107        0.168    0.281   0.111   0.198
## 14 Water   Normal          0.140     0.110        0.173    0.225   0.149   0.204
## 15 Normal  Water           0.153     0.117        0.183    0.213   0.117   0.216
## 16 Bug     Normal          0.128     0.102        0.160    0.226   0.194   0.190
## 17 Normal  Water           0.157     0.119        0.188    0.202   0.112   0.222
## 18 Psychic Normal          0.122     0.102        0.160    0.226   0.200   0.190
## 19 Fire    Normal          0.126     0.104        0.163    0.229   0.186   0.192
## # … with abbreviated variable names ¹.pred_Normal, ².pred_Psychic, ³.pred_Water
```

Exercise 8

```
fit_1 <- roc_auc1 %>%
  roc_auc(type_1, .pred_Bug:.pred_Water)
fit_1 #0.744
```

```
## # A tibble: 1 × 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 roc_auc hand_till      0.547
```

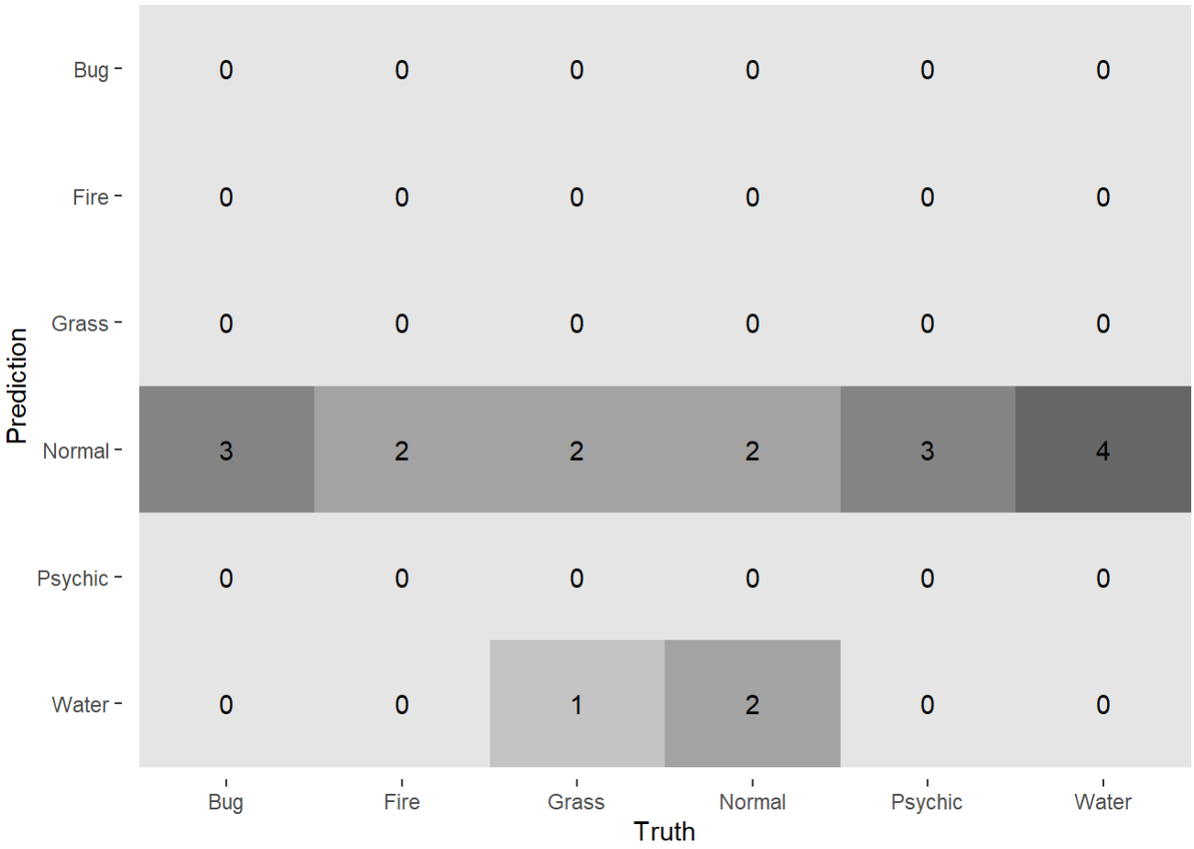
```
fit_2 <- roc_auc1 %>%
  roc_curve(type_1, .pred_Bug:.pred_Water)
ggplot2::autoplot(fit_2)
```



```
#roc_curve

fit_3 <- augment(ridge_final_fit, new_data = pokemon_test) %>%
  conf_mat(truth = type_1, estimate = .pred_class)

autoplot(fit_3, type = "heatmap")
```



I notice that Grass has a really high ROC curve, which is the best at predicting. Also, Normal, Psychic and Water also have a semi high ROC curve and are also best are predicting but not as high as Grass types. Fire and Bug aren't that good at predicting and don't have as high ROC curve. This may be due to not having as much values (observations) as Grass, Normal, Psychic and Water which causes Fire and Bug to not have as high ROC curves.