Ingersoll_Lab3

Sofia Ingersoll

2024-02-01

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.

Data Splitting

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

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.

```
#-------Split data for model------
# creating initial data split, default is 70/30
split <- initial_split(abdat)

# training data
ab_train <- training(split)

# testing data
ab_test <- testing(split)
```

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.

```r
#------Create Predictor Matrix------
# Create training feature matrices using model.matrix()
# (auto encodes of categorical variables)

# for Rings, assess the relationship amongst all variables and remove
# the intercept variable of -1.
X <- model.matrix(Rings ~ ., data = ab_train)[,-1]


# Using skim(ab_train$Rings) in the console revealed a mean of 9.92 with
# a sd of 3.19. The histogram provided looked right skewed, indicating
# a potential need for a log transformation. Below we will apply
# a log transformation and run skim(Y) to test our theory.




#------Log Transform Rings------
# transform y with log() transformation
Y <- log(ab_train$Rings)

# In the Console, running skim(Y) displayed the results of the transformation.
# The new mean is 2.25 with a significantly better sd of 0.317.
# The histogram percentiles juxtaposed to skim(ab_train$Rings) are
# more evenly distributed.
```

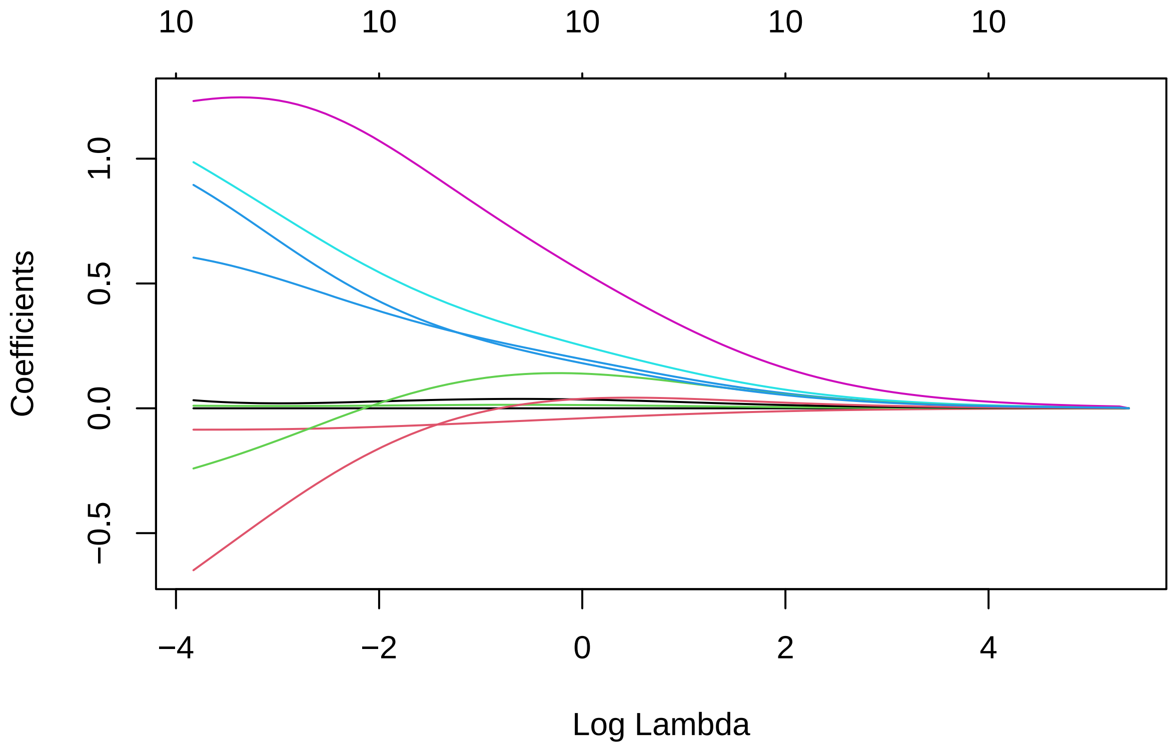
- ***Question 3***. Fit a ridge model (controlled by the alpha parameter) using the glmnet() function. Make a plot showing how the estimated coefficients change with lambda. (Hint: You can call plot() directly on the glmnet() objects).

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.

```r
#------k-fold Cross Validation------
# fit a ridge model, passing X,Y, and alpha = 0 to glmnet()
ridge <- glmnet(
  x = X,
  y = Y,
  alpha = 0                   # tells function whether we want to model
                              # ridge = 0, lasso = 1,
                              # anything in between is Elastic.
)


#------Plot Model Object------
# plot() the glmnet() model object
# we are simplifying the model by including this penalty term to the sum of
# squared errors. This is to remove unnecessary coefficients
plot(ridge,
     xvar = 'lambda')
```



```
# As we increase the size of lambda, we can see the coefficients are
# shrinking towards 0.
```

- ***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.

```
#------Double Cross Validation Model Fit------
# Cross validation for tuning.
# We are going to be resampling our data by breaking it into pieces and running
# the multi-folds. The average of this will be our assessment of model
# efficiency on unseen data. The default number of folds is 10-fold.

#------Ridge Regression Model------
# cv is telling us to use cross validation
ridge <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 0
)


#------Lasso Model------
# Apply CV lasso regression to Ames data
```

```r
lasso <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 1
)
```

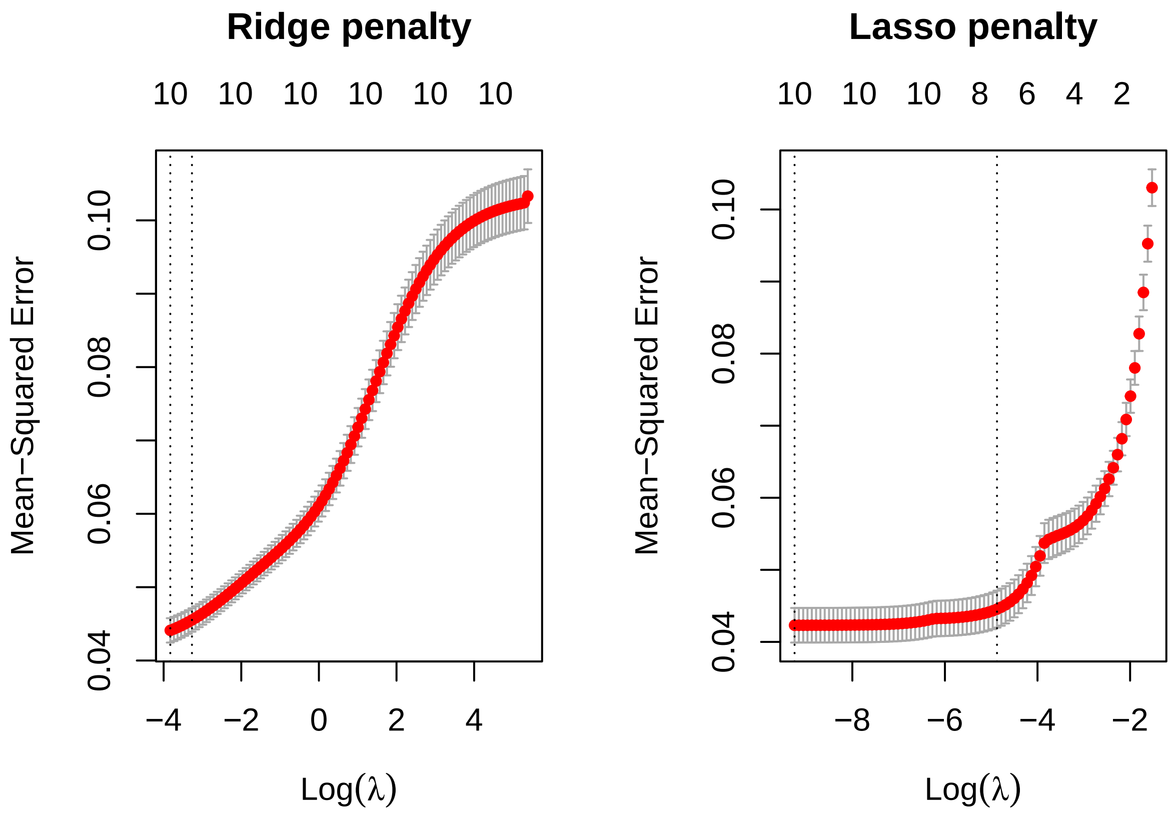
- ***Question 5***. Interpret the graphs. What is being displayed on the axes here? How does the performance of the models change with the value of lambda?

Each plot is displaying how well each model is predicting the hold out cell – the average mean-squared error the entire cross validation model. As lambda increases, the MSE is increasing. We want a lambda that minimizes MSE. This is telling us where our optimal points are for MSE. The dotted lines are created by the one standard error rule (within 1 std error of the folds).

```r
#------Interpreting Model Performance------
# plot results
par(mfrow = c(1, 2))

#------Ridge Model Plot------
ridge_mod_plot <- plot(ridge, main = "Ridge penalty\n\n")

#------Lasso Model Plot------
lasso_mod_plot <- plot(lasso, main = "Lasso penalty\n\n")
```



Ridge penalty

Lasso penalty

- ***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 min MSE for the ridge model is ~`0.0450` with the value of `0.0208` for the associated lambda.

```
#-------Ridge model----------------------

# minimum MSE output in console
min(ridge$cvm)
```

```
## [1] 0.04411032
```

```
# lambda value at the min MSE
ridge$lambda.min
```

```
## [1] 0.02174702
```

```
# 1-SE rule
# output a lambda value that is separated by the smallest 1-squared error
ridge$cvm[ridge$lambda == ridge$lambda.1se]
```

```
## [1] 0.04552346
```

```
# lambda for this MSE
ridge$lambda.1se
```

```
## [1] 0.03800354
```

- ***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?

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).

The min MSE for the lasso model is `0.0426` with the value of `0.000101` for the associated lambda.

```
#-------Lasso model--------

# minimum MSE
min(lasso$cvm)
```

```
## [1] 0.04230738
```

```
# lambda value at the min MSE
lasso$lambda.min
```

```
## [1] 9.635282e-05
```

```
# 1-SE rule
# same as above

# lambda for this MSE
lasso$lambda.1se
```

```
## [1] 0.007635796
```

- ***Question 8.*** Find the number of predictors associated with this model (hint: the $nzero is the # of predictors column).

The number of predictors initially is 10 and is reduced to 7 using the lasso model penalty.

```
#-------Lasso model----------

# No. of coef | 1-SE MSE
# this is to access our feature selection
# numbers that went to zero
lasso$nzero[lasso$lambda == lasso$lambda.min]
```

```
## s83
##  10
```

```
# what is the affect of using that standard rule of 1-SE, we reduce the number of coefficients from 10
lasso$nzero[lasso$lambda == lasso$lambda.1se]
```

```
## s36
##   7
```

- ***Question 9.*** Which regularized regression worked better for this task, ridge or lasso? Explain your answer.

The min MSE for both models is relatively similar, ~`0.0450`. However the ridge model reflected an associated lambda value of `0.0208` for the MSE. There is a significantly smaller margin for the 1-SE rule observed in the ridge regression model. Additionally, the convergence to zero is swifter and more uniform in the ridge model. For these reasons, the best regularized regression model applied was the ridge.

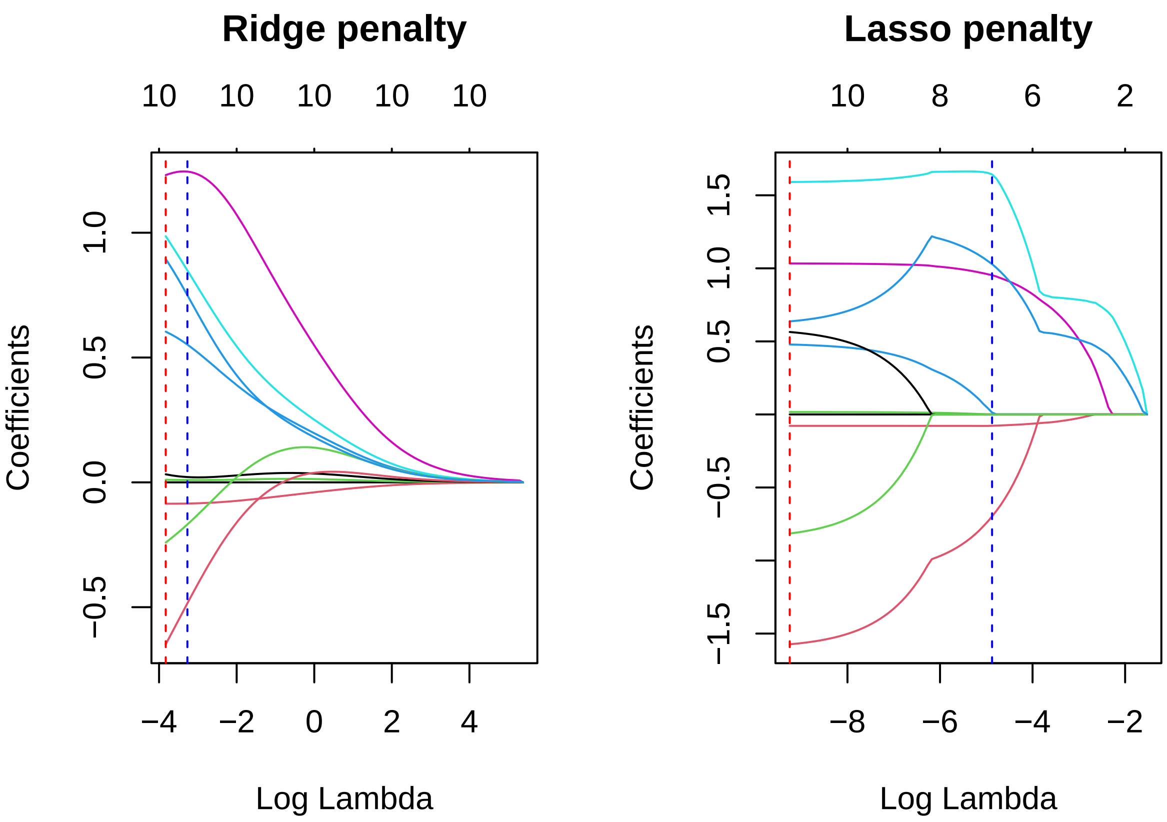
```
#------Ridge model-------
ridge_min <- glmnet(
  x = X,
  y = Y,
  alpha = 0
)

#------Lasso model------
lasso_min <- glmnet(
  x = X,
  y = Y,
  alpha = 1
)
```

```r
par(mfrow = c(1, 2))

#------plot ridge model------
plot(ridge_min,
     xvar = "lambda",
     main = "Ridge penalty\n\n")
abline(v = log(ridge$lambda.min),
       col = "red", lty = "dashed")
abline(v = log(ridge$lambda.1se),
       col = "blue", lty = "dashed")

#-------plot lasso model------
plot(lasso_min,
     xvar = "lambda",
     main = "Lasso penalty\n\n")
abline(v = log(lasso$lambda.min),
       col = "red", lty = "dashed")
abline(v = log(lasso$lambda.1se),
       col = "blue", lty = "dashed")
```


Ridge penalty
Coefficients
Log Lambda
Lasso penalty
Coefficients
Log Lambda