HAHN_Lab3

Hope Hahn

1/31/2024

Lab 3: Predicting the age of abalone

Abalones are marine snails. Their flesh is widely considered to be a desirable food, and is consumed raw or cooked by a variety of cultures. The age of abalone is determined by cutting the shell through the cone, staining it, and counting the number of rings through a microscope – a boring and time-consuming task. Other measurements, which are easier to obtain, are used to predict the age.

The data set provided includes variables related to the sex, physical dimensions of the shell, and various weight measurements, along with the number of rings in the shell. Number of rings is the stand-in here for age.

Data Exploration

Pull the abalone data from Github and take a look at it.

```
# read in data and look at it
abdat<- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/data/al
```

```
## New names:
## Rows: 4177 Columns: 10
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): Sex dbl (9): ...1, Length, Diameter, Height, Whole_weight, Shucked_weight,
## Visce...
## 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.
## * `` -> `...1`
```

```
glimpse(abdat)
```

```
## Rows: 4,177
## Columns: 10
## $ ...1           <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, ~
## $ Sex            <chr> "M", "M", "F", "M", "I", "I", "F", "F", "M", "F", "F", ~
## $ Length         <dbl> 0.455, 0.350, 0.530, 0.440, 0.330, 0.425, 0.530, 0.545,~
## $ Diameter       <dbl> 0.365, 0.265, 0.420, 0.365, 0.255, 0.300, 0.415, 0.425,~
## $ Height         <dbl> 0.095, 0.090, 0.135, 0.125, 0.080, 0.095, 0.150, 0.125,~
## $ Whole_weight   <dbl> 0.5140, 0.2255, 0.6770, 0.5160, 0.2050, 0.3515, 0.7775,~
## $ Shucked_weight <dbl> 0.2245, 0.0995, 0.2565, 0.2155, 0.0895, 0.1410, 0.2370,~
## $ Viscera_weight <dbl> 0.1010, 0.0485, 0.1415, 0.1140, 0.0395, 0.0775, 0.1415,~
## $ Shell_weight   <dbl> 0.150, 0.070, 0.210, 0.155, 0.055, 0.120, 0.330, 0.260,~
## $ Rings          <dbl> 15, 7, 9, 10, 7, 8, 20, 16, 9, 19, 14, 10, 11, 10, 10, ~
```

Data Splitting

- ***Question 1***. Split the data into training and test sets. Use a 70/30 training/test split.

```
# Data splitting with {rsample}
set.seed(123) #set a seed for reproducibility
split <- initial_split(abdat)
split
```

```
## <Training/Testing/Total>
## <3132/1045/4177>
```

```
# save training and testing data
train_data <- training(split)
test_data  <- testing(split)
```

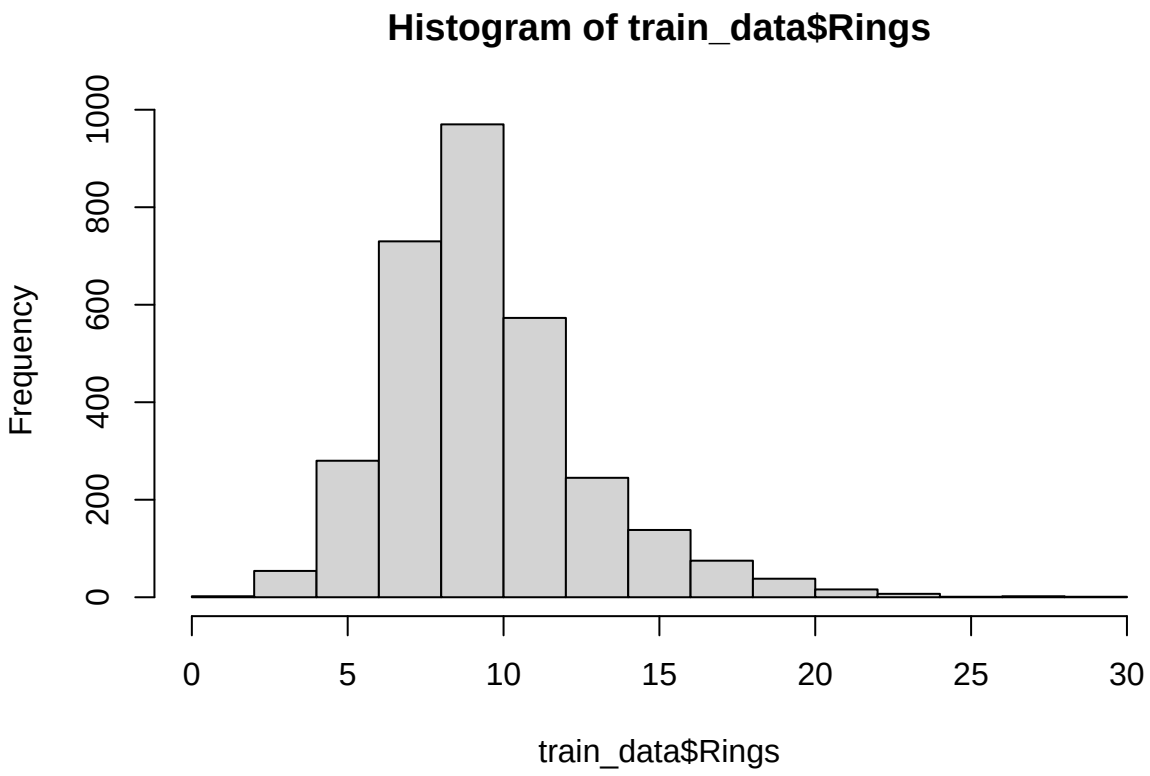
We'll follow our textbook's lead and use the caret package in our approach to this task. We will use the glmnet package in order to perform ridge regression and the lasso. The main function in this package is glmnet(), which can be used to fit ridge regression models, lasso models, and more. In particular, we must pass in an x matrix of predictors as well as a y outcome vector , and we do not use the y x syntax.

Fit a ridge regression model

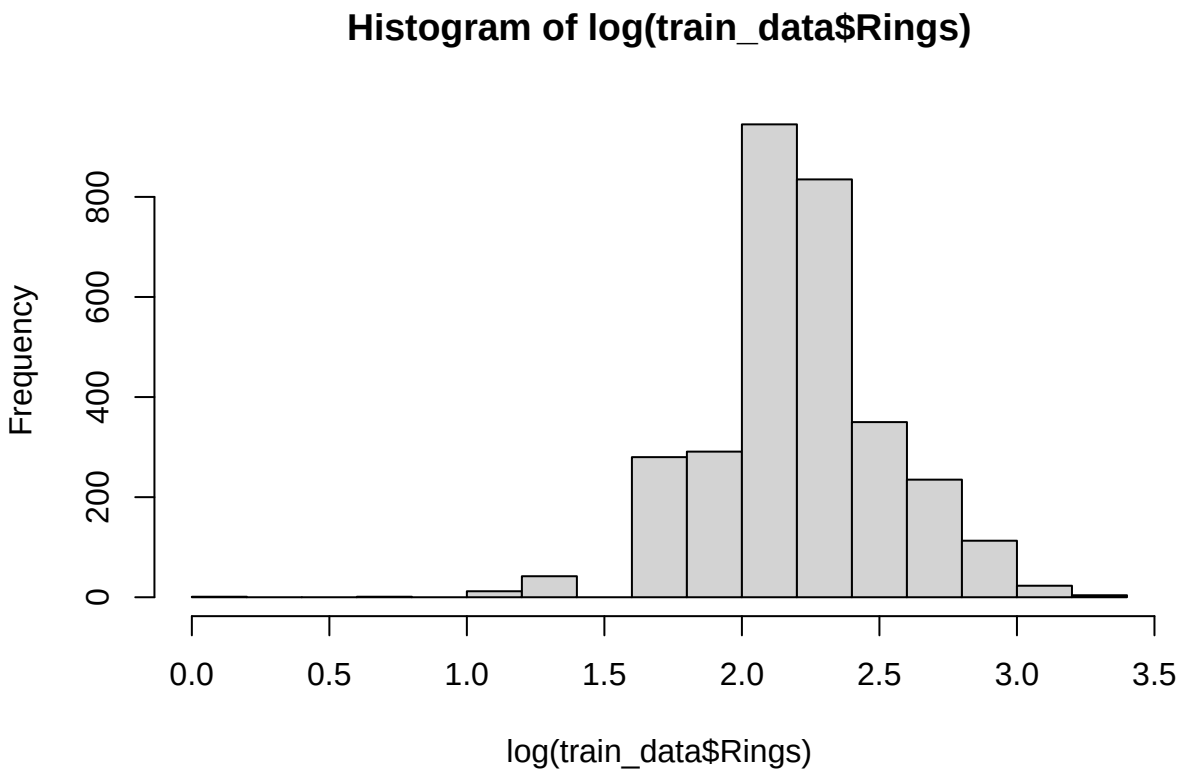
- ***Question 2***. Use the model.matrix() function to create a predictor matrix, x, and assign the Rings variable to an outcome vector, y.

```
# Create training feature matrices using model.matrix()
# rings stands in for age
x <- model.matrix(Rings ~ ., train_data)[,-1]

# check data distribution
hist(train_data$Rings)
```




```
hist(log(train_data$Rings))
```



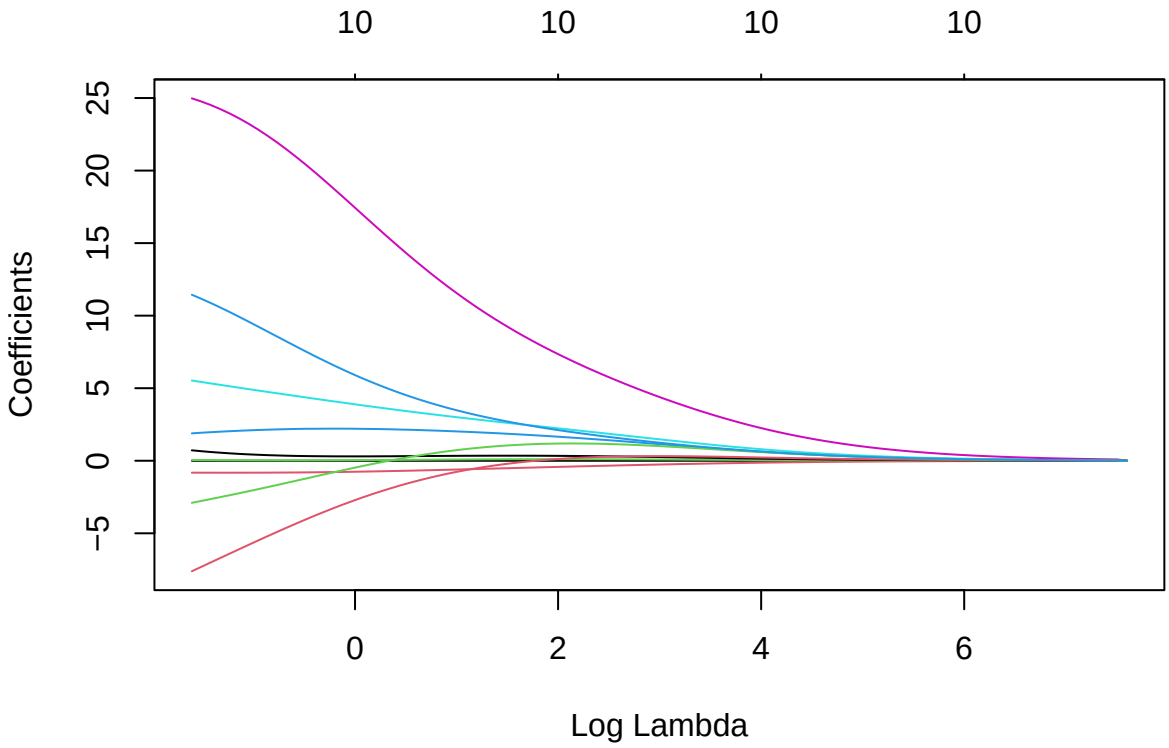

```
# data is not too much better logged, will just use original data

# training data
y <- train_data$Rings
```

- ***Question 3***. Fit a ridge model (controlled by the alpha parameter) using the glmnet() function. Make a plot showing how the estimated coefficients change with lambda. (Hint: You can call plot() directly on the glmnet() objects).

```
# fit a ridge model, passing x, y, alpha to glmnet()
ridge <- glmnet(
  x = x,
  y = y,
  alpha = 0
)

# plot() the glmnet model object
plot(ridge, xvar = "lambda")
```



Using *k*-fold cross validation resampling and tuning our models

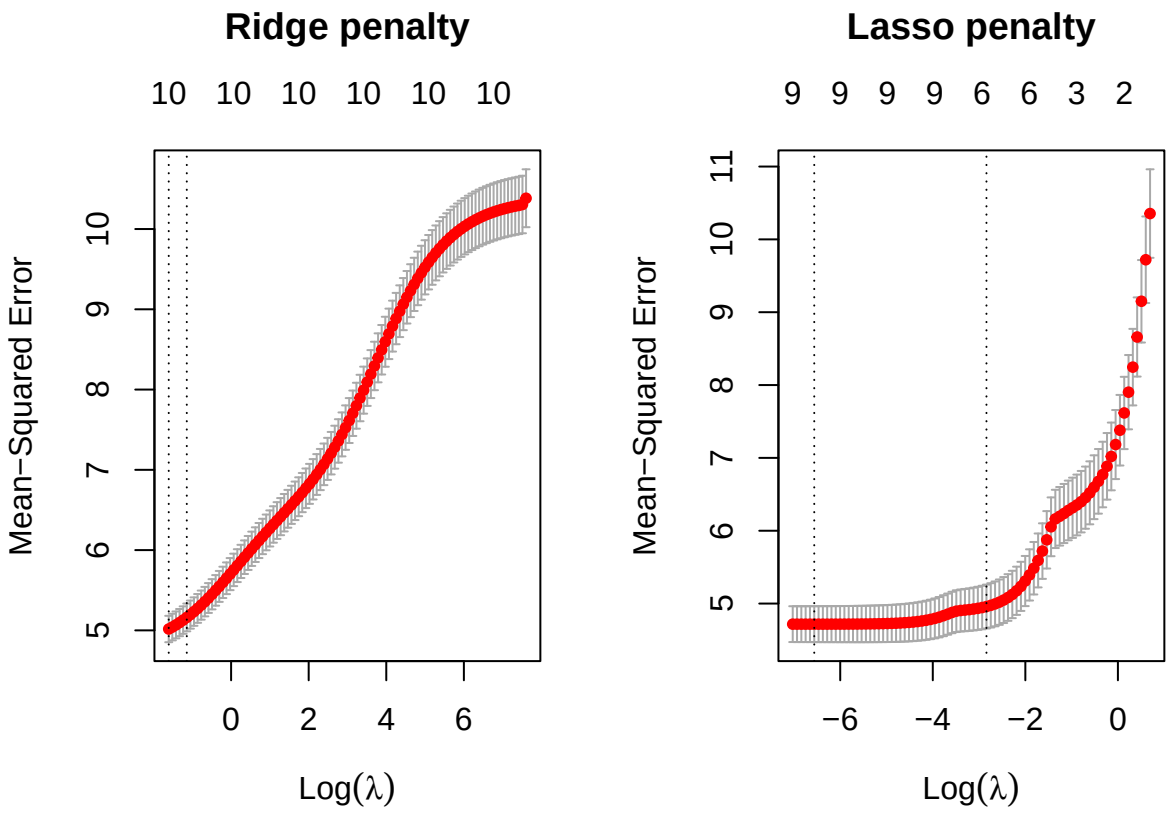
In lecture we learned about two methods of estimating our model's generalization error by resampling, cross validation and bootstrapping. We'll use the *k*-fold cross validation method in this lab. Recall that lambda is a tuning parameter that helps keep our model from over-fitting to the training data. Tuning is the process of finding the optima value of lamba.

- ***Question 4***. This time fit a ridge regression model and a lasso model, both with using cross validation. The glmnet package kindly provides a cv.glmnet() function to do this (similar to the glmnet() function that we just used). Use the alpha argument to control which type of model you are running. Plot the results.

```
# Apply k-fold CV to ridge
ridge_cv <- cv.glmnet(
  x = x,
  y = y,
  alpha = 0
)

# Apply k-fold CV to lasso
lasso_cv <- cv.glmnet(
  x = x,
  y = y,
  alpha = 1
)
```

```r
# plot results
par(mfrow = c(1, 2))
plot(ridge_cv, main = "Ridge penalty\n\n")
plot(lasso_cv, main = "Lasso penalty\n\n")
```



- ***Question 5***. Interpret the graphs. What is being displayed on the axes here? How does the performance of the models change with the value of lambda?
  - **In both plots, the dashed line on the left shows the log(lambda) of the lowest MSE while the dashed line on the right shows the log(lambda) within one standard error of the lowest MSE. The left (ridge penalty) plot shows that a lower penalty (log(lambda)) has a lower MSE. As soon as the penalty increases, the MSE also increases. Additionally, the plot on the right (lasso penalty) shows that lower penalty also has a better model fit. However, for this plot, it shows that slightly increasing the penalty decreases the MSE initially. The MSE then begins to increase as the number of features decreases and the penalty increases. This shows that the regular OLS model may overfit the data, but adding too much penalty decreases model accuracy.**
- ***Question 6***. Inspect the ridge model object you created with cv.glmnet(). The $cvm column shows the MSEs for each CV fold. What is the minimum MSE? What is the value of lambda associated with this MSE minimum?
  - **The minimum MSE for the ridge model is 5.016978, and the value of lambda with the minimum MSE is 0.2004132.**

```r
# Ridge model
# minimum MSE
ridge_min_MSE <- min(ridge_cv$cvm)

# lambda for min MSE
ridge_min_lam <- ridge_cv$lambda.min

ridge_min_MSE
```

```
## [1] 5.016978
```

```r
ridge_min_lam
```

```
## [1] 0.2004132
```

- ***Question 7***. Do the same for the lasso model. What is the minimum MSE? What is the value of lambda associated with this MSE minimum?
  - **The minimum MSE for the lasso model is 4.71733, and the value of lambda for this MSE is 0.001413875.**

```r
# Lasso model
# minimum MSE
lasso_min_MSE <- min(lasso_cv$cvm)

# lambda for min MSE
lasso_min_lam <- lasso_cv$lambda.min

lasso_min_MSE
```

```
## [1] 4.71733
```

```r
lasso_min_lam
```

```
## [1] 0.001413875
```

Data scientists often use the "one-standard-error" rule when tuning lambda to select the best model. This rule tells us to pick the most parsimonious model (fewest number of predictors) while still remaining within one standard error of the overall minimum cross validation error. The cv.glmnet() model object has a column that automatically finds the value of lambda associated with the model that produces an MSE that is one standard error from the MSE minimum ($lambda.1se).

- ***Question 8.*** Find the number of predictors associated with this model (hint: the $nzero is the # of predictors column).
  - **Using the "one-standard-error" rule, the number of predictors would be 6.**

```r
# predictors for 1 standard error away from minimum MSE
lasso_cv$nzero[lasso_cv$lambda == lasso_cv$lambda.1se]
```

```
## s38
##   6
```

- ***Question 9.*** Which regularized regression worked better for this task, ridge or lasso? Explain your answer.
  - **The lasso regression worked better for this task because it minimized the mean squared errors and selected the most relevant features.**