3a – Aniket Maheshwari

UB Person Number: 50360266

Setting up our environment and importing important libraries:

```
### Clear the environment
rm(list = ls())


### First we will set the directory of the R script
setwd("C:/Users/anike/Desktop/Sem 1/EAS 506 Statistical Data
Mining/Homework/Homework 3")


## Loading all the libraries
library(ISLR)
library(corrplot)

## corrplot 0.90 loaded

library(MASS)
library(klaR)
library(leaps)
library(lattice)
library(ggplot2)
library(corrplot)
library(car)

## Loading required package: carData

library(caret)
library(class)
```

Part a)

In this question, I need to create my own simulated dataset of matrix 1000*20.

```
set.seed(1)
X <- rnorm(1000 * 20)
X <- matrix(X,1000,20)
colnames(X) <- paste("X", 1:20 , sep = "")
X[1:5,1:5]

##              X1          X2          X3         X4         X5
## [1,] -0.6264538  1.13496509 -0.88614959  0.7391149 -1.1346302
## [2,]  0.1836433  1.11193185 -1.92225490  0.3866087  0.7645571
## [3,] -0.8356286 -0.87077763  1.61970074  1.2963972  0.5707101
```

```
## [4,]  1.5952808  0.21073159  0.51926990 -0.8035584 -1.3516939
## [5,]  0.3295078  0.06939565 -0.05584993 -1.6026257 -2.0298855
```

Now, I'll add Beta values to this matrix. I'll made 4 of the beta values zero so that i can make some estimations later on about my model . I've made my beta = 4,7,12 and 16 as zero. I'll Also create epsilon (Noise) value so that i don't create a perfect model but i'll scale this so that i don't have the same scale as my data.

```
set.seed(1)
beta <- runif(20)
beta[c(4,7,12,16)] = 0



set.seed(1)
epsilon <- 0.001 * rnorm(1000)
```

Now, I'll create my Y (response variable) as Y = X*Beta +epsilon

```
Y <- X%*%beta + epsilon
length(Y)

## [1] 1000
```

Merging both the X and Y to get the full dataset:

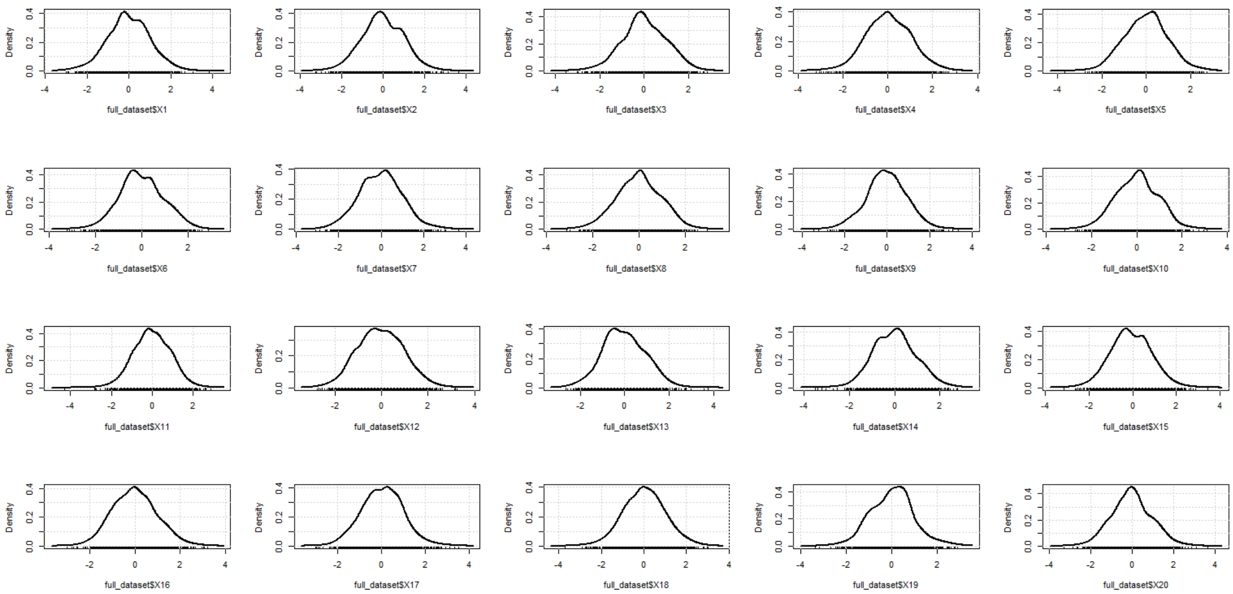
```
full_dataset <- data.frame(X,Y)
dim(full_dataset)

## [1] 1000   21
```

Density Plots:

```
x11()
par(mfrow=c(4,5))
densityPlot(full_dataset$X1)
densityPlot(full_dataset$X2)
densityPlot(full_dataset$X3)
densityPlot(full_dataset$X4)
densityPlot(full_dataset$X5)
densityPlot(full_dataset$X6)
densityPlot(full_dataset$X7)
densityPlot(full_dataset$X8)
densityPlot(full_dataset$X9)
densityPlot(full_dataset$X10)
densityPlot(full_dataset$X11)
densityPlot(full_dataset$X12)
densityPlot(full_dataset$X13)
densityPlot(full_dataset$X14)
densityPlot(full_dataset$X15)
densityPlot(full_dataset$X16)
densityPlot(full_dataset$X17)
densityPlot(full_dataset$X18)
```

```
densityPlot(full_dataset$X19)
densityPlot(full_dataset$X20)
```



The density plot of the data tells that all the classes in our data are in normal distribution.

Part b)

Splitting the dataset in test and train dataset:

I'll split my data in 10:90 ratio that is after the splitting my train set will have 100 observation and test set will have 900 observations.

```
train_index = sample(1:nrow(full_dataset) , nrow(full_dataset)*.1)
train_data <- full_dataset[train_index, ]
test_data <- full_dataset[-train_index, ]
dim(test_data)

## [1] 900  21

dim(train_data)

## [1] 100  21

y.test = test_data$Y
y.train = train_data$Y
```

Part c)

Best Subset Selection: Now I'll perform best subset selection to see which is the best model. As i made 4 beta's value zero, i should get 16 variable model as the best model.

```
dataset_best_subset_selection <- regsubsets(Y ~ . , data = train_data , nbest
= 1,  really.big = TRUE , nvmax = 21)
dataset_best_subset_selection_sum <- summary(dataset_best_subset_selection)

dataset_best_subset_selection_sum

## Subset selection object
## Call: regsubsets.formula(Y ~ ., data = train_data, nbest = 1, really.big =
TRUE,
##     nvmax = 21)
## 20 Variables  (and intercept)
##     Forced in Forced out
## X1      FALSE      FALSE
## X2      FALSE      FALSE
## X3      FALSE      FALSE
## X4      FALSE      FALSE
## X5      FALSE      FALSE
## X6      FALSE      FALSE
## X7      FALSE      FALSE
## X8      FALSE      FALSE
## X9      FALSE      FALSE
## X10     FALSE      FALSE
## X11     FALSE      FALSE
## X12     FALSE      FALSE
## X13     FALSE      FALSE
## X14     FALSE      FALSE
## X15     FALSE      FALSE
## X16     FALSE      FALSE
## X17     FALSE      FALSE
## X18     FALSE      FALSE
## X19     FALSE      FALSE
## X20     FALSE      FALSE
## 1 subsets of each size up to 20
## Selection Algorithm: exhaustive
##           X1  X2  X3  X4  X5  X6  X7  X8  X9  X10 X11 X12 X13 X14 X15 X16
X17
## 1  ( 1 )  " " " " " " " " " " " " " " " " " " " " " " " " "*" " " " " " "
" "
## 2  ( 1 )  " " " " " " " " " " "*" " " " " " " " " " " " " "*" " " " " " "
" "
## 3  ( 1 )  " " " " " " " " " " "*" " " " " " " " " " " " " "*" " " "*" " "
" "
## 4  ( 1 )  " " " " " " " " " " "*" " " " " " " " " " " " " "*" " " "*" " "
" "
## 5  ( 1 )  " " " " " " " " " " "*" " " " " " " " " " " " " "*" " " "*" " "
```

```
"*"
## 6  ( 1 )  " " " " " " " " " " "*" " " " " " " " " " " " " "*" " " "*" " "
"*"
## 7  ( 1 )  " " " " " " " " " " "*" " " "*" " " " " " " " " "*" " " "*" " "
"*"
## 8  ( 1 )  " " " " "*" " " " " "*" " " "*" " " " " " " " " "*" " " "*" " "
"*"
## 9  ( 1 )  " " " " "*" " " " " "*" " " "*" "*" " " " " " " "*" " " "*" " "
"*"
## 10  ( 1 ) "*" " " "*" " " " " "*" " " "*" "*" " " " " " " "*" " " "*" " "
"*"
## 11  ( 1 ) " " " " "*" " " " " "*" " " "*" "*" " " " " " " "*" "*" "*" " "
"*"
## 12  ( 1 ) " " "*" "*" " " " " "*" " " "*" "*" " " " " " " "*" "*" "*" " "
"*"
## 13  ( 1 ) "*" "*" "*" " " " " "*" " " "*" "*" " " " " " " "*" "*" "*" " "
"*"
## 14  ( 1 ) "*" "*" "*" " " " " "*" " " "*" "*" " " "*" " " "*" "*" "*" " "
"*"
## 15  ( 1 ) "*" "*" "*" " " "*" "*" " " "*" "*" " " "*" " " "*" "*" "*" " "
"*"
## 16  ( 1 ) "*" "*" "*" " " "*" "*" " " "*" "*" "*" "*" " " "*" "*" "*" " "
"*"
## 17  ( 1 ) "*" "*" "*" "*" "*" "*" " " "*" "*" "*" "*" " " "*" "*" "*" " "
"*"
## 18  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" " " "*" "*" "*" " "
"*"
## 19  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" " " "*" "*" "*" "*"
"*"
## 20  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"
"*"
##           X18 X19 X20
## 1  ( 1 )  " " " " " "
## 2  ( 1 )  " " " " " "
## 3  ( 1 )  " " " " " "
## 4  ( 1 )  "*" " " " "
## 5  ( 1 )  "*" " " " "
## 6  ( 1 )  "*" " " "*"
## 7  ( 1 )  "*" " " "*"
## 8  ( 1 )  "*" " " "*"
## 9  ( 1 )  "*" " " "*"
## 10  ( 1 ) "*" " " "*"
## 11  ( 1 ) "*" "*" "*"
## 12  ( 1 ) "*" "*" "*"
## 13  ( 1 ) "*" "*" "*"
## 14  ( 1 ) "*" "*" "*"
## 15  ( 1 ) "*" "*" "*"
## 16  ( 1 ) "*" "*" "*"
## 17  ( 1 ) "*" "*" "*"
## 18  ( 1 ) "*" "*" "*"
```

```
## 19  ( 1 ) "*" "*" "*"
## 20  ( 1 ) "*" "*" "*"
```

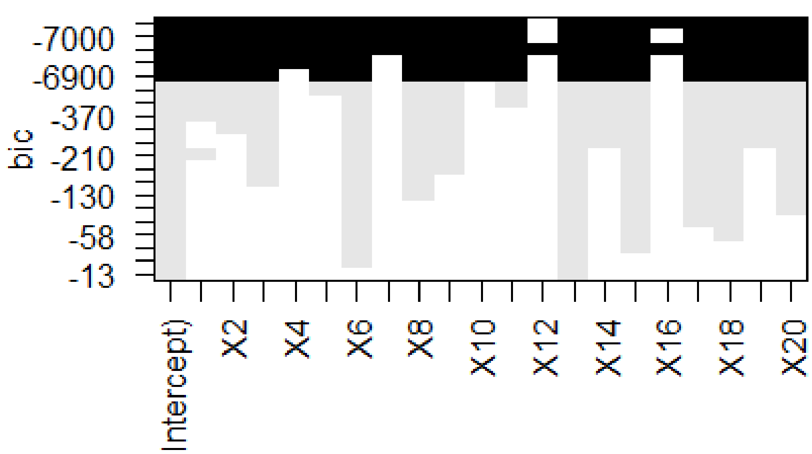
Now if i look at the best subset selection, i made my beta's zero for 4 classes that were 4,7,12 and 16, So the best 16 variable model should exclude those classes for me.

```
coef(dataset_best_subset_selection , 16)

##  (Intercept)           X1           X2           X3           X5
X6
## 3.205769e-15 2.665087e-01 3.721239e-01 5.728534e-01 2.016819e-01
8.983897e-01
##           X8           X9          X10          X11          X13
X14
## 6.607978e-01 6.291140e-01 6.178627e-02 2.059746e-01 6.870228e-01
3.841037e-01
##          X15          X17          X18          X19          X20
## 7.698414e-01 7.176185e-01 9.919061e-01 3.800352e-01 7.774452e-01
```

So our model is fitted perfectly. The best 16 variable model excludes 4,7,12 and 16.

```
plot(dataset_best_subset_selection, scale = "bic")
```



The BIC plot also has the class X4,X7,X12 and X16 as the lowest.

Fitting out model on train set:

```
predict.regsubsets = function(object,newdata,id , ...){
  form = as.formula((object$call[[2]]))
  mat = model.matrix(form, newdata)
  coefi = coef(object , id = id)
  xvars = names(coefi)
  mat[,xvars]%*%coefi

}

training_error_value <- matrix(rep(NA,20))
y_true_train = train_data$Y


for (i in 1:20){
  training_pred = predict(dataset_best_subset_selection , newdata =
train_data , id = i )
  training_error_value[i] =  (1/length(y_true_train)) * sum ((y_true_train -
training_pred ) ^ 2) #MSE training error
}
training_error_value
```

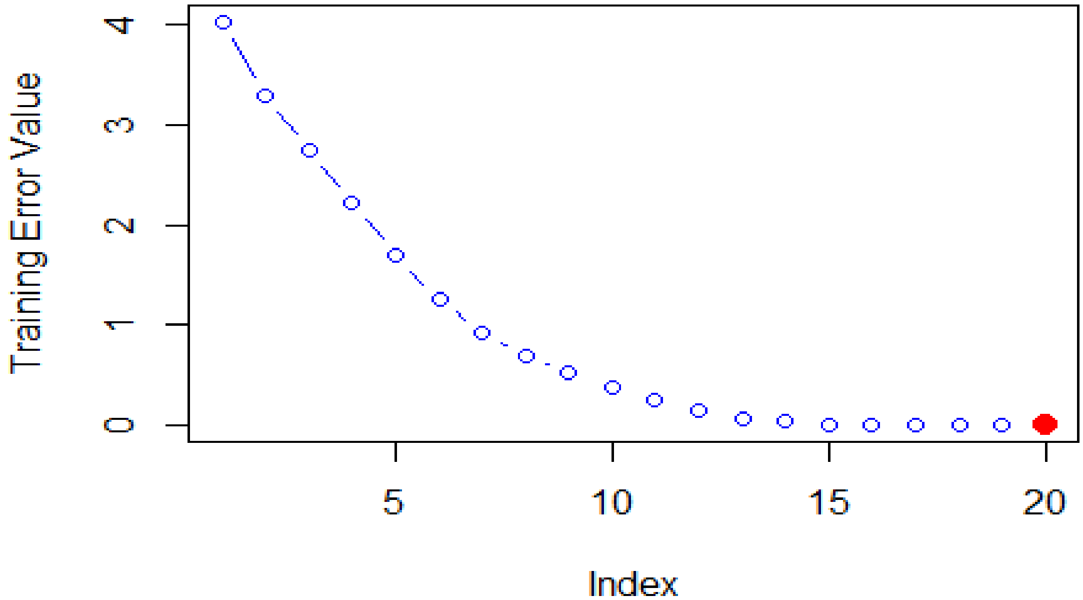
```
##               [,1]
##  [1,] 4.027980e+00
##  [2,] 3.300227e+00
##  [3,] 2.737178e+00
##  [4,] 2.228156e+00
##  [5,] 1.692807e+00
##  [6,] 1.259145e+00
##  [7,] 9.116538e-01
##  [8,] 6.864932e-01
##  [9,] 5.119322e-01
## [10,] 3.783410e-01
## [11,] 2.579054e-01
## [12,] 1.421895e-01
## [13,] 6.836531e-02
## [14,] 3.571408e-02
## [15,] 3.095459e-03
## [16,] 5.872195e-28
## [17,] 5.891018e-28
## [18,] 5.979518e-28
## [19,] 5.975569e-28
## [20,] 5.953250e-28
```

Plotting the training set MSE associated with the best model of each size:

```
plot(training_error_value, col= "blue" , type = "b" , ylab = "Training Error
Value")
which.min(training_error_value)
```

```
## [1] 16
```

```
points(20, training_error_value[20],col="red" , cex = 2 , pch = 20)
```



Part D)

Fitting out model on Test set:

```
testing_error_value <- matrix(rep(NA,20))
y_true_test = test_data$Y
for (i in 1:20){
  testing_pred = predict(dataset_best_subset_selection , newdata = test_data
, id = i )
  testing_error_value[i] = (1/length(y_true_test)) * sum ((y_true_test -
testing_pred ) ^ 2) # MSE testing error
}
testing_error_value

##               [,1]
##  [1,] 5.209411e+00
##  [2,] 4.275635e+00
##  [3,] 3.876153e+00
##  [4,] 3.143943e+00
##  [5,] 2.666414e+00
##  [6,] 2.128168e+00
##  [7,] 1.543290e+00
##  [8,] 1.194799e+00
##  [9,] 7.189567e-01
## [10,] 6.672265e-01
```

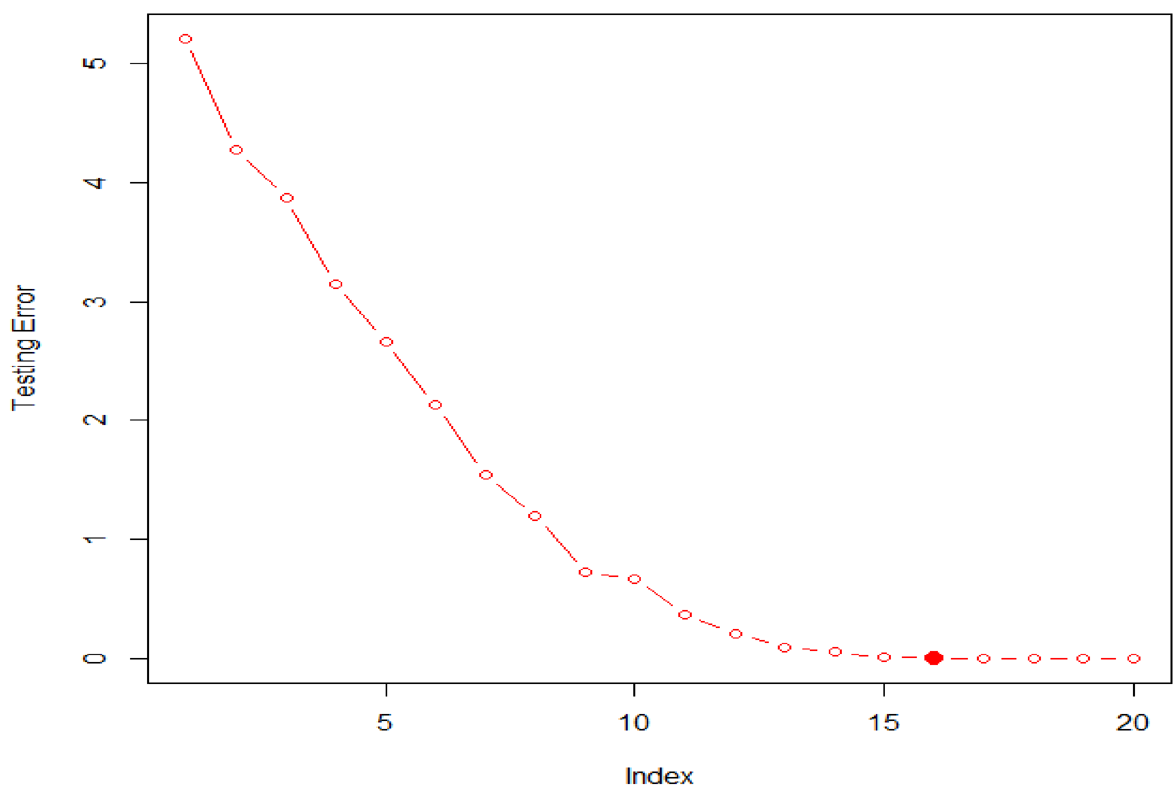
```
## [11,] 3.688628e-01
## [12,] 2.042332e-01
## [13,] 9.173711e-02
## [14,] 4.885612e-02
## [15,] 4.136706e-03
## [16,] 7.087129e-28
## [17,] 7.189867e-28
## [18,] 7.556354e-28
## [19,] 7.619103e-28
## [20,] 7.533194e-28
```

Plotting the test set MSE associated with the best model of each size:

```
plot(testing_error_value , col= "red" , type = "b")
which.min(testing_error_value)

## [1] 16

points(16, testing_error_value[20],col="red" , cex = 2 , pch = 20)
```



Part E)

```
which.min(testing_error_value)

## [1] 16
```

We have our lowest testing error for model with 16 variables. Our the best fit selection has worked perfectly on the test data because I took beta values zero for 4 variables (4,7,12,16) and best fit model gives the lowest testing error for model with 16 variable excluding those four variables.

Part F)

The best fit model gives 16 variable model as the lowest test set MSE which compare to the true model used to generate the data is correct.

```
coef(dataset_best_subset_selection , 16)

##  (Intercept)           X1           X2           X3           X5 
X6 
## 3.205769e-15 2.665087e-01 3.721239e-01 5.728534e-01 2.016819e-01 
8.983897e-01 
##           X8           X9          X10          X11          X13 
X14 
## 6.607978e-01 6.291140e-01 6.178627e-02 2.059746e-01 6.870228e-01 
3.841037e-01 
##          X15          X17          X18          X19          X20 
## 7.698414e-01 7.176185e-01 9.919061e-01 3.800352e-01 7.774452e-01
```

All the features are positively co-related to our response feature.

Part G)

Creating a plot displaying:

for a range of values, r, where Bj is the jTH coefficient estimate for the best model containing r coefficients.

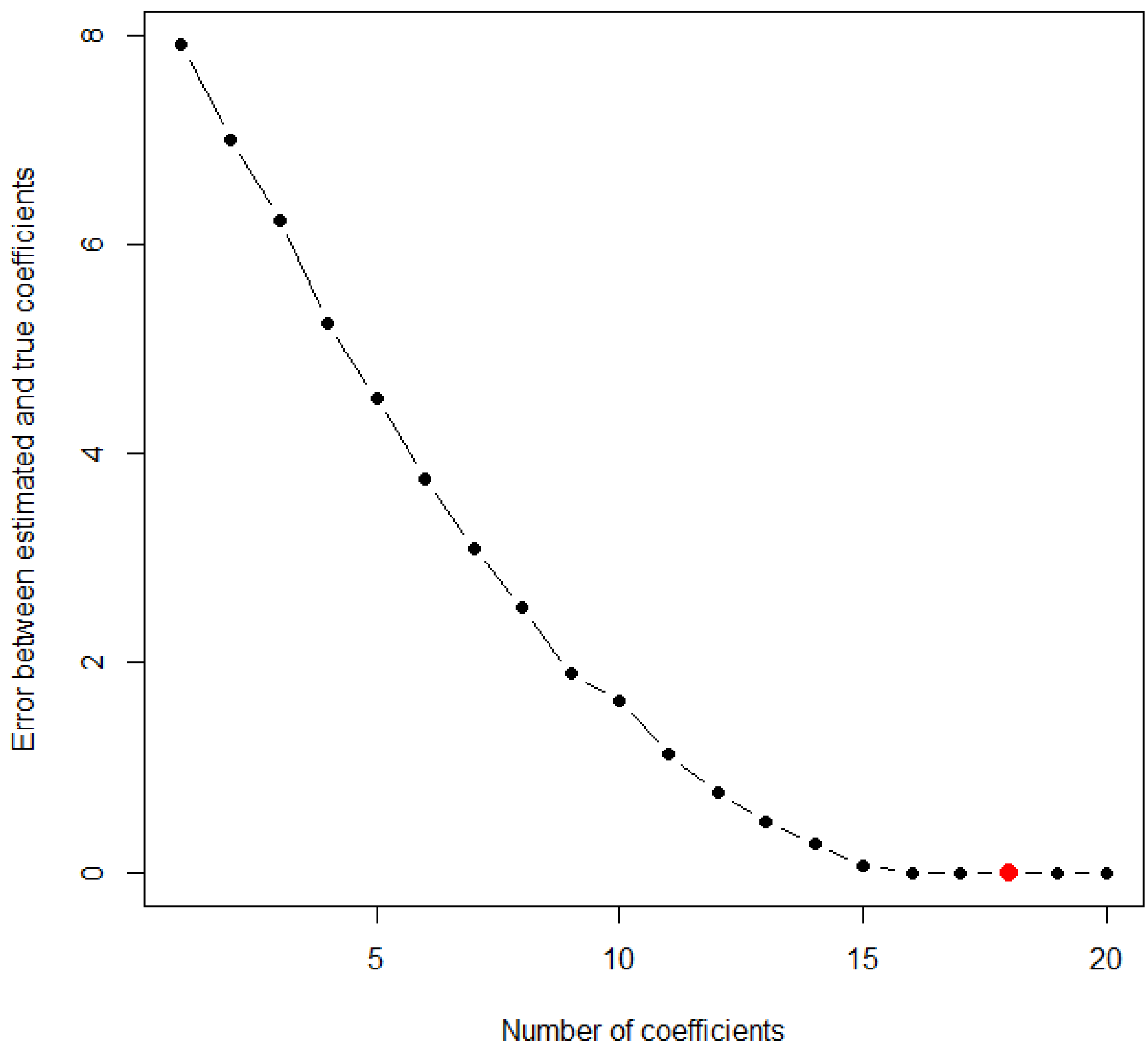
```
val.errors <- rep(NA, 20)
x_cols = colnames(X, do.NULL = FALSE, prefix = "X")
for (i in 1:20) {
  coefi <- coef(dataset_best_subset_selection, id = i)
  val.errors[i] <- sqrt(sum((beta[x_cols %in% names(coefi)] - 
coefi[names(coefi) %in% x_cols])^2) + sum(beta[!(x_cols %in% 
names(coefi))])^2)
}

val.errors
```

```
##  [1] 7.90250787 6.99897360 6.23179430 5.24994962 4.53468301 3.76297211
##  [7] 3.09536818 2.52410638 1.89669651 1.64350380 1.13355682 0.76372523
## [13] 0.47877631 0.27240952 0.06538332 0.00100000 0.00100000 0.00100000
## [19] 0.00100000 0.00100000
```

Plotting the error plot:

```
plot(val.errors, xlab = "Number of coefficients", ylab = "Error between
estimated and true coefficients", pch = 19, type = "b")
which.min(val.errors)
points(18, testing_error_value[18],col="red" , cex = 2 , pch = 20)
```



Here, as compared to the test MSE error, 18 variable model has the lowest error value. Although it does have the same decreasing plot as the test MSE error.