Lab Week 8 : Feature Selection

STAT5003

Dr. Justin Wishart

Semester 2, 2022

Contents

Preparation and assumed knowledge

- Viewed the feature selection data content in Module 8.
- Installed the `glmnet` package on your system.
- Installed the `ISLR` package on your system.

Aims

- Exploring feature selection - subset selection, ridge regression and the lasso.

1 Feature Selection

The dataset we are using is one provided by the ISLR package. It contains Major League Baseball Data from the 1986 and 1987 seasons with 322 observations with 20 variables. It has some missing cases that isn't the focus of this module and can be removed with the following code.

```
data(Hitters, package = "ISLR")
Hitters <- na.omit(Hitters)
```

1.1 Forward selection

Implement a forward stepwise selection procedure to find the best five features. The response feature here is `Salary` which is numeric, so this is a regression and not a classification problem.

```
selectFeatureDirect <- function(train, test, cls.train, cls.test, features) {
  ## identify a feature to be selected
  current.smallest.mse <- Inf
  selected.i <- NULL
  for(i in 1:ncol(train)) {
    current.f <- colnames(train)[i]

    # Can't add features that are already in our list
    if(current.f %in% c(features, "Salary")) { next }
    model <- lm(data=cbind(train[, c(features, current.f, "Salary")]), Salary ~ .)
```

```r
    # Calculate the mean squared error
    test.mse <- mean((cls.test - predict.lm(model, test[,c(features, current.f, "Salary")])) ^ 2)

    if(test.mse < current.smallest.mse) {
        current.smallest.mse <- test.mse
        selected.i <- colnames(train)[i]
    }
  }
  selected.i
}

library(caret)
```

```
## Loading required package: ggplot2
## Loading required package: lattice
```

```r
set.seed(1)
inTrain <- createDataPartition(Hitters$Salary, p = .6)[[1]]

# Remove the column with the Salary
train <- Hitters[ inTrain,]
test  <- Hitters[-inTrain,]
salary.train <- Hitters$Salary[inTrain]
salary.test <- Hitters$Salary[-inTrain]

features.direct <- NULL
current.min.mse <- Inf

# Find the top five features
for (i in 1:5) {
  selected.i <- selectFeatureDirect(train, test, salary.train, salary.test, features.direct)
  print(selected.i)

  # add the best feature from current run
  features.direct <- c(features.direct, selected.i)
}
```

```
## [1] "CHmRun"
## [1] "Hits"
## [1] "Division"
## [1] "PutOuts"
## [1] "League"
```

```r
# Top five features selected
print(features.direct)
```

```
## [1] "CHmRun"   "Hits"     "Division" "PutOuts"  "League"
```

```r
## Alternatively, can implement a forward selection procedure using an indirect evaluation tool such as
selectFeatureIndirect <- function(features) {
  ## identify a feature to be selected
  new.features <- setdiff(names(Hitters), c(features, "Salary"))
  BICs <- sapply(new.features, function(f) AIC(lm(Salary ~ ., data = Hitters[, c(features, "Salary", f)]
  names(which.min(BICs))
}
```

```r
features.indirect <- NULL
for (i in 1:5) {
  selected.i <- selectFeatureIndirect(features.indirect)
  print(features.indirect)

  # add the best feature from current run
  features.indirect <- c(features.indirect, selected.i)
}
```

```
## NULL
## [1] "CRBI"
## [1] "CRBI" "Hits"
## [1] "CRBI"    "Hits"    "PutOuts"
## [1] "CRBI"     "Hits"     "PutOuts"  "Division"
```

```r
print(features.indirect)
```

```
## [1] "CRBI"     "Hits"     "PutOuts"  "Division" "AtBat"
```

1.2 Lasso regression

1.2.1 Split data and fit

Split the dataset into 60% train and 40% test. Perform a lasso regression using the `glmnet` package.

```r
library(glmnet)
```

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

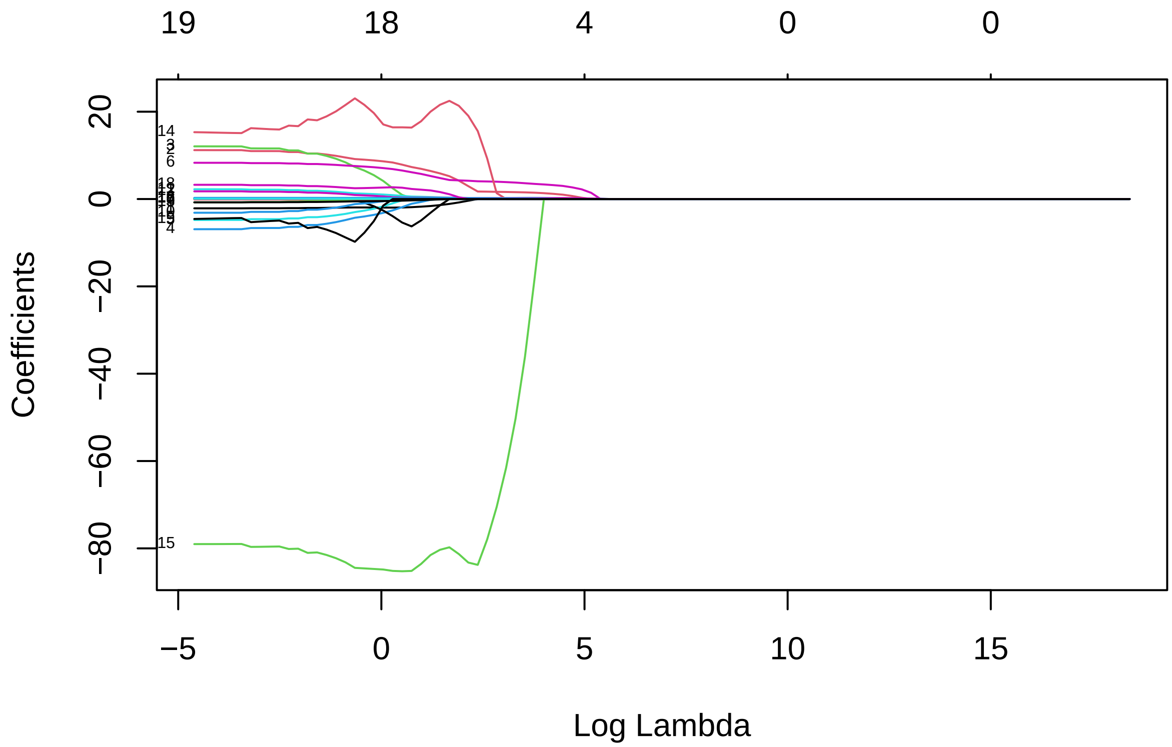
```
## Loaded glmnet 4.1-4
```

```r
set.seed(1)
inTrain <- createDataPartition(Hitters$Salary, p = .6)[[1]]

# Convert to design matrix format
x <- model.matrix(Salary ~ ., Hitters)[,-1]
y <- Hitters$Salary
```

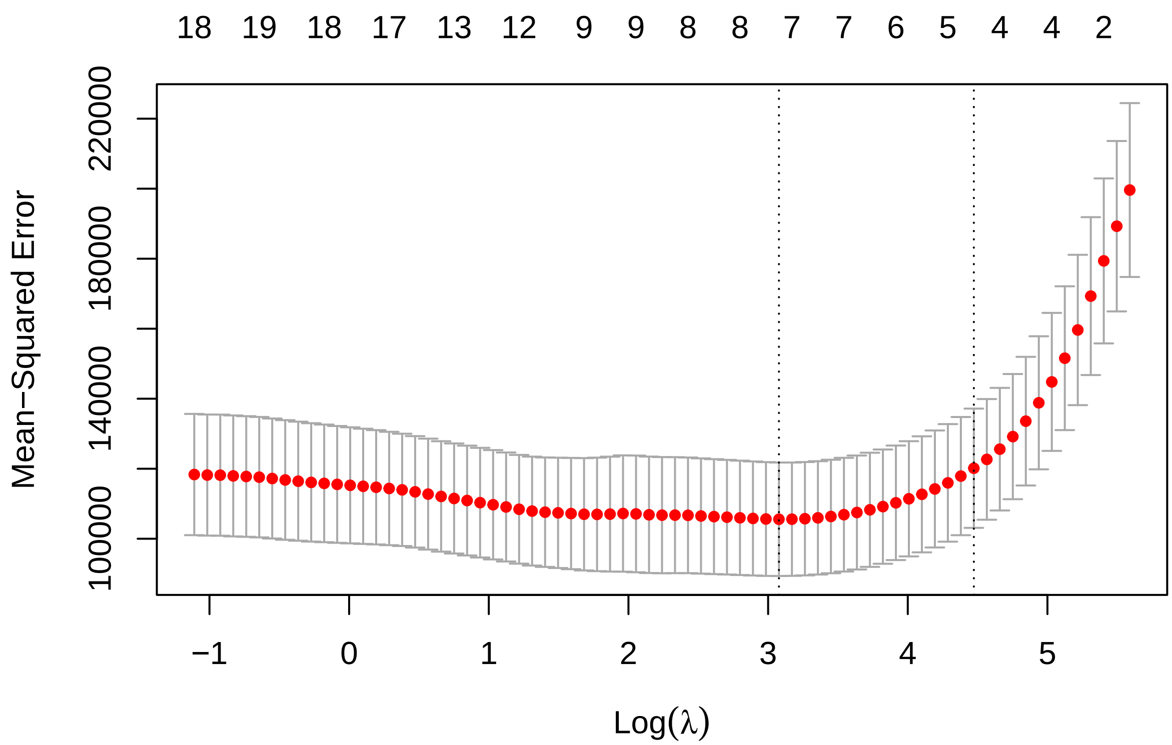
1.2.2 Inspect coefficients as function of λ

Plot the lasso regression coefficients as a function of λ. Use cross-validation to find the best λ value (You can use the inbuilt CV function in glmnet).

```r
# set the range of lambda values to be tested.
grid <- 10^seq(8,-2, length=100)

# Set alpha = 1 for Lasso
lasso.mod <- glmnet(x[inTrain,], y[inTrain], alpha=1, lambda=grid, standardize = TRUE)
plot(lasso.mod, xvar="lambda", label=TRUE)
```




```r
# Using the inbuilt function to do cross-validation
set.seed(1)
cv.out <- cv.glmnet(x[inTrain,], y[inTrain], alpha=1)
plot(cv.out)
```




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

```
## [1] 21.70636
```

1.2.3 Check if model is sparse

Inspect the coefficients in the best model above and check if there there are any coefficients that have been shrunk to zero.

```r
# Extract the features for the best fit
best.betas <- cv.out$glmnet.fit$beta[,which(cv.out$lambda == bestlam)]
best.betas
```

```
##       AtBat        Hits       HmRun        Runs         RBI       Walks
##  0.00000000  1.61713182  0.00000000  0.00000000  0.00000000  3.87700197
##       Years      CAtBat       CHits      CHmRun       CRuns        CRBI
##  0.00000000  0.00000000  0.04217819  0.00000000  0.23439012  0.23247202
##      CWalks     LeagueN   DivisionW     PutOuts     Assists      Errors
##  0.00000000  0.00000000 -61.21683826  0.18358716  0.00000000  0.00000000
##  NewLeagueN
##  0.00000000
```

```r
# Shrunken betas (zero)
shrunk.betas <- best.betas[best.betas == 0]
shrunk.betas
```

```
##     AtBat     HmRun      Runs       RBI     Years    CAtBat    CHmRun
##         0         0         0         0         0         0         0
##    CWalks   LeagueN   Assists    Errors NewLeagueN
##         0         0         0         0         0
```

```r
# Non-zero betas
non.zero.betas <- best.betas[best.betas != 0]
non.zero.betas
```

```
##        Hits       Walks       CHits       CRuns        CRBI   DivisionW
##  1.61713182  3.87700197  0.04217819  0.23439012  0.23247202 -61.21683826
##     PutOuts
##  0.18358716
```

1.2.4 Assess on the test set

Using the optimal trained model predict the salary of the test dataset and calculate the mean squared error.

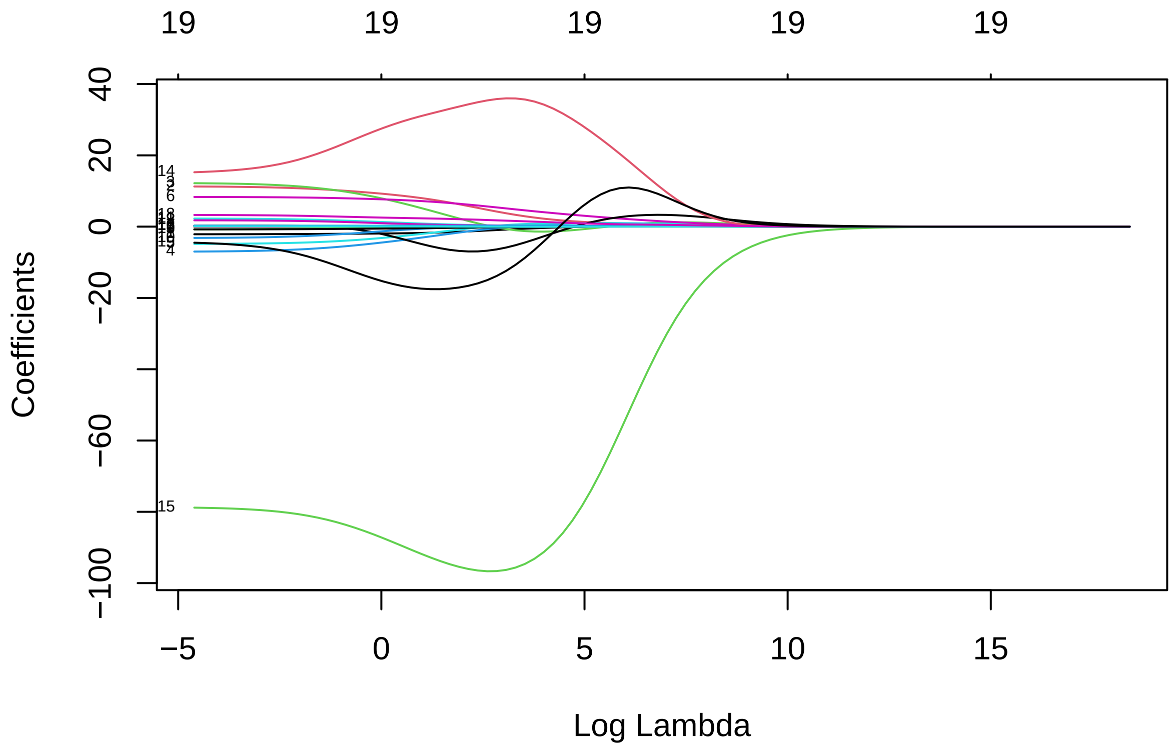
```r
# Use the best CV model to predict the test set Salary
test.pred <- predict(cv.out, x[-inTrain,])[,1]
mse.lasso <- mean((test.pred - y[-inTrain])^2)
print(mse.lasso)
```

```
## [1] 145130.3
```

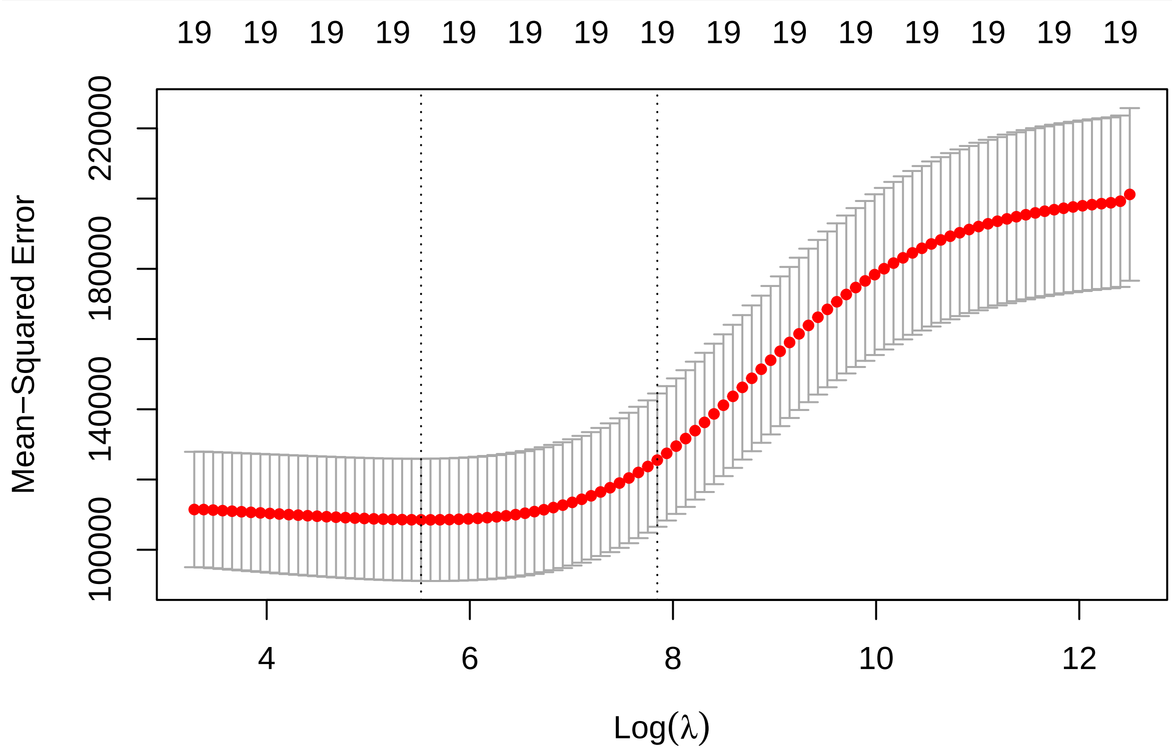
1.3 Optional Ridge regression

Repeat the prevous question with the Ridge. Perform ridge regression with the `glmnet` package on the dataset.

```r
ridge.mod <- glmnet(x[inTrain,], y[inTrain], alpha=0, lambda=grid, standardize = TRUE)
plot(ridge.mod, xvar="lambda", label=TRUE)
```




```r
# Using the inbuilt function to do cross-validation
set.seed(1)
cv.out <- cv.glmnet(x[inTrain,], y[inTrain], alpha=0)
plot(cv.out)
```




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

```
## [1] 249.5704
```

```r
# Use the best CV model to predict the test set Salary
test.pred <- predict(cv.out, x[-inTrain,])[,1]
mse <- mean((test.pred - y[-inTrain])^2)
```