# 131-Homework6

Caleb Mazariegos

2022-05-23

## Exercise 1

```
#  Read in the data and set things up as in Homework 5:

# Use clean_names()
pokemon_codebook <- read.csv("/Users/calebmazariegos/Desktop/homework-5/Pokemon.csv")
pokemon_codebook <- clean_names(pokemon_codebook)
head(pokemon_codebook)
```

```
##   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
```

```
# Filter out the rarer Pokémon types
pokemon_filter <- pokemon_codebook %>%
  filter((type_1 == "Bug" | type_1 == "Fire" | type_1 == "Grass" | type_1 == "Normal" | type_1 == "Wate

head(pokemon_filter)
```

```
##   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
```

```
# converting type 1 and legendary in factors
pokemon_codebook$type_1 <- as.factor(pokemon_codebook$type_1)
pokemon_codebook$legendary <- as.factor(pokemon_codebook$legendary)
pokemon_codebook$generation <- as.factor(pokemon_codebook$generation)

pokemon_filter$type_1 <- as.factor(pokemon_filter$type_1)
pokemon_filter$legendary <- as.factor(pokemon_filter$legendary)
pokemon_filter$generation <- as.factor(pokemon_filter$generation)
```

```
# Do an initial split of the data; you can choose the percentage for splitting. Stratify on the outcome
set.seed(3465)

pokemon_split <- initial_split(pokemon_filter, prop = 0.70, stata = type_1)

pokemon_train <- training(pokemon_split)

pokemon_test <- testing(pokemon_split)
```

```
# Fold the training set using v-fold cross-validation, with v = 5. Stratify on the outcome variable.

set.seed(234)
pokemon_folds <- vfold_cv(pokemon_train, v=5)
pokemon_folds
```

```
## #  5-fold cross-validation
## # A tibble: 5 x 2
##   splits           id
##   <list>           <chr>
## 1 <split [256/64]> Fold1
## 2 <split [256/64]> Fold2
## 3 <split [256/64]> Fold3
## 4 <split [256/64]> Fold4
## 5 <split [256/64]> Fold5
```

```
# Set up a recipe to predict type_1 with legendary, generation, sp_atk, attack, speed, defense, hp, and
# dummy code legendary and generation; Center and scale all predictors.

pokemon_recipe <- recipe(type_1 ~ legendary + generation + sp_atk + attack + speed + defense + hp + sp_
  step_dummy(legendary) %>%
  step_dummy(generation) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
```

## Exercise 2

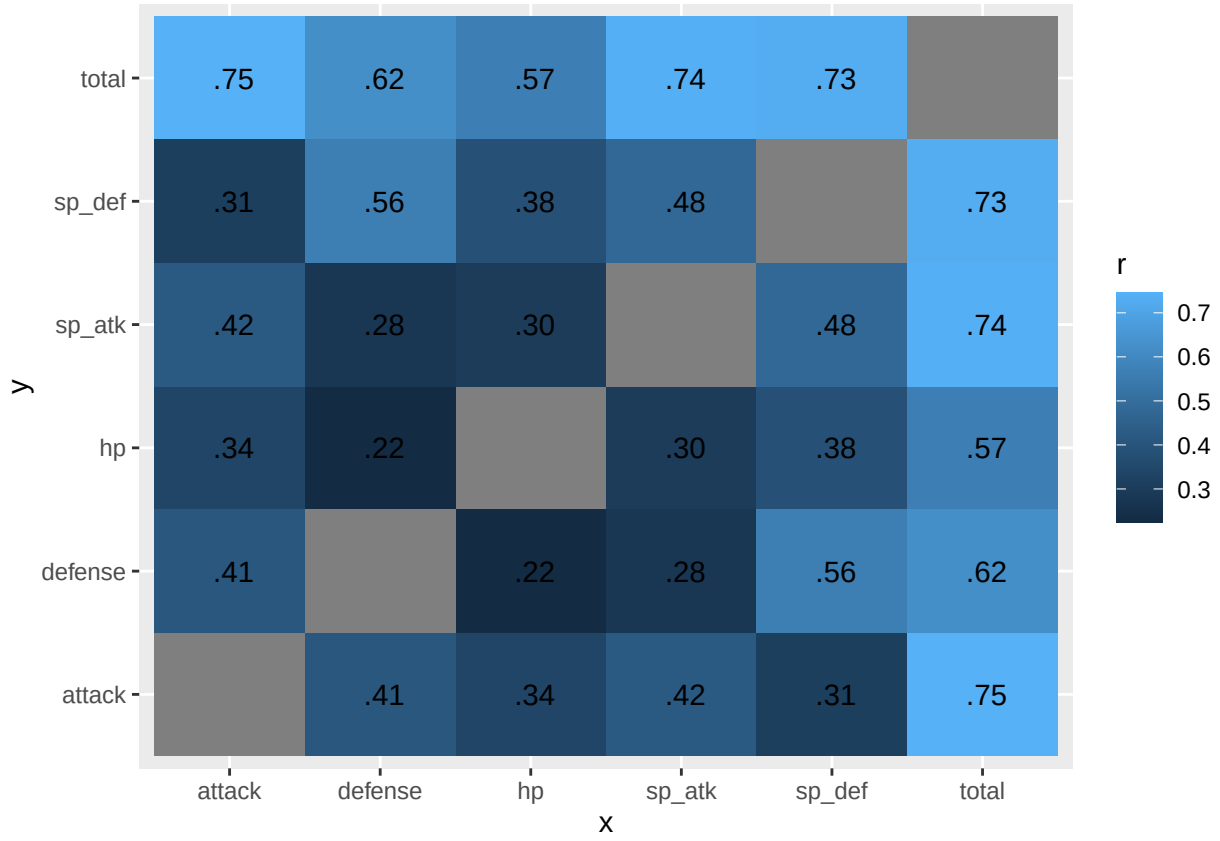
Create a correlation matrix of the training set, using the corrplot package. Note: You can choose how to handle the continuous variables for this plot; justify your decision(s).

What relationships, if any, do you notice? Do these relationships make sense to you?

```
cor_pokemon_train <- pokemon_train %>%
  select(total, hp, attack, defense, sp_atk, sp_def) %>%
  correlate()
```

```
##
## Correlation method: 'pearson'
## Missing treated using: 'pairwise.complete.obs'
```

```
cor_pokemon_train %>%
  stretch() %>%
  ggplot(aes(x,y, fill = r)) +
  geom_tile() +
  geom_text(aes(label = as.character(fashion(r))))
```




I decided to not include speed, because even though it is a numerical value, I do not consider it to be continuous. I see strong positive relationships between total and attack, total and sp_atk, total and sp_def. I see weak relationships between defense and sp_atk, and hp and defense. The rest of the relationships are moderate.

## Exercise 3

First, set up a decision tree model and workflow. Tune the cost_complexity hyperparameter. Use the same levels we used in Lab 7 – that is, range = c(-3, -1). Specify that the metric we want to optimize is roc_auc.

Print an autoplot() of the results. What do you observe? Does a single decision tree perform better with a smaller or larger complexity penalty?

tree_spec <- decision_tree() %>% set_engine("rpart")

class_tree_spec <- tree_spec %>% set_mode("classification")

class_tree_fit <- class_tree_spec %>% fit(type_1 ~., data = pokemon_train)