Homework 2

PSTAT 131/231

Contents

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. (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`!

Question 1

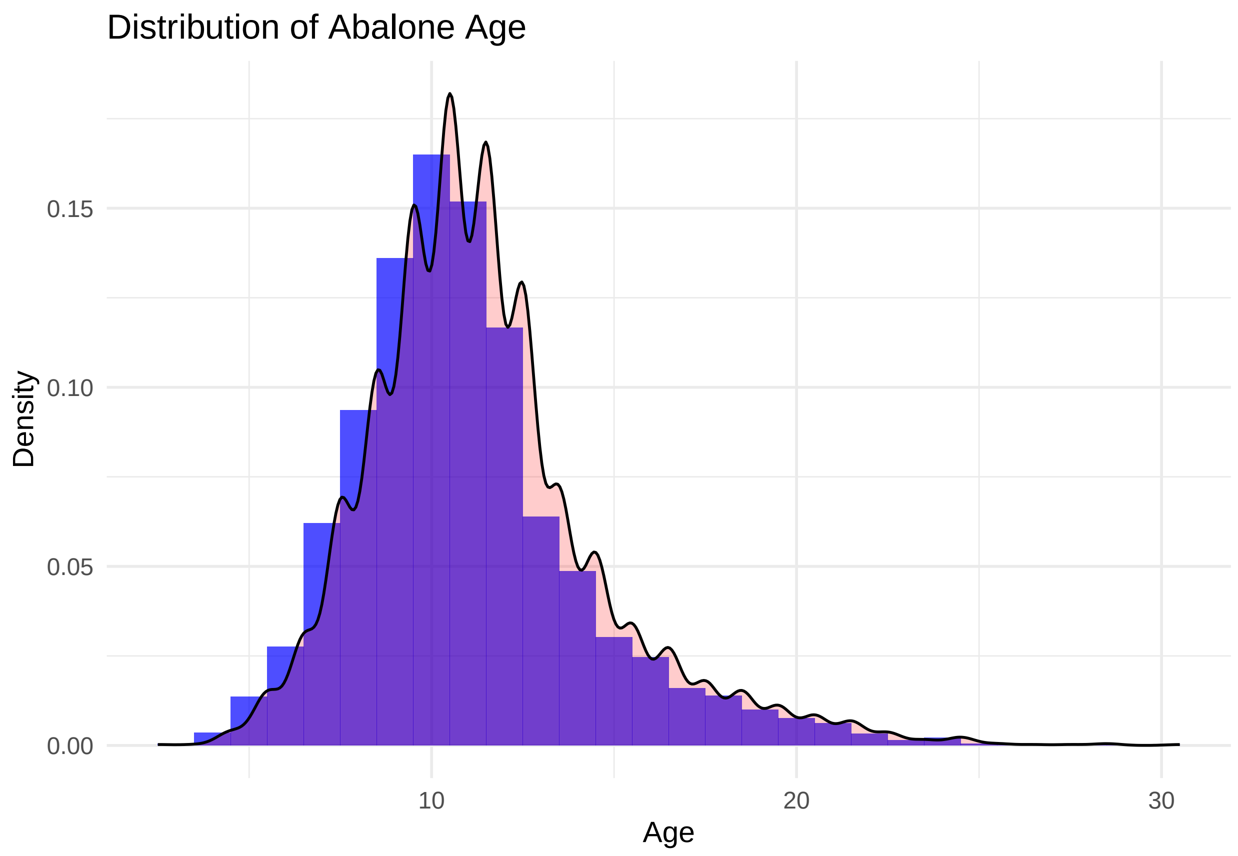
Your goal is to predict abalone age, which is calculated as the number of rings plus 1.5. Notice there currently is no `age` variable in the data set. Add `age` to the data set.

```r
# Read the dataset.
abalone_data <- read_csv("abalone.csv")

# Calculate the age (which is calculated as the number of rings + 1.5)
abalone_data <- abalone_data %>%
  mutate(age = rings + 1.5) # Also create new variable for 'age'

# View the first few rows to check the new 'age' column
head(abalone_data)
```

```
## # A tibble: 6 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
## 6 I              0.425    0.3    0.095        0.352        0.141          0.0775
## # i 3 more variables: shell_weight <dbl>, rings <dbl>, age <dbl>
```

Assess and describe the distribution of `age`.

```
# Create a histogram to assess and describe the distribution of age
ggplot(abalone_data, aes(x = age)) +
  geom_histogram(aes(y = ..density..), binwidth = 1, alpha = 0.7, fill = "blue") +
  geom_density(alpha = 0.2, fill = "red") +
  ggtitle("Distribution of Abalone Age") +
  xlab("Age") +
  ylab("Density") +
  theme_minimal()
```



```
## From the histogram, we can observe and conclude that:
#1. The age distribution is somewhat right-skewed, meaning that there are more
#younger abalones (yonger than 13) than older ones in the dataset.
#2. Most abalones have an age between approximately 8 and 12
```

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.

```r
# Set a seed.
set.seed(123)

# Perform stratified sampling to split the data into training and testing sets
data_split <- initial_split(abalone_data, prop = 0.8, strata = "type")
train_data <- training(data_split)
test_data <- testing(data_split)

# View and check the first several rows of the training and testing sets
head(train_data)
```

```
## # A tibble: 6 x 10
##   type  longest_shell diameter height whole_weight shucked_weight viscera_weight
##   <chr>         <dbl>    <dbl>  <dbl>        <dbl>          <dbl>          <dbl>
## 1 F             0.53     0.415  0.15         0.778          0.237          0.142
## 2 F             0.545    0.425  0.125        0.768          0.294          0.150
## 3 F             0.55     0.44   0.15         0.894          0.314          0.151
## 4 F             0.525    0.38   0.14         0.606          0.194          0.148
## 5 F             0.535    0.405  0.145        0.684          0.272          0.171
## 6 F             0.44     0.34   0.1          0.451          0.188          0.087
## # i 3 more variables: shell_weight <dbl>, rings <dbl>, age <dbl>
```

```r
head(test_data)
```

```
## # A tibble: 6 x 10
##   type  longest_shell diameter height whole_weight shucked_weight viscera_weight
##   <chr>         <dbl>    <dbl>  <dbl>        <dbl>          <dbl>          <dbl>
## 1 F             0.53     0.42   0.135        0.677          0.256         0.142
## 2 M             0.49     0.38   0.135        0.542          0.218         0.095
## 3 F             0.47     0.355  0.1          0.476          0.168         0.0805
## 4 I             0.355    0.28   0.085        0.290          0.095         0.0395
## 5 M             0.365    0.295  0.08         0.256          0.097         0.043
## 6 M             0.45     0.32   0.1          0.381          0.170         0.075
## # i 3 more variables: shell_weight <dbl>, rings <dbl>, age <dbl>
```

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`.

```r
# The reason why we should not include 'rings' to pridict 'age' is that
#'rings' is essentially used to calculate the age. And this would give
#unrealistically optimistic performance metrics, as the model would have
#access to information it shouldn't have during the training phase. The issue
#is also know as 'data leakage', which means using a variable in the model
#that directly or indirectly includes the information we're trying to predict.
```

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.

```
# Create the recipe with manual interactions using step_mutate()
abalone_recipe <- recipe(age ~ ., data = train_data) %>%
  update_role(rings, new_role = "ID variable") %>%
  step_dummy(type, one_hot = TRUE) %>%
  step_mutate(interaction_1 = type_M * shucked_weight,
              interaction_2 = longest_shell * diameter,
              interaction_3 = shucked_weight * shell_weight) %>%
  step_center(all_predictors()) %>%
  step_scale(all_predictors())

# Print the recipe to check it
abalone_recipe
```

Question 4

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

```
# Create a linear regression model specification
linear_reg_spec <- linear_reg() %>%
  set_engine("lm")

# Print the linear regression specification to check it
linear_reg_spec
```

```
## Linear Regression Model Specification (regression)
##
## Computational engine: lm
```

Question 5

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

```
# Create a KNN model specification with mode explicitly set to "regression"
knn_spec <- nearest_neighbor(neighbors = 7, mode = "regression") %>%
  set_engine("kknn")

# Print the KNN model specification to check it
knn_spec
```

```
## K-Nearest Neighbor Model Specification (regression)
##
## Main Arguments:
##   neighbors = 7
##
## Computational engine: kknn
```

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.

```
# Create a workflow for the linear regression model
linear_reg_wf <- workflow() %>%
  add_model(linear_reg_spec) %>%
  add_recipe(abalone_recipe)

# Fit the linear regression model to the training data
linear_reg_fit <- linear_reg_wf %>%
  fit(data = train_data)

# Create a workflow for the KNN model
knn_wf <- workflow() %>%
  add_model(knn_spec) %>%
  add_recipe(abalone_recipe)

# Fit the KNN model to the training data
knn_fit <- knn_wf %>%
  fit(data = train_data)

# Print the fitted models to check them
linear_reg_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 4 Recipe Steps
##
## * step_dummy()
## * step_mutate()
## * step_center()
## * step_scale()
##
## -- Model -----------------------------------------------------------------------
```

```
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
##    (Intercept)  longest_shell       diameter         height   whole_weight
##       11.43623        0.60430        2.15167        0.64423        4.71620
## shucked_weight  viscera_weight   shell_weight         type_F         type_I
##       -3.70711       -0.99759        1.72939        0.03056       -0.32593
##         type_M  interaction_1  interaction_2  interaction_3
##             NA        0.04967       -3.07420       -0.47508
```

```
knn_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: nearest_neighbor()
##
## -- Preprocessor ------------------------------------------------------------------
## 4 Recipe Steps
##
## * step_dummy()
## * step_mutate()
## * step_center()
## * step_scale()
##
## -- Model -------------------------------------------------------------------------
##
## Call:
## kknn::train.kknn(formula = ..y ~ ., data = data, ks = min_rows(7,     data, 5))
##
## Type of response variable: continuous
## minimal mean absolute error: 1.662025
## Minimal mean squared error: 5.540684
## Best kernel: optimal
## Best k: 7
```

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.

```
# Create a data frame for the corrected hypothetical female abalone
# Also add a placeholder for 'rings' to avoid errors
hypothetical_abalone_corrected <- tibble(
  type = "F",
  longest_shell = 0.50,
  diameter = 0.10,
  height = 0.30,
  whole_weight = 4,
  shucked_weight = 1,
  viscera_weight = 2,
```

```r
  shell_weight = 1,
  rings = NA  # Placeholder
)

# Use the predict function
predicted_age_corrected <- predict(linear_reg_fit, new_data =
                                     hypothetical_abalone_corrected)

# Print the predicted age
predicted_age_corrected
```

```
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  22.4
```

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).

```r
# Create a metric set as required
metric_set <- metric_set(rsq, rmse, mae)
```

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).

```r
# Predictions for the linear regression model
linear_reg_preds <- predict(linear_reg_fit, new_data = test_data) %>%as_tibble()

linear_reg_preds <- bind_cols(linear_reg_preds, actual_age = test_data$age)

# Predictions for the KNN model
knn_preds <- predict(knn_fit, new_data = test_data) %>% as_tibble()

knn_preds <- bind_cols(knn_preds, actual_age = test_data$age)
```

3. Finally, apply your metric set to the tibble, report the results, and interpret the *R^2* value.

```r
# Apply metric set to the tibble for the linear regression model
linear_reg_metrics <- linear_reg_preds %>%
  metric_set(.pred, truth = actual_age)

# Print and interpret the metrics for the linear regression model
print("Metrics for Linear Regression Model")
```

```
## [1] "Metrics for Linear Regression Model"
```

```
print(linear_reg_metrics)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.518
## 2 rmse    standard       2.23
## 3 mae     standard       1.61
```

```
# Apply metric set to the tibble for the KNN model
knn_metrics <- knn_preds %>%
  metric_set(.pred, truth = actual_age)

# Print and interpret the metrics for the KNN model
print("Metrics for KNN Model")
```

```
## [1] "Metrics for KNN Model"
```

```
print(knn_metrics)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rsq     standard       0.471
## 2 rmse    standard       2.33
## 3 mae     standard       1.63
```

```
## R^2 reflects the proportion of the variance in the 'age' variable can be
#explained by the model.

## From the results, in Linear Regression Model, we have R^2=0.52, which means
#approximately 52% of the variance in the 'age' variable can be explained by the
#model. And in KNN model, We have R^2=0.47, which means approximately 47% of the
#variance in the 'age' variable can be explained by the model.
```

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?

```
## From Question 8, in Linear Regression Model, we have R^2=0.52, which means
#approximately 52% of the variance in the 'age' variable can be explained by the
#model. RMSE=2.23, which means The model's predictions are, on average, about
#2.232 units away from the actual values in terms of the root mean square error.
#MAE=1.61, which means the model's predictions are, on average, about 1.61 units
#away from the actual values in terms of the mean absolute error.

## And in KNN model, We have R^2=0.47, which means approximately 47% of the
#variance in the 'age' variable can be explained by the model. RMSE=2.33, which
#means The model's predictions are, on average, about 2.33 units away from the
```

```
#actual values in terms of the root mean square error. And MAE=1.63, which means
#the model's predictions are, on average, about 1.61 units away from the actual
#values in terms of the mean absolute error.

## Conlucsion: Both models have similar performance, but the linear regression
#model is slightly better, because of smaller/lower R^2, RMSE, and MAE.

## Linear regression models assume a linear relationship between the independent
#and dependent variables, while KNN does not make such an assumption. If the true
#underlying relationship is closer to being linear, then linear regression would
#naturally perform better, which seems to be the case here.

## It is not surprised to see that the linear regression model perform better,
#Given that it incorporates assumptions about the relationship between variables.
#But it is surprising that the performances are really close between two models,
#which shows that for this specific dataset/problem, model choise is not crucial.
```