Data 621 Blog 3

Bryan Persaud

10/25/2020

Ridge Regression

For my third blog I will be demonstrating ridge regression. Ridge regression can be used to create a model when the number of predictors you have are more than the number of observations. It can also be used for a dataset that has multicollinearity.

Load Dataset

I will be using the mtcars dataset to demonstrate this. To also help demonstrate how to do ridge regression I will be using the glmnet package.

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.0-2
```

Ridge Regression Model

To create the model we will use the glmnet function. We start by selecting our predictors for x and y and a set of values for lambda.

```
x_variable <- data.matrix(mtcars[, c("mpg", "cyl", "qsec")])
y_variable <- mtcars$hp
lambda_values <- 10 ^ seq(2, -2, by = -0.1)
model <- glmnet(x_variable, y_variable, alpha = 0, lambda = lambda_values)
summary(model)
```

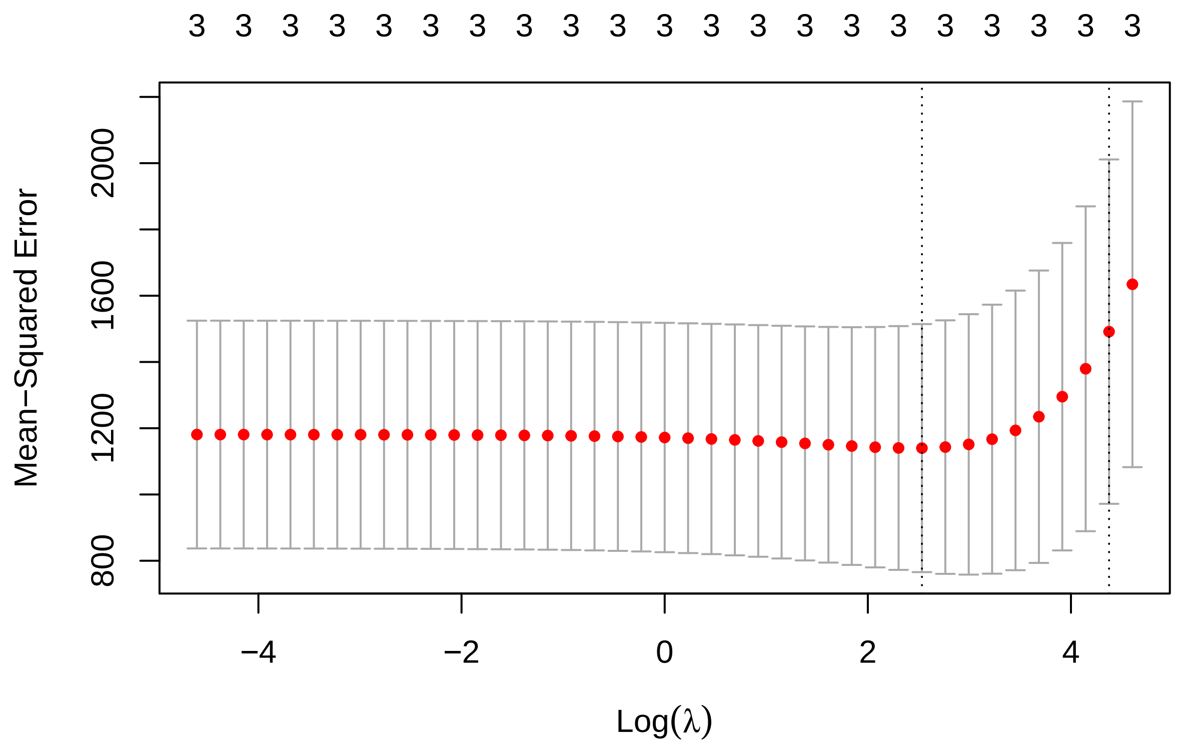
```
##           Length Class     Mode
## a0         41    -none-    numeric
## beta      123    dgCMatrix S4
## df         41    -none-    numeric
## dim         2    -none-    numeric
## lambda     41    -none-    numeric
## dev.ratio  41    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        5    -none-    call
## nobs        1    -none-    numeric
```

Ridge Regression is done by using glmnet and lambda. The model is run many times with different values of lambda. To find the best value to use for lambda we use the cv.glmnet function. This uses cross-validation to generalize how well each model works.

```
cv <- cv.glmnet(x_variable, y_variable, alpha = 0, lambda = lambda_values)
```

Let's plot this.

```
plot(cv)
```



Here we see a curve of the log values of lambda. The best value for lambda will be the lowest point in the curve. We can get this value as well as all the models.

```
lambda_best <- cv$lambda.min
lambda_best
```

```
## [1] 12.58925
```

```
model <- cv$glmnet.fit
summary(model)
```

```
##            Length Class     Mode   
## a0          41    -none-    numeric
## beta       123    dgCMatrix S4     
## df          41    -none-    numeric
## dim          2    -none-    numeric
## lambda      41    -none-    numeric
## dev.ratio   41    -none-    numeric
## nulldev      1    -none-    numeric
## npasses      1    -none-    numeric
## jerr         1    -none-    numeric
## offset       1    -none-    logical
## call         5    -none-    call   
## nobs         1    -none-    numeric
```

We can now use these to predict the data and calculate r-squared.

```
prediction <- predict(model, s = lambda_best, newx = x_variable)
sum_errors <- sum((prediction - y_variable) ^ 2)
sum_total <- sum((y_variable - mean(y_variable)) ^ 2)
r_squared <- 1 - sum_errors / sum_total
r_squared
```

```
## [1] 0.7939945
```