Hw6

Stone Cai

2024-04-11

```
#The "seatpos" data
 library(faraway)
```

```
## Warning: package 'faraway' was built under R version 4.3.2
```

```
 data(seatpos)
 pred <-data.frame(Age=64.8, Weight=263.7, HtShoes=181.08, Ht=178.56,
 Seated=91.44, Arm=35.64, Thigh=40.95, Leg=38.79)

 ##View(seatpos)
 ##install.packages('glmnet', dependencies=TRUE, type="binary")
```

Here, I take an initial glance and see n=38 with 8 predictors.

```
library(MASS)
library( dplyr)
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:MASS':
##
##     select
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.3.3
```

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

```
## Warning: package 'Matrix' was built under R version 4.3.2
```

```
## Loaded glmnet 4.1-8
```

Part a). I will look to fit a Ridge regression. In ridge regression, we select a value for lambda that produces the lowest possible test MSE (mean squared error).

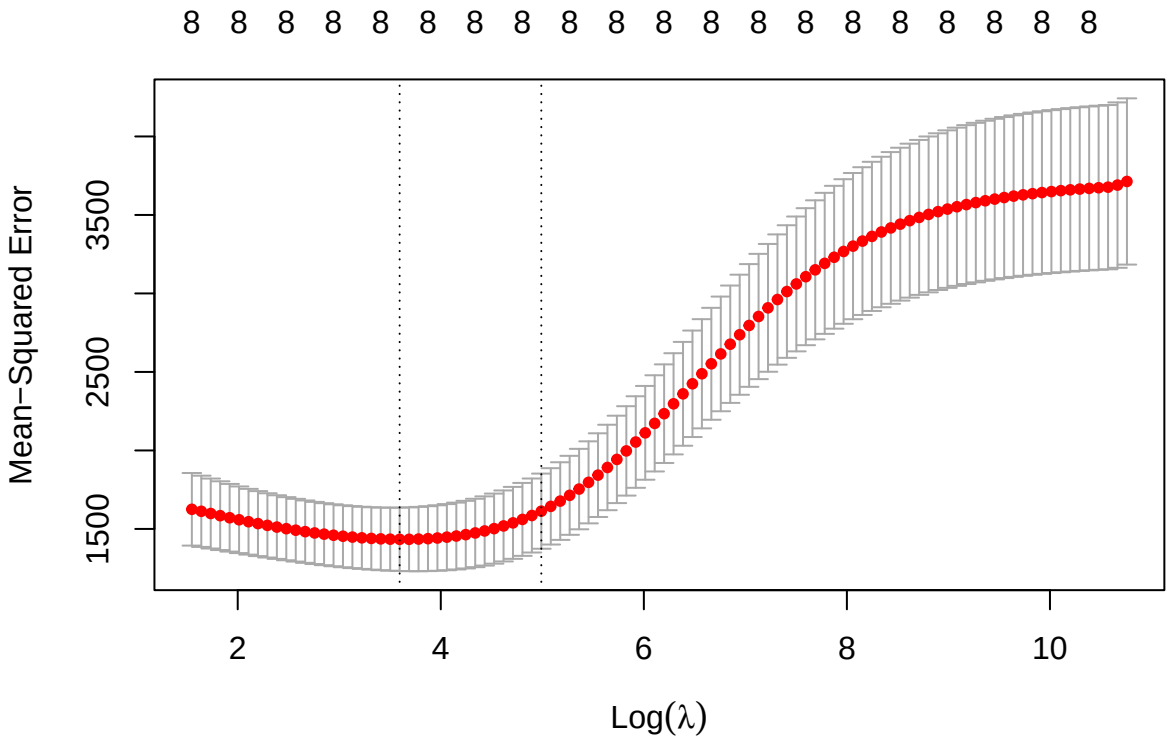
```
#define response variable
y <- seatpos$hipcenter

#define matrix of predictor variables
x <- data.matrix(seatpos[, c('Age', 'Weight', 'HtShoes', 'Ht', 'Seated', 'Arm', 'Thigh', 'Leg')])

#fit ridge regression model
model <- glmnet(x, y, alpha = 0)

#view summary of model
summary(model)
```

```
##           Length Class     Mode
## a0        100    -none-    numeric
## beta      800    dgCMatrix S4
## df        100    -none-    numeric
## dim         2    -none-    numeric
## lambda    100    -none-    numeric
## dev.ratio 100    -none-    numeric
## nulldev     1    -none-    numeric
## npasses     1    -none-    numeric
## jerr        1    -none-    numeric
## offset      1    -none-    logical
## call        4    -none-    call
## nobs        1    -none-    numeric
```

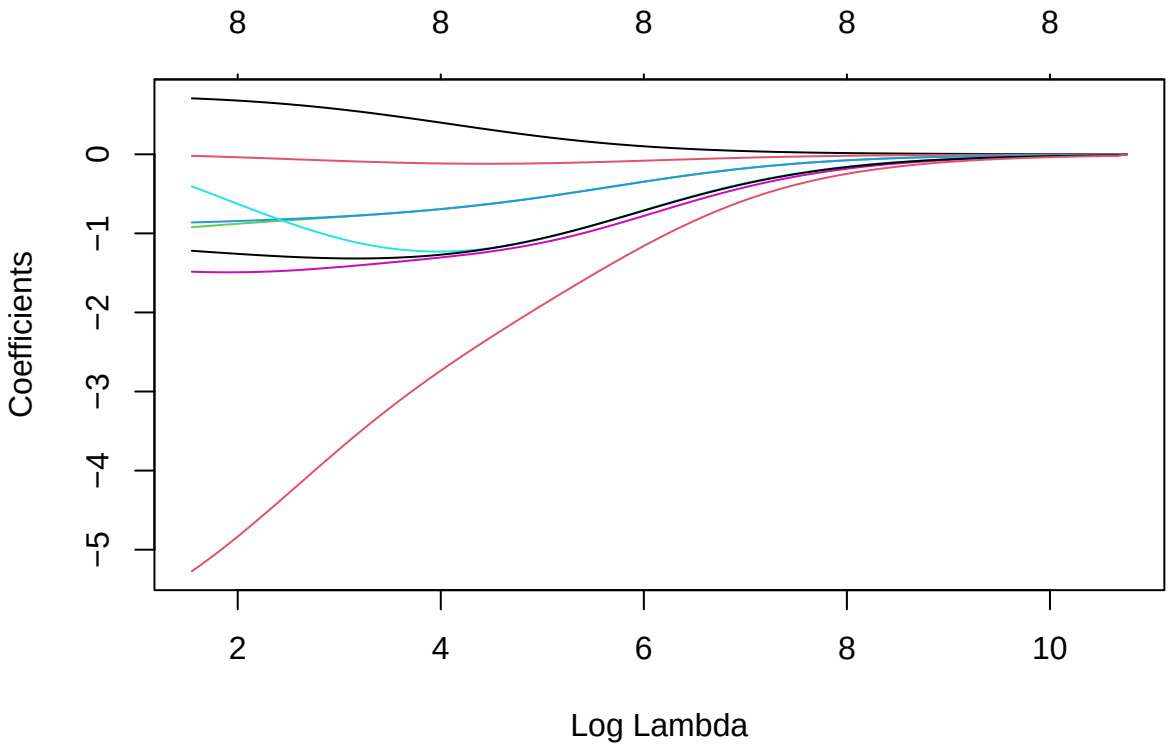
```
cv_model <- cv.glmnet(x, y, alpha = 0)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_model$lambda.min
best_lambda
```

```
## [1] 36.40799
```

```
#produce plot of test MSE by lambda value
plot(cv_model)
```




```
plot(model, xvar = "lambda")
```



First I define the reponse variable and a matrix of predictor variables. From there, I fit a ridge regression model(Note we set alpha=0 to obtain ridge model). Next, we identify the lambda value that produces the lowest test mean squared error (MSE) by using cross-validation method.Here, I also have plot of test MSE based on lambda values. I also produce a trace plot which should reflect the minimum lambda at which coefficients start to stabilize.

```
best_model <- glmnet(x, y, alpha = 0, lambda = best_lambda)
coef(best_model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 400.3627167
## Age           0.4739906
## Weight       -0.1058193
## HtShoes      -0.7389606
## Ht           -0.7384387
## Seated       -1.2039750
## Arm          -1.3560685
## Thigh        -1.3041906
## Leg          -3.1102470
```

```
y_predicted <- predict(model, s = best_lambda, newx = as.matrix(pred))
y_predicted
```

```
##             s1
## [1,] -194.9596
```

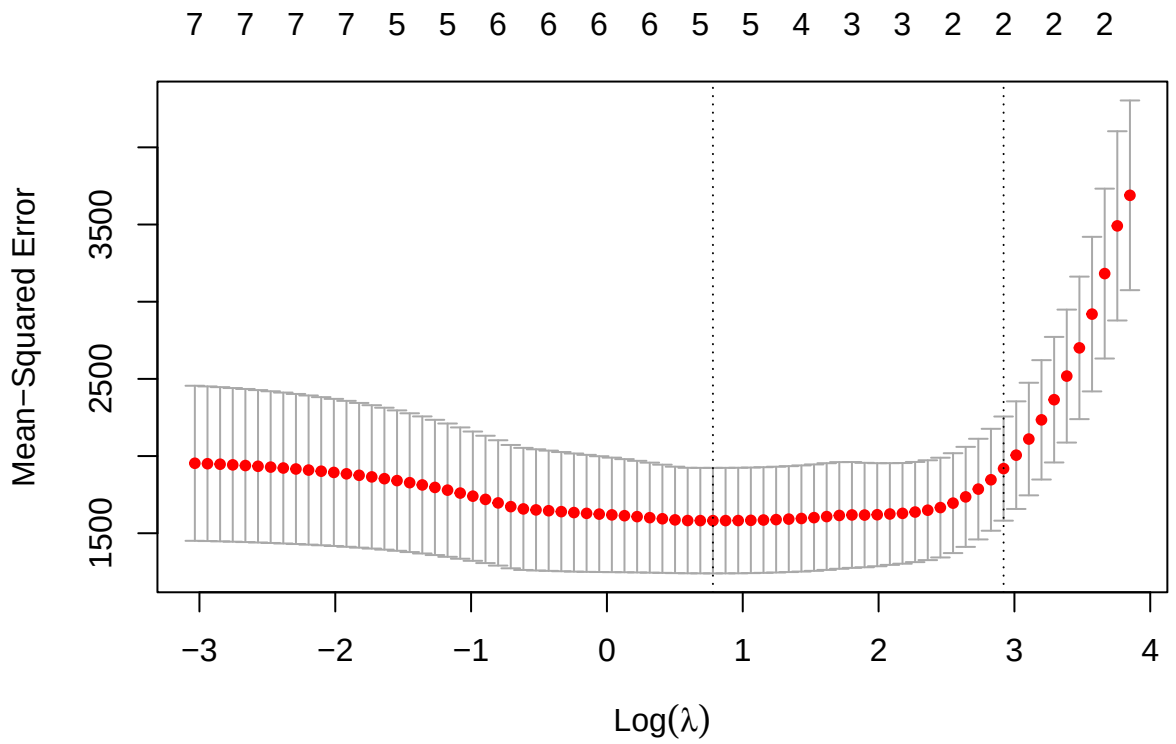
Finally,here I obtain my best model based on the best lambda I've determined from the previous step of this process. Using that lambda, we determine the regression coefficients given that lambda. Finally, I make a prediction based on the the regression coefficient of best model and the fixed data observation specified at the beginning. Here we see most of the coefficients have been shrunk toward zero.

part b) Fit a Lasso Regression. The lasso shrinkage penalty differs from the Ridge penalty (absolute value vs squared). In lasso regression, we select a value for lambda that produces the lowest possible test MSE (mean squared error).

```
#define response variable
y1 <- seatpos$hipcenter

#define matrix of predictor variables
x1 <- data.matrix(seatpos[, c('Age', 'Weight', 'HtShoes', 'Ht', 'Seated', 'Arm', 'Thigh', 'Leg')])

cv_lasso <- cv.glmnet(x1, y1, alpha = 1)

#find optimal lambda value that minimizes test MSE
best_lambda <- cv_lasso$lambda.min
best_lambda
```

```
## [1] 2.182602
```

```
#produce plot of test MSE by lambda value
plot(cv_lasso)
```



Here, I define our response variable and matrix of predictor variables. Next, we use the glmnet() function to

fit the lasso regression model and specify alpha=1.Like in ridge regression we use cross validation to identify the lambda value that produces the lowest test mean squared error (MSE). Based on the best lambda we can also plot the MSE vs log (lambda) and see where MSE is minimized.

```
best_model <- glmnet(x1, y1, alpha = 1, lambda = best_lambda)
coef(best_model)
```

```
## 9 x 1 sparse Matrix of class "dgCMatrix"
##                       s0
## (Intercept) 428.6051556
## Age           0.4889959
## Weight        .
## HtShoes      -1.5181903
## Ht           -0.5223699
## Seated        .
## Arm           .
## Thigh        -0.7862517
## Leg          -6.3925616
```

```
predict(best_model, s = best_lambda,newx = as.matrix(pred))
```

```
##             s1
## [1,] -188.0606
```

Next we fit best model based on the best lambda we obtained from previous step of process. Here we obtain the regression coefficients of the best model and notice that a few predictors are not present because the lasso method has Shrunk these predictors out of the model. This means it was completely dropped from the model because it wasn't influential enough. Ridge regression shrinks all coefficients towards zero, but lasso regression has the potential to remove predictors from the model by shrinking the coefficients completely to zero. Finally, using these remaining regression coefficients we make the same prediction for the observation specified in the problem. The predicted value here is pretty similar to the one obtained through ridge method.