PSTAT131HW02

Yifei Zhang

2022-04-10

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.

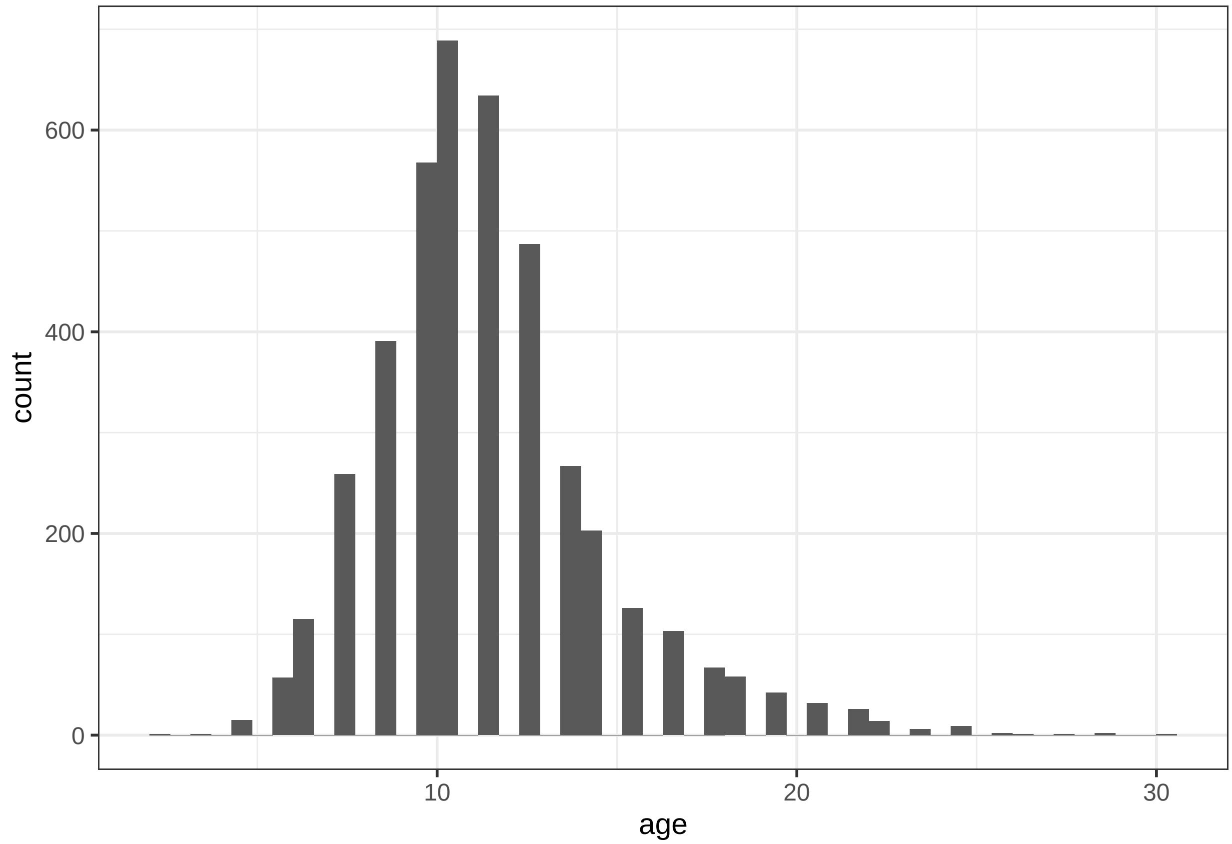
```
abalone <- read.csv("abalone.csv")

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

Assess and describe the distribution of `age`.

As we access the distribution of the new variable age, we can see it looks relatively normal, but is skewed a bit to the right, centering around 10.

```
new_abalone %>%
  ggplot(aes(x = age)) +
  geom_histogram(bins = 50) +
  theme_bw()
```



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(2728)
new_abalone <- new_abalone %>%
  mutate(num_type = case_when(type == "M" ~ 0,
                              type == "F" ~ 1,
                              type == "I" ~ 2))
#I need to put the above lines of code there so I dont run
#into problems later when knitting, its a logical error if
#I dont put them there. Because splitting the training and
#test data before dummy coding type will result in them not
#having the new variable.

abalone_split <- initial_split(new_abalone, prop = 0.80,
                               strata = age)
abalone_train <- training(abalone_split)
abalone_test <- testing(abalone_split)
```

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`.

I should not use rings to predict age, because we have used rings to calculate age in step 1, and we wont be able to see how other predictors can predict by having rings as one of the predictors, since we already have age = rings + 1.5 function.

Steps for your recipe:

1. dummy code any categorical predictors

new_abalone$gender_m < -ifelse(new_abalone$type == “M”, 1, 0) new_abalone$gender_f < -ifelse(new_abalone$type == “F”, 1, 0) new_abalone$gender_i < -ifelse(new_abalone$type == “I”, 1, 0) this don't seem to work with the latter steps

```r
new_abalone <- new_abalone %>%
  mutate(num_type = case_when(type == "M" ~ 0,
                              type == "F" ~ 1,
                              type == "I" ~ 2))
# or use step_dummy()

sapply(new_abalone, class)
```

```
##           type  longest_shell       diameter         height   whole_weight
##    "character"      "numeric"      "numeric"      "numeric"      "numeric"
## shucked_weight viscera_weight   shell_weight          rings            age
##      "numeric"      "numeric"      "numeric"      "integer"      "numeric"
##       num_type
##      "numeric"
```

2. create interactions between

    - `type` and `shucked_weight`,
    - `longest_shell` and `diameter`,
    - `shucked_weight` and `shell_weight`

```r
abalone_recipe <- recipe(age ~ longest_shell + diameter + height +
                           whole_weight + shucked_weight +
                           viscera_weight + shell_weight + num_type,
                         data = new_abalone)
interact1 <- abalone_recipe %>%
  step_interact(terms = ~ num_type : shucked_weight)

interact2 <- abalone_recipe %>%
  step_interact(terms = ~ longest_shell : diameter, data)

interact3 <- abalone_recipe %>%
  step_interact(terms = ~ shell_weight : shucked_weight)
```

I think my seed number is too large, so in order to keep the pdf short I am not going to show the result for the next two parts. I tried to show the result output, but that made my knitted pdf more than a thousand pages long!!! Way to scary. If you want to check out the result you should still be able to see it when you run my rmarkdown file, the results are just not knitted into the pdf file.

3. center all predictors, and

```
# or
center_aba <- abalone_recipe %>%
  step_center(longest_shell, diameter, height, whole_weight,
              shucked_weight, viscera_weight, shell_weight, num_type
              )
center_aba
```

or we can use

scale(new_abalone$longest_shell, center = TRUE, scale = FALSE)

scale(new_abalone$diameter, center = TRUE, scale = FALSE)

scale(new_abalone$height, center = TRUE, scale = FALSE)

scale(new_abalone$whole_weight, center = TRUE, scale = FALSE)

scale(new_abalone$shucked_weight, center = TRUE, scale = FALSE)

scale(new_abalone$viscera_weight, center = TRUE, scale = FALSE)

scale(new_abalone$shell_weight, center = TRUE, scale = FALSE)

scale(new_abalone$num_type, center = TRUE, scale = FALSE)

4. scale all predictors.

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

```
scale_aba <- abalone_recipe %>%
  step_scale(longest_shell, diameter, height, whole_weight,
              shucked_weight, viscera_weight, shell_weight, num_type
              )
scale_aba
```

or we can use

scale(new_abalone$longest_shell)

scale(new_abalone$diameter)

scale(new_abalone$height)

scale(new_abalone$whole_weight)

scale(new_abalone$shucked_weight)

scale(new_abalone$viscera_weight)

scale(new_abalone$shell_weight)

scale(new_abalone$num_type)

Creating recipe

```r
abalone_recipe <- recipe(age ~ longest_shell + diameter + height +
                           whole_weight + shucked_weight +
                           viscera_weight + shell_weight + num_type,
                         data = new_abalone)
abalone_recipe
```

```
## Data Recipe
##
## Inputs:
##
##       role #variables
##    outcome          1
##  predictor          8
```

Question 4

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

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

# I ended up using what we used in the lab, but what is the difference?

fit <- lm(age ~ longest_shell + diameter + height +
                           whole_weight + shucked_weight +
                           viscera_weight + shell_weight + num_type,
                         data = new_abalone)
summary(fit)
```

```
##
## Call:
## lm(formula = age ~ longest_shell + diameter + height + whole_weight +
##     shucked_weight + viscera_weight + shell_weight + num_type,
##     data = new_abalone)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10.5991  -1.3120  -0.3549   0.8968  14.0582
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)      5.2593     0.2827  18.603  < 2e-16 ***
## longest_shell   -0.8264     1.8122  -0.456    0.648
## diameter        11.9640     2.2254   5.376 8.02e-08 ***
## height          11.2045     1.5374   7.288 3.75e-13 ***
## whole_weight     9.0702     0.7270  12.476  < 2e-16 ***
## shucked_weight -20.1061     0.8168 -24.617  < 2e-16 ***
## viscera_weight -10.1551     1.2941  -7.847 5.36e-15 ***
## shell_weight     8.7011     1.1277   7.716 1.49e-14 ***
## num_type        -0.3885     0.0467  -8.319  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Residual standard error: 2.2 on 4168 degrees of freedom
## Multiple R-squared:  0.5353, Adjusted R-squared:  0.5345
## F-statistic: 600.3 on 8 and 4168 DF,  p-value: < 2.2e-16
```

```
summary(lm_model)
```

```
##          Length Class    Mode
## args     2      -none-   list
## eng_args 0      quosures list
## mode     1      -none-   character
## method   0      -none-   NULL
## engine   1      -none-   character
```

Question 5

Now:

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_recipe)
```

Question 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, abalone_train)

test <- data.frame(longest_shell = 0.50,
                   diameter = 0.10,
                   height =0.30,
                   whole_weight = 4,
                   shucked_weight = 1,
                   viscera_weight = 2,
                   shell_weight = 1,
                   num_type = 1)

lm_fit %>%
  extract_fit_parsnip() %>%
  tidy()
```

```
## # A tibble: 9 x 5
##   term           estimate std.error statistic   p.value
##   <chr>             <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)        5.30     0.315      16.8  6.16e- 61
```

```
## 2 longest_shell    -0.311    1.99      -0.157 8.76e-  1
## 3 diameter         11.7      2.44       4.80  1.69e-  6
## 4 height            9.68     1.60       6.05  1.56e-  9
## 5 whole_weight      9.13     0.818     11.2   1.99e- 28
## 6 shucked_weight  -20.1      0.914    -22.0   2.46e-100
## 7 viscera_weight  -10.5      1.45      -7.23  6.09e- 13
## 8 shell_weight      8.88     1.25       7.09  1.65e- 12
## 9 num_type         -0.414    0.0514    -8.05  1.11e- 15
```

```r
predict(lm_fit, new_data = test)
```

```
## # A tibble: 1 x 1
##   .pred
##   <dbl>
## 1  13.1
```

```r
lm_fit
```

```
## == Workflow [trained] ==========================================================
## Preprocessor: Recipe
## Model: linear_reg()
##
## -- Preprocessor ----------------------------------------------------------------
## 0 Recipe Steps
##
## -- Model -----------------------------------------------------------------------
##
## Call:
## stats::lm(formula = ..y ~ ., data = data)
##
## Coefficients:
##    (Intercept)   longest_shell        diameter          height    whole_weight
##         5.3023         -0.3110         11.7018          9.6847          9.1312
## shucked_weight  viscera_weight    shell_weight        num_type
##       -20.1247        -10.4850          8.8776         -0.4144
```

Question 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.

From what we got for R^2, it is not significant enough to show a strong correlation, it can only be considered relatively strong, but it is not significant enough to compare with our initial function which is age = rings + 1.5. Though the question dint require us to plot a graph, but the scatter plot should be a clear visual representation that our model didn't do well.

```r
abalone_train_res <- predict(lm_fit, new_data = abalone_train %>%
                               select(-age, -rings, -type))

abalone_train_res %>%
  head()
```

```
## # A tibble: 6 x 1
##   .pred
##   <dbl>
## 1  9.34
## 2  8.27
## 3  9.40
## 4  9.81
## 5  9.95
## 6  9.99
```

```r
abalone_train_res <- bind_cols(abalone_train_res, abalone_train %>% select(age))

abalone_train_res %>%
  head()
```

```
## # A tibble: 6 x 2
##   .pred   age
##   <dbl> <dbl>
## 1  9.34   8.5
## 2  8.27   8.5
## 3  9.40   9.5
## 4  9.81   8.5
## 5  9.95   9.5
## 6  9.99   9.5
```

```r
rmse(abalone_train_res, truth = age, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard        2.18
```

```r
abalone_metrics <- metric_set(rmse, rsq, mae)

abalone_metrics(abalone_train_res, truth = age,
                estimate = .pred)
```

```
## # A tibble: 3 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       2.18
## 2 rsq     standard       0.542
## 3 mae     standard       1.58
```

```
abalone_train_res %>%
  ggplot(aes(x = .pred, y = age)) +
  geom_point(alpha = 0.2) +
  geom_abline(lty = 2) +
  theme_bw() +
  coord_obs_pred()
```