Muir_Lab3

Sam Muir

30 January 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
abdat <- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/data/a

#glimpse(abdat)
#skim(abdat)
# looking at the distribution of the var. we are interested in (Rings) we can see that it is roughly no

# remove second indexing column
abdat <- abdat %>%
  select(-...1)
```

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. 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.

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

# Data splitting with {rsample}
split <- initial_split(abdat, prop = 0.7) # indicate 70/30 split
# shows amount of obs. in training/testing/total
split
```

```
## <Training/Testing/Total>
## <2923/1254/4177>
```

```r
# pull train and test data from the split
ab_train <- training(split)
ab_test  <- testing(split)
```

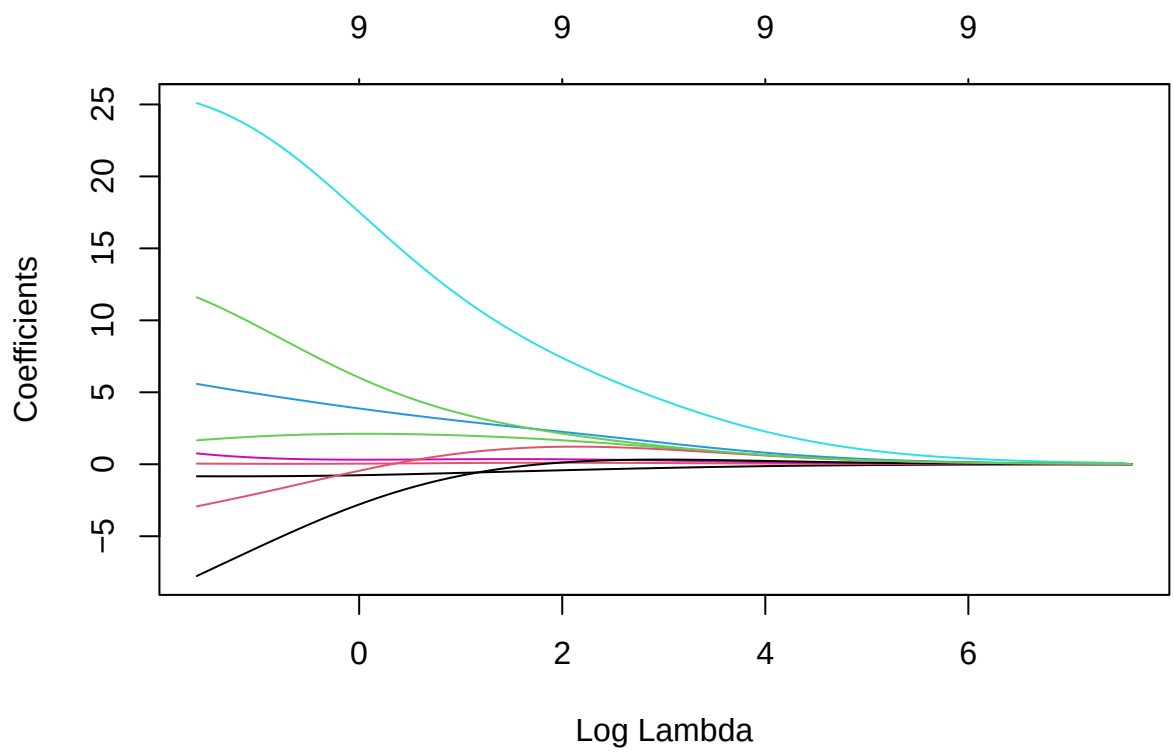
Fit a ridge regression model

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

```r
# predictor
X <- model.matrix(Rings~., data = ab_train)
# outcome
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).

```r
# ridge model
ab_ridge <- glmnet(x = X,
                   y = Y,
                   alpha = 0) # 0 for ridge
# plot ridge model
plot(ab_ridge, xvar = "lambda")
```



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

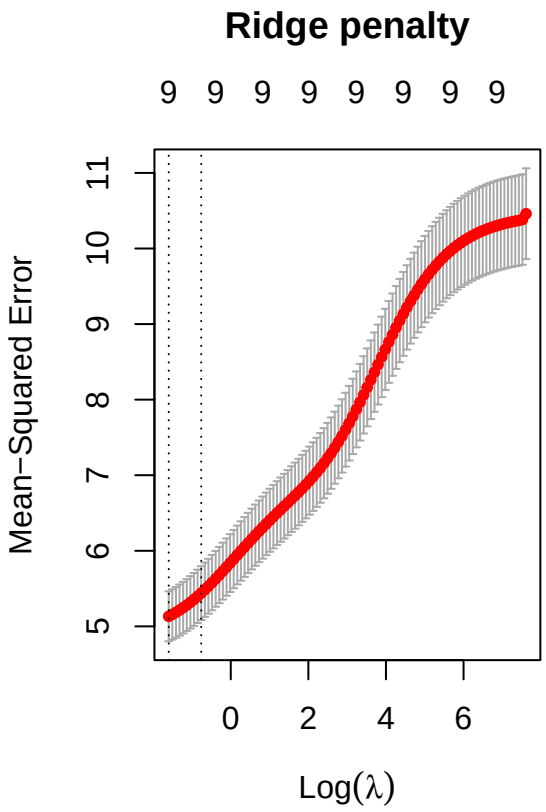
In lecture we learned about two methods of estimating our model's generalization error by resampling, cross validation and bootstrapping. We'll use the *k*-fold cross validation method in this lab. Recall that lambda is a tuning parameter that helps keep our model from over-fitting to the training data.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

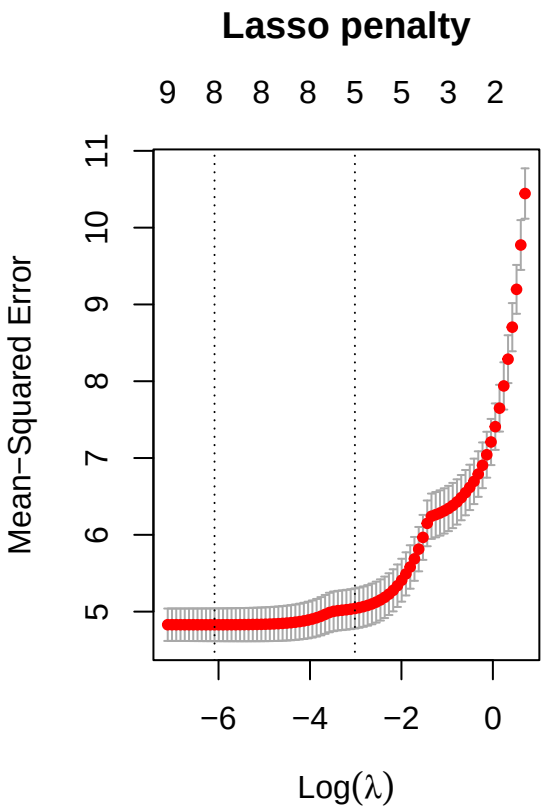
- ***Question 4***. This time fit a ridge regression model and a lasso model, both with using cross validation. The glmnet package kindly provides a cv.glmnet() function to do this (similar to the glmnet() function that we just used). Use the alpha argument to control which type of model you are running.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

```
# Apply CV ridge regression
ab_ridge <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 0 # specify ridge
)

# Apply CV lasso regression
ab_lasso <- cv.glmnet(
  x = X,
  y = Y,
  alpha = 1 # specify lasso
  )

# plot results
par(mfrow = c(1, 2))
# see how well the model is performing within the folds
plot(ab_ridge, main = "Ridge penalty\n\n")
plot(ab_lasso, 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?

The axis are showing the log of lambda on the x-axis and the mean-squared error on the y-axis. The first dotted vertical line on the plots shows which value of lambda gives the lowest MSE. The second vertical line shows the lambda value for the 1 standard error rule, which is the lambda point for the most parsimonious model. For the ridge penalty model, as log(lambda) increases, so does the MSE. This is also generally true for the lasso penalty model, but a noticeable increase in MSE begins at values log(lambda) approximately > -4. The MSE also increases drastically at log(lambda) = 0. So generally, as log(lambda) increases, the performance of the model decreases.

- ***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?

```
# Ridge model ..............
# minimum MSE
print(paste("The minimum MSE is", min(ab_ridge$cvm)))
```

```
## [1] "The minimum MSE is 5.1329196906744"
```

```
print(paste("The value of lambda associated with this MSE minimum is", ab_ridge$lambda.min))
```

```
## [1] "The value of lambda associated with this MSE minimum is 0.202110669973881"
```

- ***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?

```r
# Lasso model ..............
# minimum MSE
print(paste("The minimum MSE is", min(ab_lasso$cvm)))
```

```
## [1] "The minimum MSE is 4.82869118071646"
```

```r
# lambda for this min MSE
print(paste("The value of lambda associated with this MSE minimum is", ab_lasso$lambda.min))
```

```
## [1] "The value of lambda associated with this MSE minimum is 0.00227035772902951"
```

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).

```r
# the number of predictors in the ridge model
ab_ridge$nzero[ab_ridge$lambda == ab_ridge$lambda.1se]
```

```
## s90
##   9
```

```r
# the number of predictors in the lasso model
ab_lasso$nzero[ab_lasso$lambda == ab_lasso$lambda.1se]
```

```
## s40
##   5
```

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

For this task, the lasso worked better. When just looking at the minimum MSE value, the lasso is smaller than ridge. When using the one-standard-error rule, the lasso model uses fewer predictors than the ridge. The lasso seems to be a more parsimonious model with fewer predictors within one standard error of the overall minimum cross validation error.