Homework 2

PSTAT 131/231

Contents


```
knitr::opts_chunk$set(echo = TRUE, message = FALSE,
                      warning = FALSE)
```

Linear Regression and KNN

For this assignment, 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.

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(tidyverse)
library(tidymodels)
abalone_data = read_csv('abalone.csv')
abalone_data
```

```
## # A tibble: 4,177 x 9
##    type  longest_shell diameter height whole_weight shucked_weight
##    <chr>         <dbl>    <dbl>  <dbl>        <dbl>          <dbl>
##  1 M             0.455    0.365  0.095        0.514         0.224
##  2 M             0.35     0.265  0.09         0.226         0.0995
##  3 F             0.53     0.42   0.135        0.677         0.256
##  4 M             0.44     0.365  0.125        0.516         0.216
##  5 I             0.33     0.255  0.08         0.205         0.0895
##  6 I             0.425    0.3    0.095        0.352         0.141
##  7 F             0.53     0.415  0.15         0.778         0.237
##  8 F             0.545    0.425  0.125        0.768         0.294
##  9 M             0.475    0.37   0.125        0.509         0.216
## 10 F             0.55     0.44   0.15         0.894         0.314
## # i 4,167 more rows
## # i 3 more variables: viscera_weight <dbl>, shell_weight <dbl>, rings <dbl>
```

Question 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.

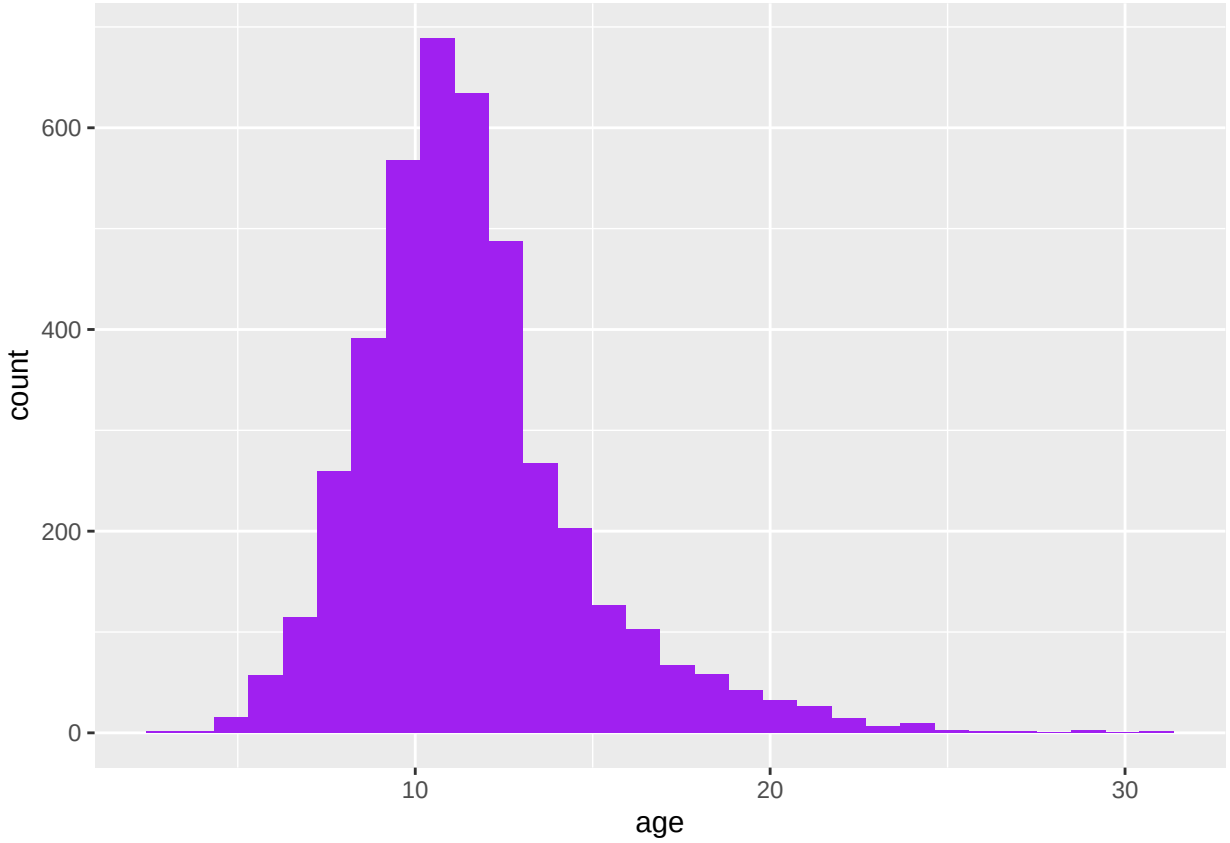
Assess and describe the distribution of `age`.

```
abalone_data <- abalone_data %>%
  mutate(age = rings + 1.5)

summary(abalone_data$age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    2.50    9.50   10.50   11.43   12.50   30.50
```

```
ggplot(abalone_data, aes(x=age)) +
geom_histogram(fill='purple')
```



Our mean abalone age is 11.43, and our median is 10.50. From the histogram, we see that it is rare that an abalone is above 20 years old, as well as below 5 years old. The most frequent ages are between 5-15 years, with a large majority of them being about 10 years old.

Question 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(2232)
split_abalone = initial_split(abalone_data, strata = age, prop=0.75)
abalone_train = training(split_abalone)
abalone_test = testing(split_abalone)
split_abalone
```

```
## <Training/Testing/Total>
## <3131/1046/4177>
```

Question 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 = recipe(age ~ ., data = abalone_train) %>%
  step_rm(rings) %>%
  step_dummy(type) %>%
  step_interact(~ starts_with('type'):shucked_weight + longest_shell:diameter + shucked_weight:shell_wei
  step_center(all_predictors()) %>%
  step_scale(all_predictors())
abalone_recipe %>%    # to see the contents of our recipe.
prep() %>%
bake(new_data = abalone_train)
```

```
## # A tibble: 3,131 x 14
##    longest_shell diameter height whole_weight shucked_weight viscera_weight
##            <dbl>    <dbl>  <dbl>        <dbl>          <dbl>          <dbl>
##  1        -1.42    -1.41  -1.16        -1.22          -1.16          -1.19
##  2        -1.58    -1.51  -1.39        -1.26          -1.20          -1.27
##  3        -1.38    -1.26  -1.27        -1.09          -1.18          -1.27
##  4        -0.471   -0.512 -0.803       -0.700         -0.581         -0.504
##  5        -2.33    -2.31  -2.21        -1.54          -1.47          -1.42
##  6        -2.62    -2.56  -1.98        -1.59          -1.49          -1.50
##  7        -2.57    -2.56  -2.10        -1.59          -1.53          -1.52
```

```
##  8        -1.09    -1.11  -1.04        -1.26          -1.21          -1.22
##  9        -0.512   -0.312 -0.450       -0.738         -0.805         -0.631
## 10        -1.63    -1.61  -1.63        -1.35          -1.27          -1.40
## # i 3,121 more rows
## # i 8 more variables: shell_weight <dbl>, age <dbl>, type_I <dbl>,
## #   type_M <dbl>, type_I_x_shucked_weight <dbl>, type_M_x_shucked_weight <dbl>,
## #   longest_shell_x_diameter <dbl>, shucked_weight_x_shell_weight <dbl>
```

You shouldn't use rings to predict age because their correlation with each other is exactly 1, since we created the variable age, based on rings (age was derived from rings). When rings increases, age will increase by the same amount each time. By including rings as a predictor, you would be using data that wouldn't be present in a real prediction scenario and it also means that there is data leakage. We would need to predict age without the use of rings.

Question 4

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

```
lm_model = linear_reg() %>%
  set_engine("lm")
```

Question 5

Create and store a KNN object using the `"kknn"` engine. Specify `k = 7`.

```
library(kknn)
knn_model = nearest_neighbor(neighbors = 7) %>%
  set_engine("kknn") %>%
  set_mode("regression")
```

Question 6

Now, for each of these models (linear regression and KNN):

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

Note that you should be setting up two separate workflows.

Fit both models to the training set.

```
lm_wflow = workflow() %>%
  add_model(lm_model) %>%
  add_recipe(abalone_recipe)

knn_wflow = workflow() %>%
  add_model(knn_model) %>%
  add_recipe(abalone_recipe)

 lm_fit  = fit(lm_wflow,  abalone_train)
 knn_fit = fit(knn_wflow, abalone_train)
```

Question 7

Use your linear regression `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, and shell_weight = 1.

```
lm_fit %>%  # makes our lm_fit readable (we can see information about all our predictors)
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 14 x 5
##    term                          estimate std.error statistic  p.value
##    <chr>                            <dbl>     <dbl>     <dbl>    <dbl>
##  1 (Intercept)                     11.4      0.0386   296.    0
##  2 longest_shell                    0.678    0.294      2.30  2.13e- 2
##  3 diameter                         2.11     0.320      6.58  5.35e-11
##  4 height                           0.229    0.0725     3.16  1.59e- 3
##  5 whole_weight                     4.92     0.401     12.3   6.82e-34
##  6 shucked_weight                  -4.23     0.258    -16.4   7.46e-58
##  7 viscera_weight                  -0.896    0.164     -5.47  4.86e- 8
##  8 shell_weight                     1.60     0.220      7.30  3.72e-13
##  9 type_I                          -0.948    0.121     -7.86  5.13e-15
## 10 type_M                          -0.327    0.107     -3.06  2.27e- 3
## 11 type_I_x_shucked_weight          0.477    0.0908     5.26  1.55e- 7
## 12 type_M_x_shucked_weight          0.334    0.112      2.98  2.94e- 3
## 13 longest_shell_x_diameter        -2.99     0.411     -7.29  3.93e-13
## 14 shucked_weight_x_shell_weight   -0.101    0.207     -0.487 6.26e- 1
```

```
new_data = data.frame(type='F', longest_shell = 0.50, diameter = 0.10, height = 0.30, whole_weight = 4,

new_data
```

```
##   type longest_shell diameter height whole_weight shucked_weight viscera_weight
## 1    F           0.5      0.1    0.3            4              1              2
##   shell_weight rings
## 1            1    NA
```

```
predicted_age = predict(lm_fit, new_data)
predicted_age
```

```
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  24.3
```

Predicted age for this female abalone is about 24.3 years old.

Question 8

Now you want to assess your models' 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 **testing 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.

Repeat these steps once for the linear regression model and for the KNN model.

```
abalone_metrics = metric_set(rsq, rmse, mae)

abalone_test_lm = predict(lm_fit, new_data = abalone_test %>% select(-age)) # use all predictors in rec
abalone_test_lm = bind_cols(abalone_test_lm, abalone_test %>% select(age)) # The first column is our mo
abalone_test_lm
```

```
## # A tibble: 1,046 x 2
##    .pred   age
##    <dbl> <dbl>
##  1  9.41   9.5
##  2 15.5   21.5
##  3 10.3    8.5
##  4 10.1   10.5
##  5 12.8   13.5
##  6 11.0    9.5
##  7 12.4   10.5
##  8 10.7   11.5
##  9 10.8    9.5
## 10  6.71   5.5
## # i 1,036 more rows
```

```
abalone_metrics_lm = abalone_metrics(abalone_test_lm, truth = age, estimate = .pred)
abalone_metrics_lm
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.529
## 2 rmse    standard       2.13
## 3 mae     standard       1.52
```

```
abalone_test_knn = predict(knn_fit, new_data = abalone_test %>% select(-age))
abalone_test_knn = bind_cols(abalone_test_knn, abalone_test %>% select(age)) # what data to display.
# We want our predictions (abalone_test_knn), and our age column from the dataset so we combine them.
abalone_test_knn
```

```
## # A tibble: 1,046 x 2
##    .pred   age
##    <dbl> <dbl>
##  1  8.80   9.5
##  2 14.8   21.5
##  3 10.5    8.5
##  4  9.10  10.5
##  5 12.3   13.5
```

```
##  6 10.1    9.5
##  7 13.0   10.5
##  8  8.96  11.5
##  9 11.0    9.5
## 10  6.28   5.5
## # i 1,036 more rows
```

```
abalone_metrics_knn = abalone_metrics(abalone_test_knn, truth = age, estimate = .pred)
abalone_metrics_knn
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.502
## 2 rmse    standard       2.21
## 3 mae     standard       1.56
```

For our linear regression model, we had an R^2 value of 0.529, a RMSE of 2.132, and a MAE of 1.521. For our KNN model, we had an R^2 value of 0.502, a RMSE of 2.206, and a MAE of 1.555.

The R^2 value tells us how much variability observed in our outcome variable, "age", in our testing data is explained by our model. So in this case, 52.9% and 50.2% of the variability observed is explained by our linear regression and KNN models, respectively.

Question 9

Which model performed better on the testing data? Explain why you think this might be. Are you surprised by any of your results? Why or why not?

On the testing data, the Linear Regression Model performed better. This could likely be because at least one of our predictors has a linear relationship with our outcome, "age". Our model will then focus on using these predictors to predict the outcome. As opposed to the KNN model where we are using the closest neighbors to predict the outcome, some predictors may not have a linear relationship and as a result, affecting our prediction since our model isn't able to use those linear predictors as a main source for modeling. I am not very surprised about the linear model doing better on all three metrics, but the KNN model did quite well, as the RMSE and MAE were quite close to the linear model which did surprise me. I am also surprised by the low R^2 values for both models as I am used to dealing with R^2 values usually above 0.7.