PSTAT 131 - Homework Assignment 2

Akshat Ataliwala (7924145)

April 10, 2022

Linear Regression

For this lab, we will be working with a data set from the UCI (University of California, Irvine) Machine Learning repository (see website here). The full data set consists of $4,177$ observations of abalone in Tasmania. (Fun fact: Tasmania supplies about 25% of the yearly world abalone harvest.)

The age of an abalone is typically determined by cutting the shell open and counting the number of rings with a microscope. The purpose of this data set is to determine whether abalone age (**number of rings + 1.5**) can be accurately predicted using other, easier-to-obtain information about the abalone.

The full abalone data set is located in the `\data` subdirectory. Read it into *R* using `read_csv()`. Take a moment to read through the codebook (`abalone_codebook.txt`) and familiarize yourself with the variable definitions.

Make sure you load the `tidyverse` and `tidymodels`!

```
library(tidymodels)
```

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

```
## v broom        0.7.12     v recipes      0.2.0
## v dials        0.1.0      v rsample      0.1.1
## v dplyr        1.0.8      v tibble       3.1.6
## v ggplot2      3.3.5      v tidyr        1.2.0
## v infer        1.0.0      v tune         0.2.0
## v modeldata    0.1.1      v workflows    0.2.6
## v parsnip      0.2.1      v workflowsets 0.2.1
## v purrr        0.3.4      v yardstick    0.0.9
```

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

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
## v readr   2.1.2     v forcats 0.5.1
## v stringr 1.4.0
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x readr::col_factor() masks scales::col_factor()
## x purrr::discard()    masks scales::discard()
## x dplyr::filter()     masks stats::filter()
## x stringr::fixed()    masks recipes::fixed()
## x dplyr::lag()        masks stats::lag()
## x readr::spec()       masks yardstick::spec()
```

```
library(ggplot2)
data <- read_csv("data/abalone.csv")
```

```
## Rows: 4177 Columns: 9
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): type
## dbl (8): longest_shell, diameter, height, whole_weight, shucked_weight, visc...
##
## 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.
```

```
data %>% head(5)
```

```
## # A tibble: 5 x 9
##   type  longest_shell diameter height whole_weight shucked_weight viscera_weight
##   <chr>         <dbl>    <dbl>  <dbl>        <dbl>          <dbl>          <dbl>
## 1 M             0.455    0.365  0.095        0.514         0.224          0.101
## 2 M             0.35     0.265  0.09         0.226         0.0995         0.0485
## 3 F             0.53     0.42   0.135        0.677         0.256          0.142
## 4 M             0.44     0.365  0.125        0.516         0.216          0.114
## 5 I             0.33     0.255  0.08         0.205         0.0895         0.0395
## # ... with 2 more variables: shell_weight <dbl>, rings <dbl>
```

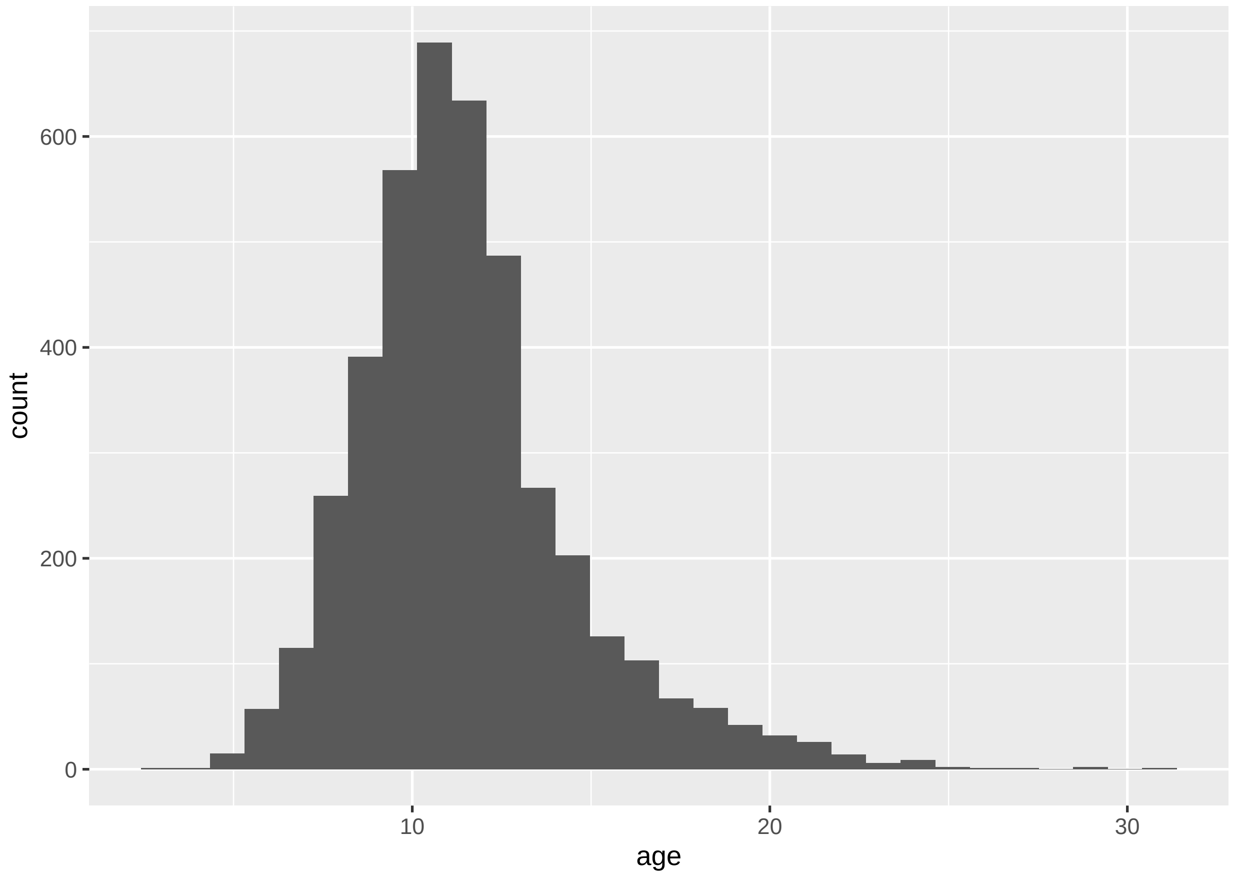
1. Your goal is to predict abalone age, which is calculated as the number of rings plus 1.5. Notice there currently is no age variable in the data set. Add age to the data set. Assess and describe the distribution of age.

```
# Adding age variable to the dataset
data = data %>% mutate(age = rings + 1.5)
data %>% head(5)
```

```
## # A tibble: 5 x 10
##   type  longest_shell diameter height whole_weight shucked_weight viscera_weight
##   <chr>         <dbl>    <dbl>  <dbl>        <dbl>          <dbl>          <dbl>
## 1 M             0.455    0.365  0.095        0.514         0.224          0.101
## 2 M             0.35     0.265  0.09         0.226         0.0995         0.0485
## 3 F             0.53     0.42   0.135        0.677         0.256          0.142
## 4 M             0.44     0.365  0.125        0.516         0.216          0.114
## 5 I             0.33     0.255  0.08         0.205         0.0895         0.0395
## # ... with 3 more variables: shell_weight <dbl>, rings <dbl>, age <dbl>
```

```
age_hist <- ggplot(data, aes(x = age)) + geom_histogram()
age_hist
```

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



The distribution of age is somewhat normal, with the majority of values being around 10 years, and both sides tapering off as they diverge from the average. Age is definitely more right tailed than left tailed, however, as there are some extremely old outliers in the 30's.

2. Split the abalone data into a training set and a testing set. Use stratified sampling. You should decide on appropriate percentages for splitting the data. ***Remember that you'll need to set a seed at the beginning of the document to reproduce your results.***

```
set.seed(3478)

data_split <- initial_split(data, prop = 0.8, strata = age)

train <- training(data_split)
test <- testing(data_split)
```

3. Using the training data, create a recipe predicting the outcome variable, `age`, with all other predictor variables. Note that you should not include `rings` to predict `age`. Explain why you shouldn't use `rings` to predict `age`.

Steps for your recipe:

1. dummy code any categorical predictors
2. create interactions between
   - `type` and `shucked_weight`,
   - `longest_shell` and `diameter`,
   - `shucked_weight` and `shell_weight`
3. center all predictors, and
4. scale all predictors.

You'll need to investigate the `tidymodels` documentation to find the appropriate step functions to use.

```
abalone <- recipe(age ~ type + longest_shell + diameter + height +
                    whole_weight + shucked_weight + viscera_weight +
                    shell_weight, data = train) %>%
  step_dummy(type, levels = 3) %>%
  step_interact(terms = ~ type_M:shucked_weight) %>%
  step_interact(terms = ~ type_I:shucked_weight) %>%
  step_interact(terms = ~ longest_shell:diameter) %>%
  step_interact(terms = ~ shucked_weight:shell_weight) %>%
  step_normalize()
```

We shouldn't use rings to predict age because age is essentially just a transformation of the the rings variable, so our model would be incredibly accurate at predicting age, which defeats the whole purpose.

4. Create and store a linear regression object using the `"lm"` engine.

```
lm_model <- linear_reg() %>%
  set_engine("lm")
```

5.

1. set up an empty workflow,
2. add the model you created in Question 4, and
3. add the recipe that you created in Question 3.

```
lm_wflow <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(abalone)
```

6. Use your `fit()` object to predict the age of a hypothetical female abalone with longest_shell = 0.50, diameter = 0.10, height = 0.30, whole_weight = 4, shucked_weight = 1, viscera_weight = 2, shell_weight = 1.

```
lm_fit <- fit(lm_wflow, train)
```

```
hfa_predict <- data.frame(type = 'F', longest_shell = 0.5, diameter = 0.1,
                          height = 0.3, whole_weight = 4, shucked_weight = 1,
                          viscera_weight = 2, shell_weight = 1)

predict(lm_fit, new_data = hfa_predict)
```

```
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  24.7
```

7. Now you want to assess your model's performance. To do this, use the `yardstick` package:

1. Create a metric set that includes R^2, RMSE (root mean squared error), and MAE (mean absolute error).
2. Use `predict()` and `bind_cols()` to create a tibble of your model's predicted values from the **training data** along with the actual observed ages (these are needed to assess your model's performance).
3. Finally, apply your metric set to the tibble, report the results, and interpret the R^2 value.

```
abalone_train_res <- predict(lm_fit, new_data = train %>% select(-age))

abalone_train_res <- bind_cols(abalone_train_res, train %>% select(age))

abalone_metrics <- metric_set(rsq, rmse, mae)
abalone_metrics(abalone_train_res, truth = age, estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.562
## 2 rmse    standard       2.13
## 3 mae     standard       1.53
```

The resulting R^2 from my model was 0.562. This essentially means that our predictors could only capture 56.2% of the data, which is pretty poor. I would say that in this case it is not wise to use a linear regression model and would instead try another model for better results.