Homework 6

Math 189
Spring 2023
Sophia Terenik
Deena Pederson
Gabriel Brewster

2023-05-12

Conceptual Problems

1. Because ridge regression is a shrinkage method, we would find it useful to utilize this method when the predictor variables are highly correlated with each other. By shrinking some of the coefficients to zero, the variance decreases. This will improve reliability and stability.
2. A disadvantage of ridge regression is that all of the predictors will be included. If you are looking for less variance via a subset of predictors, ridge regression will not be useful. Another time where we would not want to use ridge regression is when relationship between predictor and response is nonlinear. Ridge regression assumes a linear relationship.

Application Problems

3a.

```
library(ISLR2)
library(boot)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-7
```

```
data("Hitters")
```

```
hitters <- subset(Hitters, select = -c(League, Division, NewLeague))
hitters <- hitters[complete.cases(hitters), ]
dim(hitters)
```

```
## [1] 263  17
```

3b.

```
set.seed(123)
train_index <- sample(1:nrow(hitters), round(0.8 * nrow(hitters)))
train <- hitters[train_index, ]
test <- hitters[-train_index, ]
```

3c.

```
set.seed(123)
lm_model <- lm(Salary ~ ., data = train)
summary(lm_model)
```

```
##
## Call:
## lm(formula = Salary ~ ., data = train)
##
## Residuals:
##     Min      1Q Median      3Q     Max
## -789.6 -179.9  -36.2  140.2 1915.8
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 174.45475   94.28494   1.850  0.06580 .
## AtBat        -1.81331    0.76001  -2.386  0.01800 *
## Hits          5.12107    2.92236   1.752  0.08130 .
## HmRun        -5.02689    6.92000  -0.726  0.46846
## Runs         -1.14996    3.44698  -0.334  0.73903
## RBI           2.78032    3.00883   0.924  0.35661
## Walks         6.37394    2.13035   2.992  0.00313 **
## Years       -13.27666   14.31453  -0.927  0.35483
## CAtBat       -0.28595    0.15817  -1.808  0.07218 .
## CHits         0.75525    0.81379   0.928  0.35453
## CHmRun        1.57799    1.88372   0.838  0.40323
## CRuns         1.37818    0.90003   1.531  0.12735
## CRBI          0.22551    0.84573   0.267  0.79002
## CWalks       -0.63479    0.39721  -1.598  0.11165
## PutOuts       0.21738    0.08754   2.483  0.01388 *
## Assists       0.51573    0.25583   2.016  0.04520 *
## Errors       -6.85930    4.92600  -1.392  0.16538
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 316.8 on 193 degrees of freedom
## Multiple R-squared:  0.5819, Adjusted R-squared:  0.5472
## F-statistic: 16.79 on 16 and 193 DF,  p-value: < 2.2e-16
```

3d.

```
set.seed(123)
y_test <- test$Salary
y_pred <- predict(lm_model, newdata = test)
RMSE <- sqrt(mean((y_test - y_pred)^2))
RMSE
```

```
## [1] 373.854
```

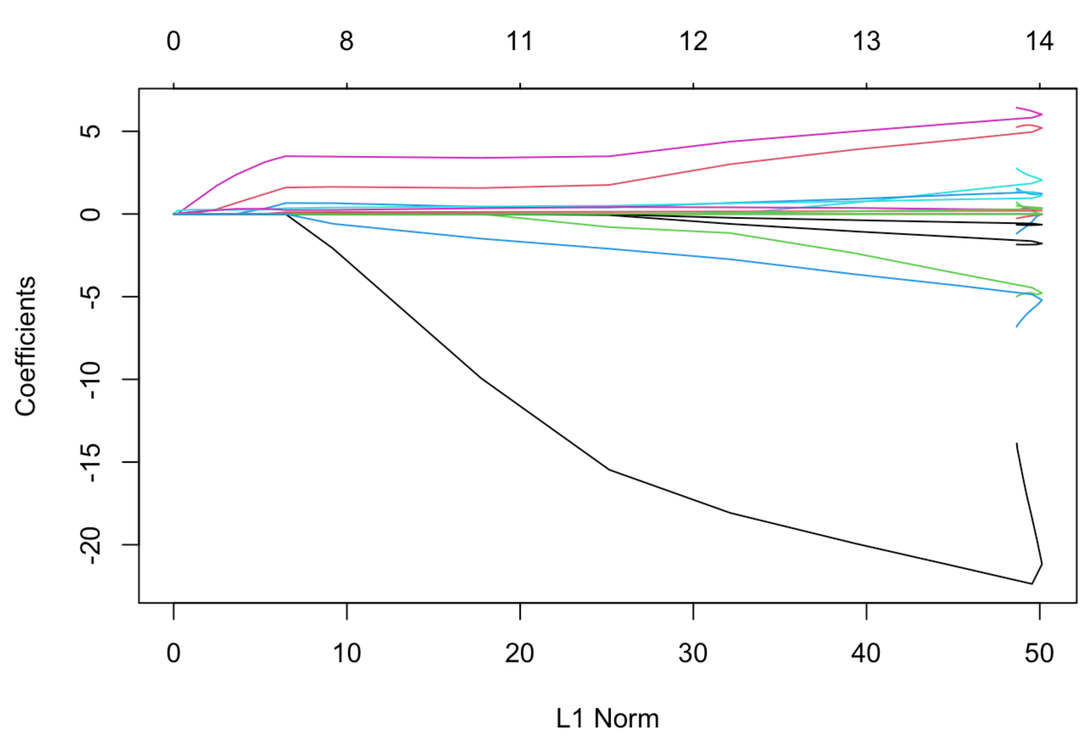
3e. We expect the RMSE to be greater than the residual standard error because RMSE is the difference in the predicted vs actual values in the test dataset whereas the residual standard error is the difference in the predicted vs actual values in the training dataset.

4a.

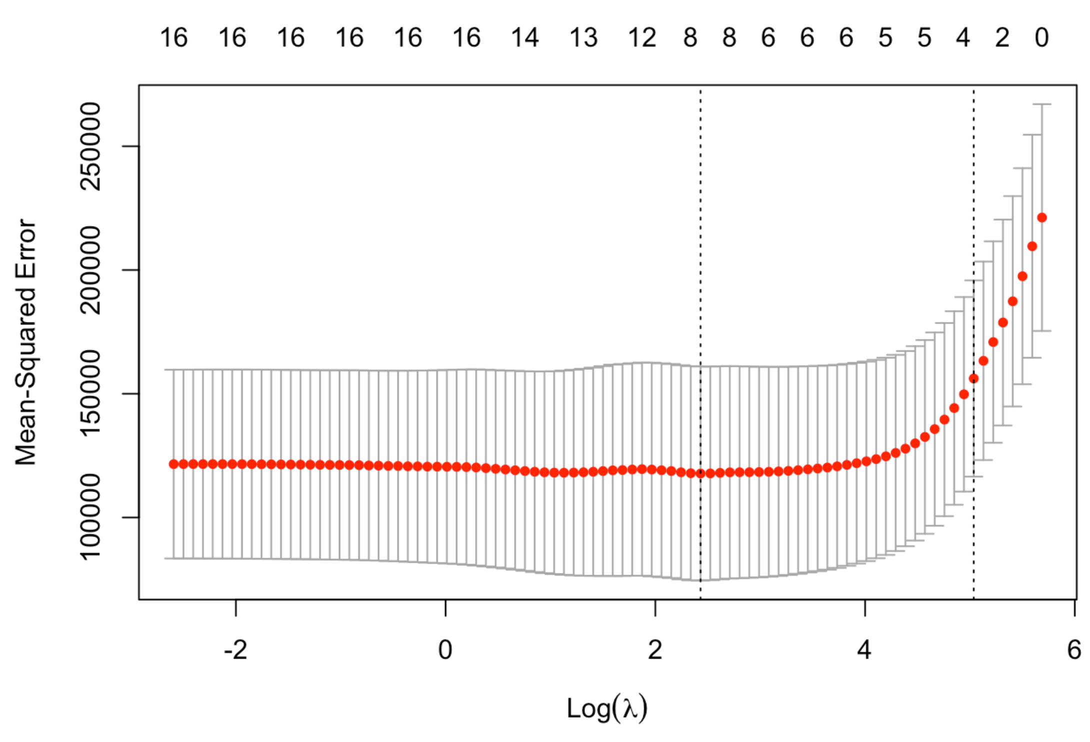
```
x <- model.matrix(Salary ~ ., hitters)[, -1]
y <- hitters$Salary
```

```
grid <- 10^seq(10, -2, length = 100)
lasso.mod <- glmnet(x[train_index, ], y[train_index], alpha = 1, lambda = grid)
plot(lasso.mod)
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
set.seed(123)
cv.out <- cv.glmnet(x[train_index, ], y[train_index], alpha = 1)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 11.3676
```

We first choose to implement the lasso model over a grid values ranging from lambda = 10^10 to 10^-2. We then use cross-validation to choose lambda. The value of lambda that results in the smallest cross-validation error is 11.

4b.

```
out <- glmnet(x, y, alpha = 1, lambda = grid)
lasso.coef <- predict(out, type = "coefficients", s = bestlam)[1:17, ]
lasso.coef
```

```
##   (Intercept)        AtBat         Hits        HmRun         Runs          RBI
## -57.89789801   0.00000000   2.01966953   0.00000000   0.00000000   0.00000000
##         Walks        Years       CAtBat        CHits       CHmRun        CRuns
##    2.31756395   0.00000000   0.00000000   0.00000000   0.07145521   0.25558599
##          CRBI       CWalks      PutOuts      Assists       Errors
##    0.36007290   0.00000000   0.23686740   0.00000000  -0.50557651
```

When LASSO zeros out a variable, it is removing it from the model by making the coefficient estimate zero. The idea behind LASSO is that selects a subset of variables that are most important. By zeroing out coefficients, LASSO has deemed that variable to be less relevant, thus removing it from the model. 4c.

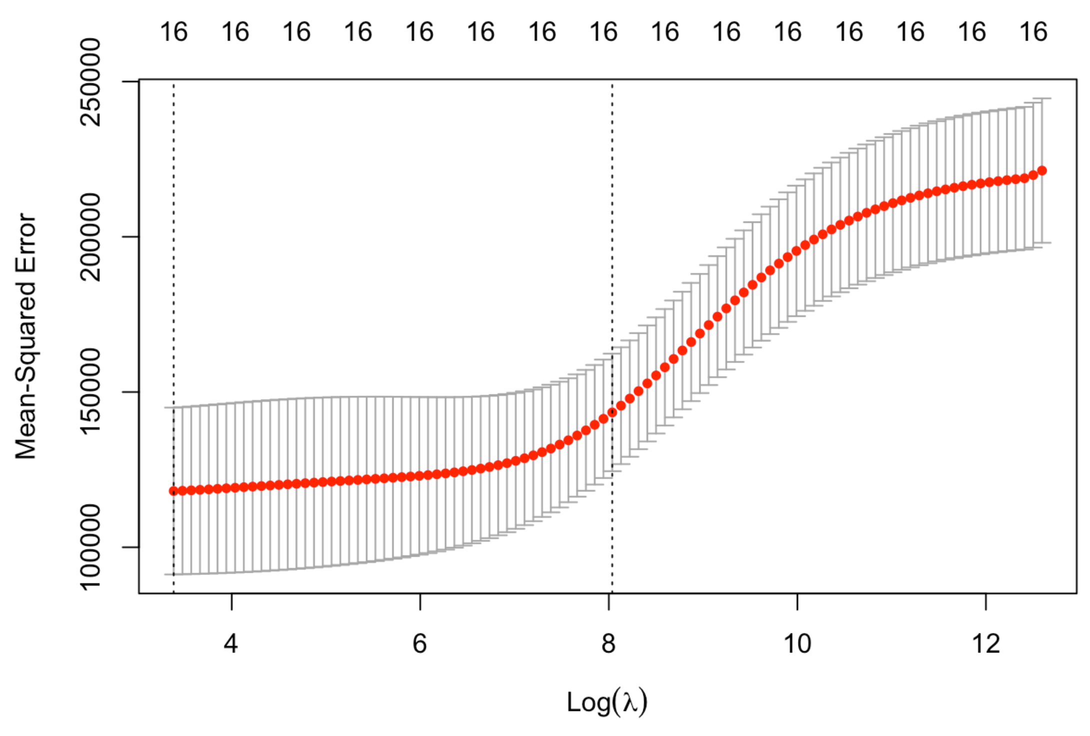
```
bestlam <- cv.out$lambda.min
lasso.pred <- predict(lasso.mod, s = bestlam, newx = x[-train_index, ])
sqrt(mean((lasso.pred - y_test)^2))
```

```
## [1] 355.9074
```

5a.

```
grid <- 10^seq(10, -2, length = 100)
ridge.mod <- glmnet(x, y, alpha = 0, lambda = grid)
```

```
set.seed(1)
cv.out <- cv.glmnet(x[train_index, ], y[train_index], alpha = 0)
plot(cv.out)
```




```
bestlam <- cv.out$lambda.min
bestlam
```

```
## [1] 29.49919
```

We again found the best lambda to use by cross validation and finding the minimum value.

5b.

```
coef(ridge.mod)[, 17]
```

```
##   (Intercept)         AtBat          Hits         HmRun          Runs
##  5.359091e+02  4.734346e-06  1.717362e-05  6.920002e-05  2.904181e-05
##           RBI         Walks         Years        CAtBat         CHits
##  3.067723e-05  3.610528e-05  1.476545e-04  4.064882e-07  1.495993e-06
##        CHmRun         CRuns          CRBI        CWalks       PutOuts
##  1.128183e-05  3.001290e-06  3.097394e-06  3.277009e-06  1.896273e-06
##       Assists        Errors
##  3.097293e-07 -1.444248e-06
```

The coefficients vary much less than the linear model in 3c. They are all much closer in value to 0, whereas the coefficients from 3c range from around -13 to 6.
5c.

```
ridge.pred = predict(ridge.mod, s = bestlam, newx = x[-train_index, ])
sqrt(mean((ridge.pred - y_test)^2))
```

```
## [1] 307.5453
```

6. We think that the LASSO model is the most useful in this scenario. The hitters dataset has many variables, some of which are less relevant than others. LASSO preforms shrinkage, effectively zeroing out less important variables. This allows the general manager to focus on the most significant variables in predicting a players salary.

We worked together and each team member did equal work on the homework.