

Lab 3

Andrew Bartnik

2023-01-25

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.

```
abdat<- dat <- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/data/abalone-data.csv")[, -1]
```

```
## 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: 9
## $ 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, ...
```

I’m cutting out the first column here - it contains the index of each row and I don’t want it to interfere with our predictions.

Data Splitting

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

```
#set seed for reproducibility
set.seed(123)

#split the data into training and testing
split <- initial_split(abdat, prop = 0.7, strata = Rings)
ab_train <- training(split)
ab_test <- testing(split)

#taking a look at the training data
skimr::skim(ab_train)
```





Data summary

Name	ab_train
Number of rows	2922
Number of columns	9
Column type frequency:	
character	1
numeric	8
Group variables	
None	

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
Sex	0	1	1	1	0	3	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Length	0	1	0.52	0.12	0.07	0.45	0.54	0.62	0.78	
Diameter	0	1	0.41	0.10	0.06	0.35	0.42	0.48	0.63	
Height	0	1	0.14	0.04	0.00	0.12	0.14	0.16	0.24	
Whole_weight	0	1	0.83	0.49	0.00	0.44	0.80	1.15	2.83	

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Shucked_weight	0	1	0.36	0.22	0.00	0.19	0.34	0.50	1.49	
Viscera_weight	0	1	0.18	0.11	0.00	0.09	0.17	0.25	0.58	
Shell_weight	0	1	0.24	0.14	0.00	0.13	0.23	0.33	0.90	
Rings	0	1	9.93	3.20	1.00	8.00	9.00	11.00	29.00	

We'll follow our text book'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.

```
# Using model.matrix to create a predictor matrix
X <- model.matrix(Rings ~ ., ab_train)[,-1]

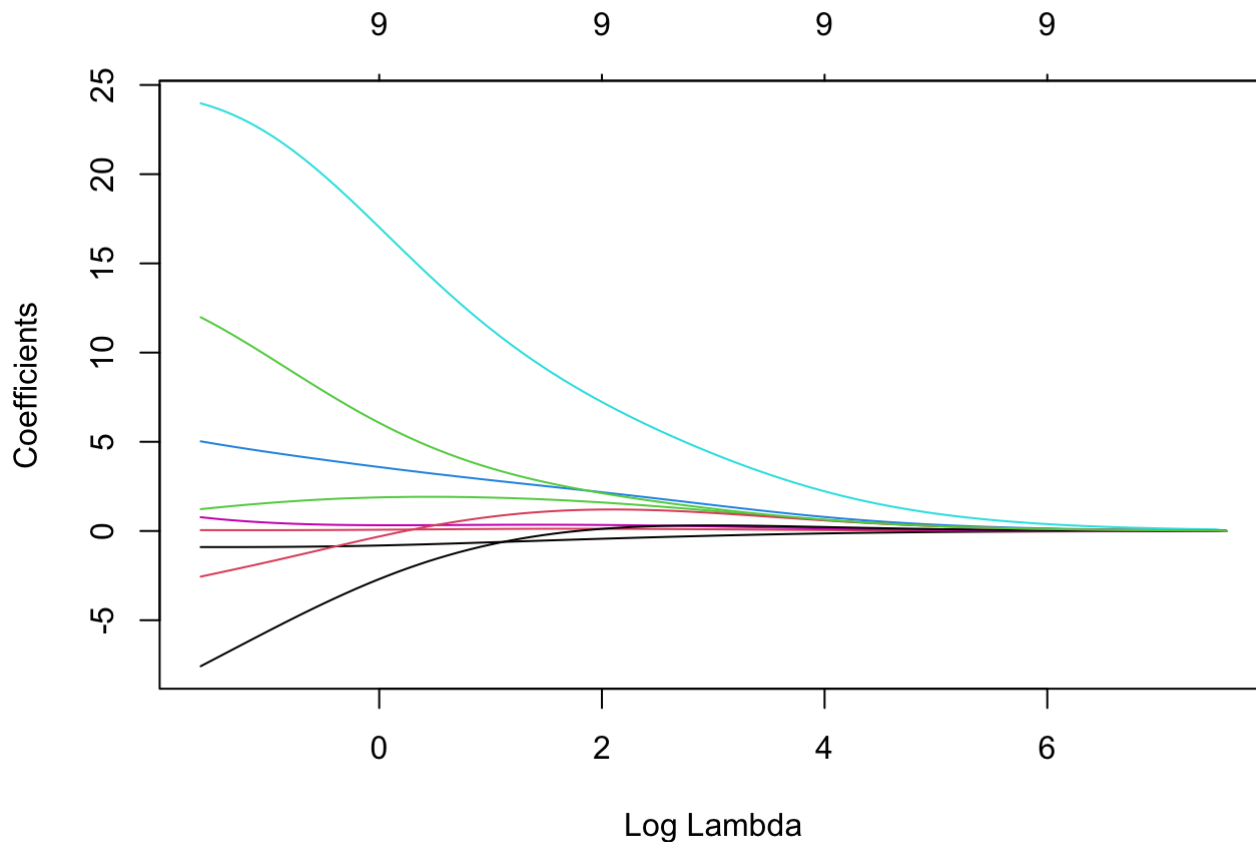
#assigning rings variable to y
y <- ab_train$Rings
```

- **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).

```
# Fitting and plotting a ridge model
ridge <- glmnet(x = X, y = y, alpha = 0)

plot(ridge, xvar = "lambda", main = "Coefficient magnitude as penalization increases\n\n")
```

Coefficient magnitude as penalization increases



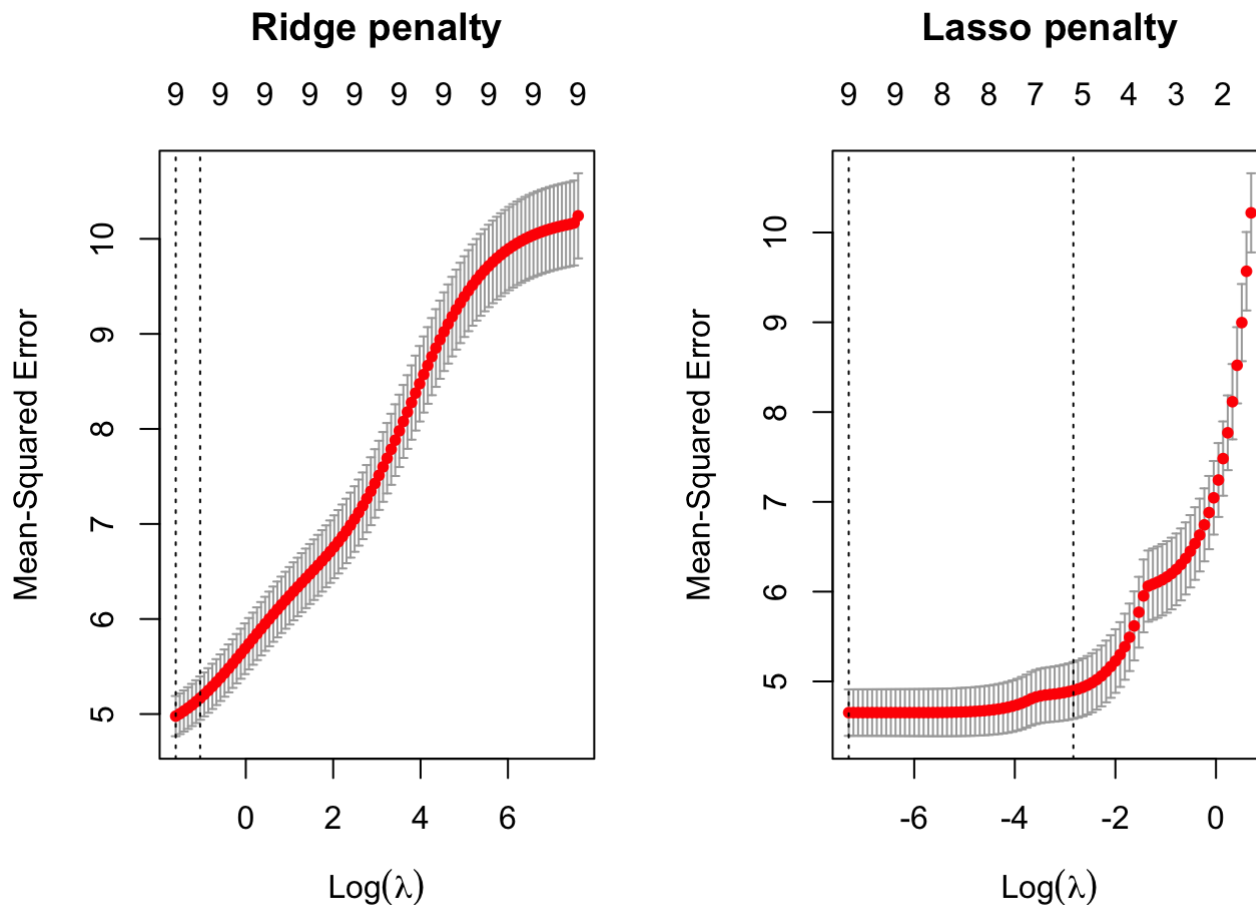
Using k -fold cross validation resampling and tuning our models

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 lambda.

- **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.

```
# fitting both ridge and lasso model with 10-fold CV
ridge2 <- cv.glmnet(x = X, y = y, alpha = 0)
lasso2 <- cv.glmnet(x = X, y = y, alpha = 1)

#plotting them
par(mfrow = c(1, 2))
plot(ridge2, xvar = "lambda", main = "Ridge penalty\n\n")
plot(lasso2, xvar = "lambda", main = "Lasso penalty\n\n")
```



- **Question 5.** Interpret the graphs. What is being shown on the axes here? How does the performance of the models change with the value of lambda?

The y axis represents the Mean Squared Error (MSE), and appears to be its lowest at the smallest x-value (our penalty, $\log \lambda$), suggesting that an OLS model will be a good choice. In general, as we increase our penalty we constrain both of our models. However, as we go farther along the x axis, the MSE increases at a much higher rate in the ridge model than the lasso model - suggesting that the lasso model will perform better than the ridge model while also selecting for predictors.

- **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?

```
#finding the minimum MSE
min(ridge2$cvm)
```

```
## [1] 4.978248
```

```
# and its lambda value
ridge2$lambda.min
```

```
## [1] 0.2012142
```

The smallest MSE for the ridge model is 4.97, and its associated lambda is 0.2012.

- **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?

```
#Finding the minimum MSE
min(lasso2$cvm)
```

```
## [1] 4.652762
```

```
# and its lambda value
lasso2$lambda.min
```

```
## [1] 0.0006743901
```

The smallest MSE for the lasso model is 4.65, and its associated lambda is 0.00067.

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).

```
# Finding number of predictors associated with the optimal model - ridge
ridge2
```

```
##
## Call: cv.glmnet(x = X, y = y, alpha = 0)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.2012   100   4.978 0.2116         9
## 1se 0.3516    94   5.168 0.2279         9
```

```
#finding lambda within 1 se of lowest MSE value
ridge2$lambda.1se
```

```
## [1] 0.3516275
```

```
#number of predictors associated with this best lambda value
ridge2$nzero[94]
```

```
## s93
## 9
```

```
# Finding number of predictors associated with the optimal model - lasso
lasso2
```

```
##
## Call:  cv.glmnet(x = X, y = y, alpha = 1)
##
## Measure: Mean-Squared Error
##
##      Lambda Index Measure      SE Nonzero
## min 0.00067    87   4.653 0.2587         9
## 1se 0.05866    39   4.899 0.3162         5
```

```
#finding lambda within 1 se of lowest MSE value
lasso2$lambda.1se
```

```
## [1] 0.05865501
```

```
#number of predictors associated with this best lambda value
lasso2$nzero[39]
```

```
## s38
##    5
```

- - There are 9 predictors associated with the optimal ridge model, and 5 predictors associated with the optimal lasso model.
- **Question 9.** Which regularized regression worked better for this task, ridge or lasso? Explain your answer.
 - The MSE for the best lasso model within 1 standard error of the MSE minimum was 4.899, while the MSE for the best ridge model within 1 standard error of the MSE minimum was 5.168, indicating that the lasso model outperforms the ridge model. Lasso also made the model more parsimonious - it was able to reduce the number of features necessary for prediction from 9 to 5 while remaining within 1 standard error of the minimum cross validation error.