Ramji_Lab3

Anna Ramji

2024-01-31

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 <- read_csv(file = "https://raw.githubusercontent.com/MaRo406/eds-232-machine-learning/main/data/a
# glimpse(abdat) # commenting this out because I got marked
# down for having too many outputs in the last lab!

abdat <- abdat |>
    select(-...1)  # remove index column
```

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)  # for reproducibility

abdat_split <- initial_split(abdat)  # default is 70:30 split

abdat_train <- training(abdat_split)

abdat_test <- testing(abdat_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.

```
# predictor matrix
```

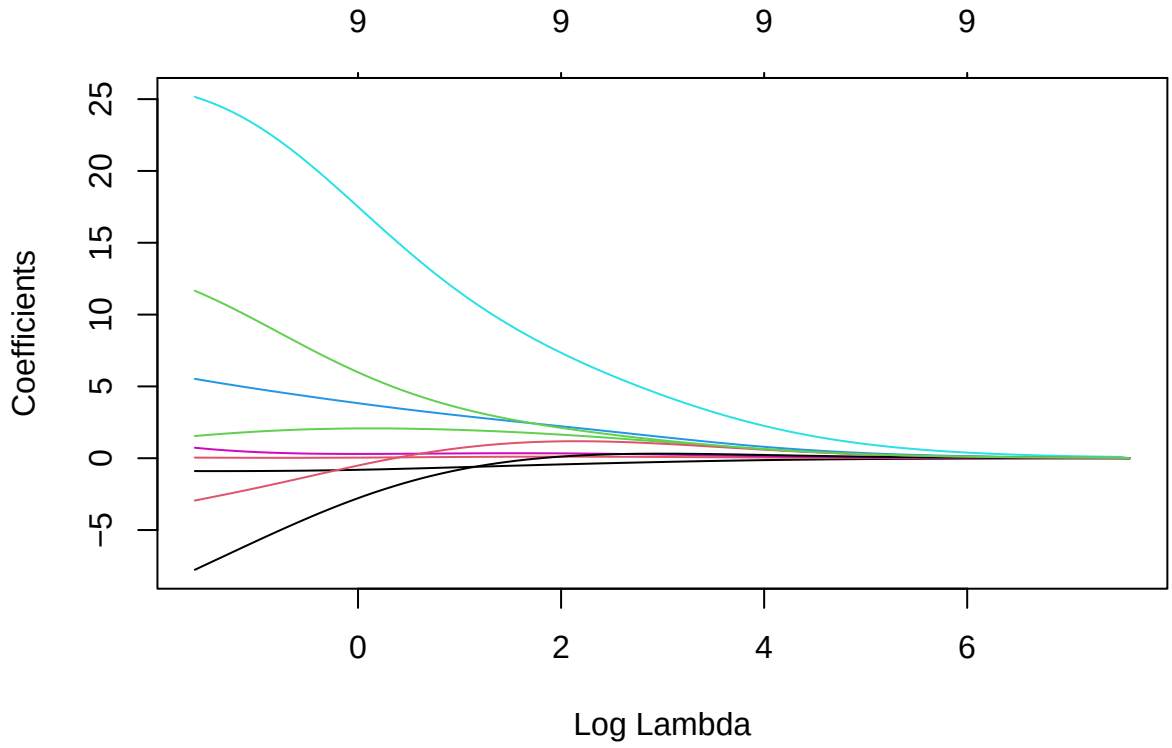
```r
x <- model.matrix(Rings ~ ., data = abdat_train)[, -1]  # remove intercept column

# after doing skim(abdat) in the console, Rings appears
# roughly normal
y <- abdat_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
# fitting ridge model
ridge <- glmnet(x = x, y = y, alpha = 0)

# plot showing how estimated coefficients change with
# lambda
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.

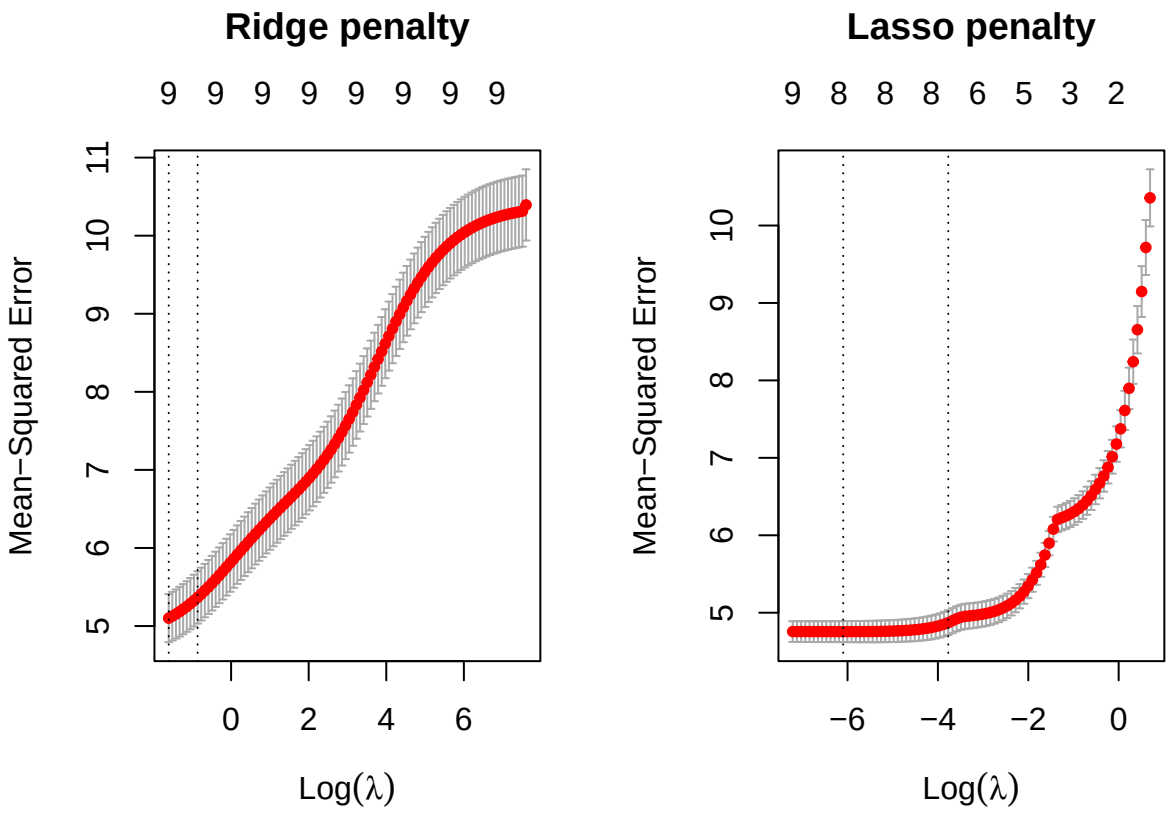
Plot the results.

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

# ridge cross validation
ridge_abd <- cv.glmnet(x = x, y = y, alpha = 0)  # ridge


# LASSO cross validation
lasso_abd <- cv.glmnet(x = x, y = y, alpha = 1)  # LASSO


# plot results
par(mfrow = c(1, 2))  # {graphics}, set or query graphical parameters
plot(ridge_abd, main = "Ridge penalty\n\n")
plot(lasso_abd, 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 x-axes are showing the values of lambda (log-transformed), and the x-axis shows the values of Mean-Squared Error. Each point on the line for the graphs shows the MSE at each value of lambda (red line showing average, grey showing variation over the 10 runs). The numbers at the top of the plots show the number of predictors in the model (constant for Ridge, decreasing for LASSO as it performs feature selection). In both models, as lambda increases, the MSE also increases. For the lasso model, the minimum MSE appears lower, and stays notably low up until around 7 or 8 predictors (features), whereas the Ridge model

starts with a MSE of around 5, and quickly increases as lambda increases. Based on visual appearance alone, the Lasso model appears to be a better fit for this data.

- ***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 min MSE
print(paste("The minimum MSE for the Ridge model is", round(min(ridge_abd$cvm),
    4)))
```

```
## [1] "The minimum MSE for the Ridge model is 5.1011"
```

```
# Lambda for min MSE
print(paste("The lambda value associated with the Ridge model's minimum MSE is",
    round(ridge_abd$lambda.min, 4)))
```

```
## [1] "The lambda value associated with the Ridge model's minimum MSE is 0.2004"
```

- ***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?

```
# ridge min MSE
print(paste("The minimum MSE in the Lasso model is", round(min(lasso_abd$cvm),
    4)))
```

```
## [1] "The minimum MSE in the Lasso model is 4.7559"
```

```
# Lambda for min MSE
print(paste("The lambda value associated with the minimum MSE in the Lasso model is",
    round(lasso_abd$lambda.min, 4)))
```

```
## [1] "The lambda value associated with the minimum MSE in the Lasso model is 0.0023"
```

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).

```
# No. of coef | 1-SE MSE (makes sense because with LASSO,
# we're performing feature selection)
lasso_predictors <- lasso_abd$nzero[lasso_abd$lambda == lasso_abd$lambda.1se]
print(paste("The number of predictors associated with this Lasso model is",
    lasso_predictors))
```

```
## [1] "The number of predictors associated with this Lasso model is 8"
```

```
# note: sometimes when I run this (even though I have
# set.seet(123) at the top, I get 7 or 8...)
```

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

Lasso seems to have worked better for this task for a couple of reasons. First, the minimum mean squared error (MSE) is lower for Lasso than Ridge – at around 4.7 (with a lambda value of 0.002) and 5.1 (with a lambda value of 0.2) respectively. Second, the Lasso model has feature selection, with 8 predictors (depending on whether or not I set `set.seed(123)` at

the beginning of my glm() function earlier, this number falls as low as 5) selected within 1 standard error (SE) of the MSE. This is relevant to evaluating which regularized regression model worked better because having fewer features, combined with a lower MSE, sets us up for less multicolinearity or potential correlation between terms, and also reduces the risk of overfitting to too many variables. The most parsimonious model has the fewest number of predictors while staying within 1 SE of the MSE, and the Lasso model has 8 predictors while the Ridge model has 9 at a slightly higher MSE than the Lasso model at 5 predictors, thus the Lasso model is the most parsimonious for this task.

Thus, overall, Lasso is a better regularized regression model for this task because it has lower minimum MSE and MSE within 1 SE from the minimum fewer predictors – having fewer features = less multicolinearity, and more features can lead us to be at risk for more overfitting.