Homework 2 Report

Akshay Adlakha (akshayad)

Problem 1:

In this problem, we are given a dataset of College in the package ISLR. We need to predict number of applications received using the other variables.

The dataset is split into 75% training and 25% testing set. Training model is created using linear regression.

```
> model <- lm(Apps ~ ., data = collegeData.train)
> summary(model)

Call:
lm(formula = Apps ~ ., data = collegeData.train)

Residuals:
    Min      1Q  Median      3Q     Max
-3262.8  -434.0   -60.1   325.9  6553.2

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) -357.51636  455.03029  -0.786 0.432374
PrivateYes  -634.41730  157.19688  -4.036 6.19e-05 ***
Accept         1.25726    0.05664  22.198  < 2e-16 ***
Enroll        -0.27787    0.20084  -1.384 0.167038
Top10perc     46.23226    6.02057   7.679 7.12e-14 ***
Top25perc    -14.28980    4.92667  -2.900 0.003871 **
F.Undergrad    0.09259    0.03576   2.589 0.009872 **
P.Undergrad    0.01370    0.03441   0.398 0.690564
Outstate      -0.04842    0.02072  -2.337 0.019785 *
Room.Board     0.17243    0.05135   3.358 0.000839 ***
Books          0.17278    0.27453   0.629 0.529365
Personal      -0.02580    0.07087  -0.364 0.715956
PhD           -7.50832    5.18958  -1.447 0.148507
Terminal      -4.40205    5.53844  -0.795 0.427054
S.F.Ratio      0.30257   14.07008   0.022 0.982851
perc.alumni   -7.16154    4.53523  -1.579 0.114876
Expend         0.07847    0.01319   5.950 4.72e-09 ***
Grad.Rate     10.13370    3.24004   3.128 0.001853 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Used the model to the testing set to check accuracy. Here is the mean squared error.

```
> pred <- predict(model,collegeData.test)
> mean((pred - collegeData.test$Apps)^2)
[1] 1783326
>
```

Fit a ridge regression model on the training dataset. Here is the mean squared error.

```
> train.matrix <- model.matrix(Apps ~ ., data = collegeData.train)
> test.matrix <- model.matrix(Apps ~ ., data = collegeData.test)
> grid = 10^seq(4, -2, length = 100)
>
> model.ridge <- glmnet(train.matrix, collegeData.train$Apps, alpha = 0, lambda$
> cv.ridge <- cv.glmnet(train.matrix, collegeData.train$Apps, alpha = 0, lambda$
> lambda.ridge <-  cv.ridge$lambda.min
> lambda.ridge
[1] 21.54435
>
> # Fiiting a ridge model on testing set.
> predict.ridge <- predict(model.ridge, s = lambda.ridge, newx = test.matrix)
> mean((predict.ridge - collegeData.test$Apps)^2)
[1] 1943866
>
```

The Mean squared error for Ridge regression is higher than a least squared model.

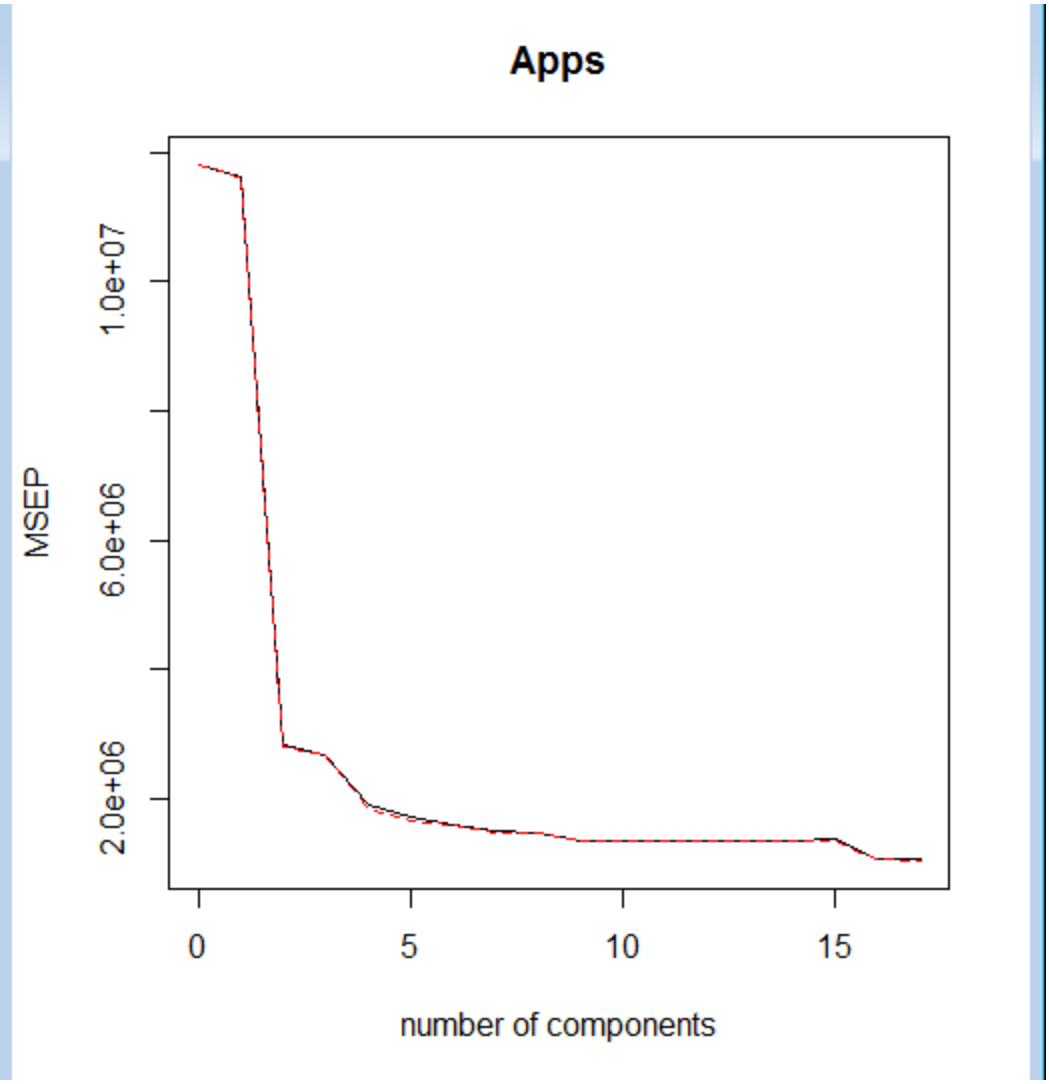
Used a lasso model on the training dataset. Here is the mean squared error.

```
> model.lasso <- glmnet(train.matrix, collegeData.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
> cv.lasso <- cv.glmnet(train.matrix, collegeData.train$Apps, alpha = 1, lambda = grid, thresh = 1e-12)
> lambda.lasso <- cv.lasso$lambda.min
> lambda.lasso
[1] 10.72267
>
> # Fitting a testing model on testing set.
> predict.lasso <- predict(model.lasso, s = lambda.lasso, newx = test.matrix)
> mean((predict.lasso - collegeData.test$Apps)^2)
[1] 1873485
>
```

The Mean squared error for Lasso regression is also higher than a least squared model.

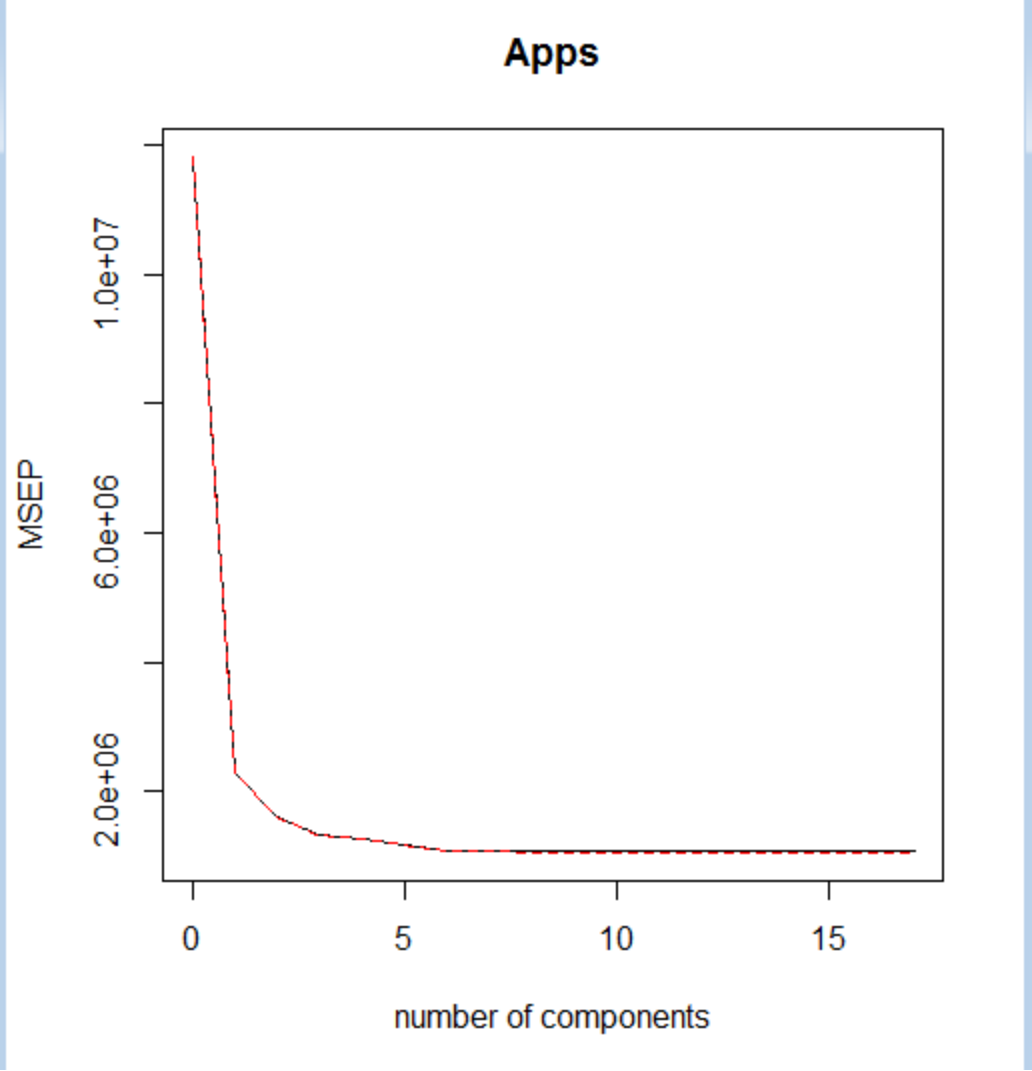
Then, the PCR model is created. Here is the mean squared error.

```
> # creating a PCR model.
> model.pcr <- pcr(Apps ~ ., data = collegeData.train, scale = TRUE, validation = "CV")
> validationplot(model.pcr, val.type = "MSEP")
>
> #fitting a PCR model on testing set.
> predict.pcr <- predict(model.pcr, collegeData.test, ncomp = 10)
> mean((predict.pcr - collegeData.test$Apps)^2)
[1] 4466547
>
```



Then, A PLS model is created. Here is the result.

```
>
> # fitting a PLS model on testing set
> predict.pls <- predict(model.pls, collegeData.test, ncomp = 10)
> mean((predict.pls - collegeData.test$Apps)^2)
[1] 1791860
>
```

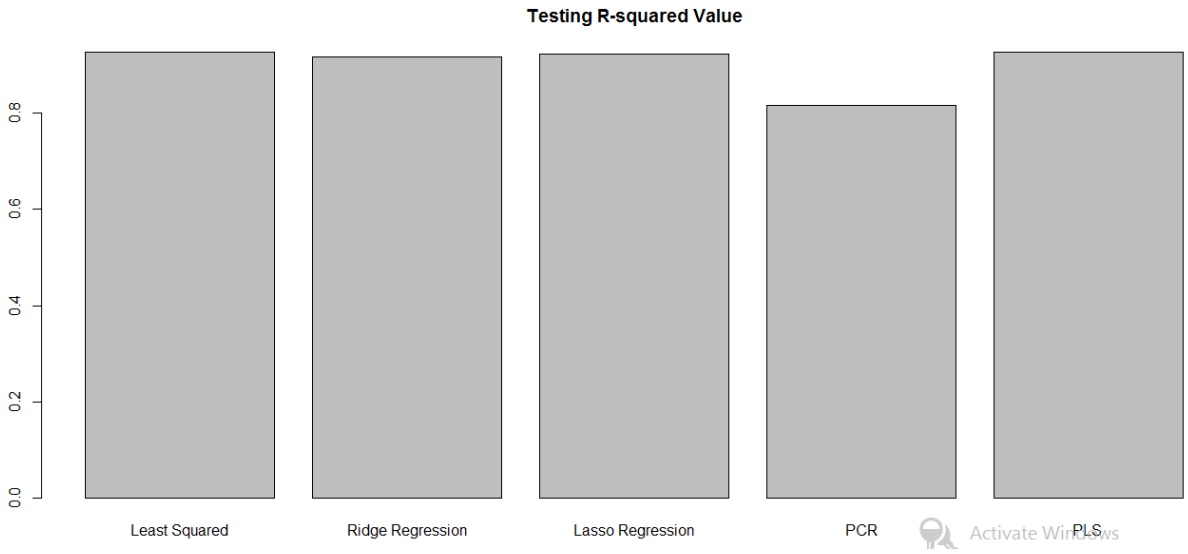


Here is the comparison table:

```
> colnames(erroData) <- c("Linear","Ridge","Lasso","PCR","PLS")
> rownames(erroData) <- c("R Squared")
> erroData <- as.table(erroData)
> erroData
             Linear     Ridge     Lasso       PCR       PLS
R Squared 0.9267440 0.9169385 0.9229332 0.8165218 0.9263934
> 
```

As we can see that all models are predicting the application number with high accuracy having value more than 0.9 except for PCR model.

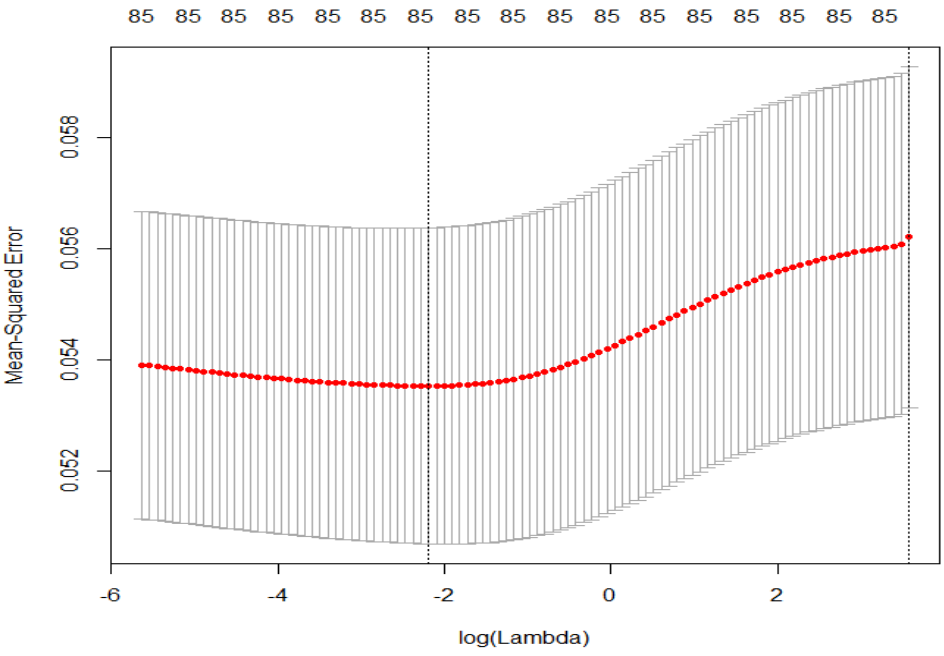
Here is the visual representation of this data:



Problem 2:

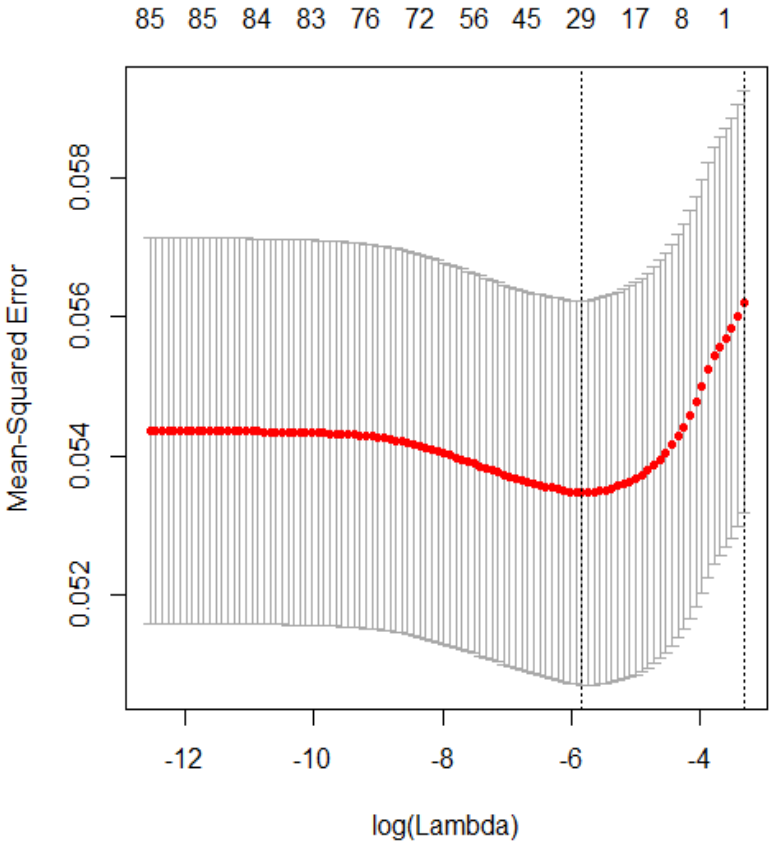
In this problem, we are given a dataset and we need to compute OLS estimates and compare them with Forward selection, backward selection, Ridge and Lasso Regression.

After doing data exploration and data cleaning, a Ridge model is created. Here are the results:



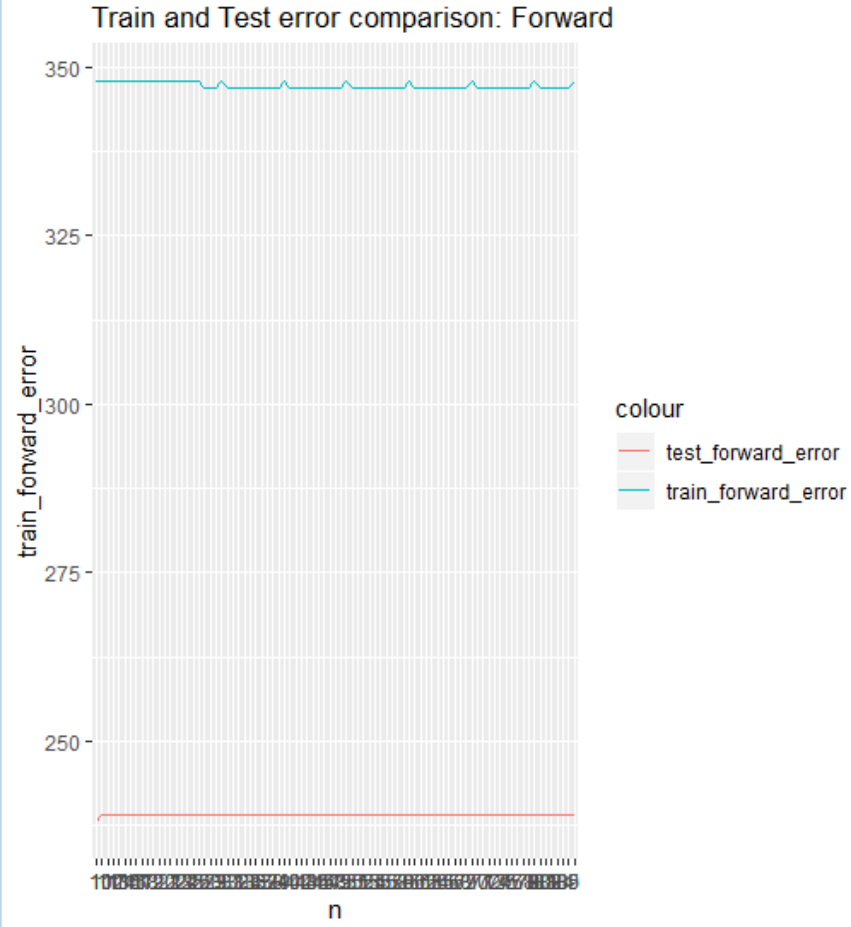
The mean squared error is increasing with the increase in lambda which indicates that the shrinkage of coefficients has been too much as is leading to under-fitting.

A Lasso model's graph as cross validation variation.

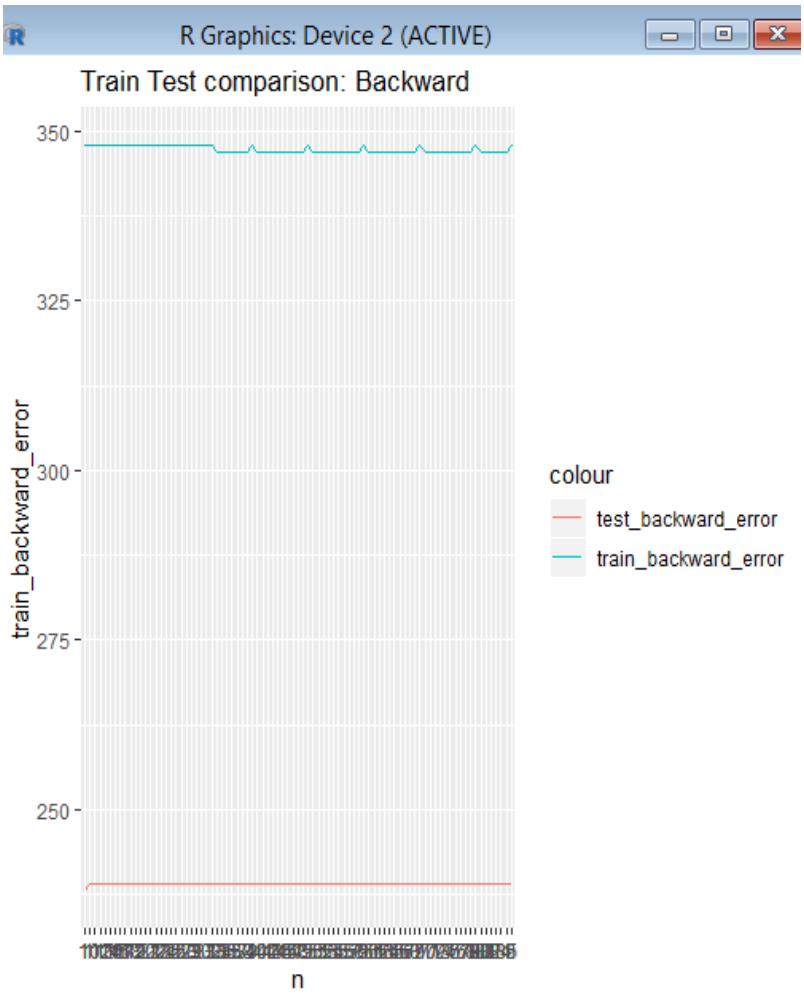


The mean squared error is increasing with the increase in lambda.

Here are the result for forward selection.

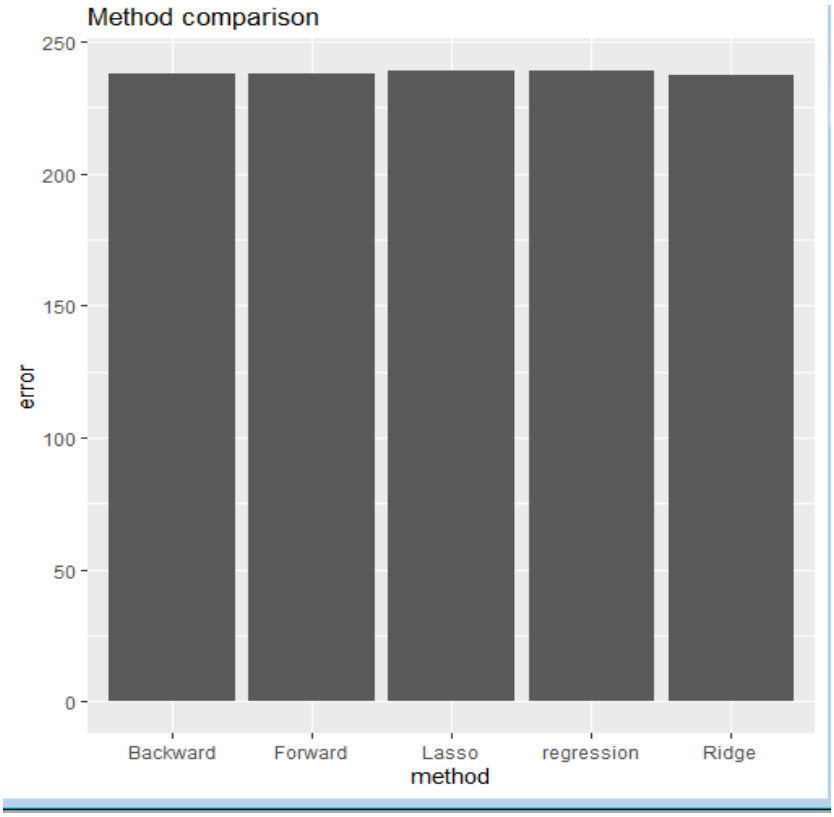


Below is the result for backward selection.



There is no improvement in the predictions for any i for both forward and backward selection.

Then, compared the results of all models created. Here is the plot for comparison.



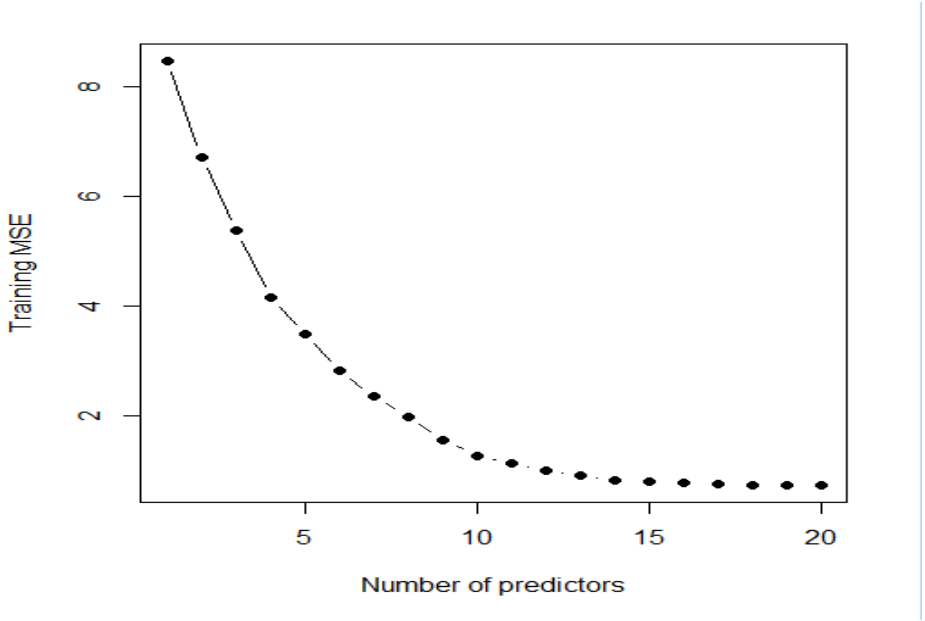
From this graph, we can infer that all the modes are not that good in predicting. So it is difficult to predict who is interested in buying an insurance policy.

Problem 3:

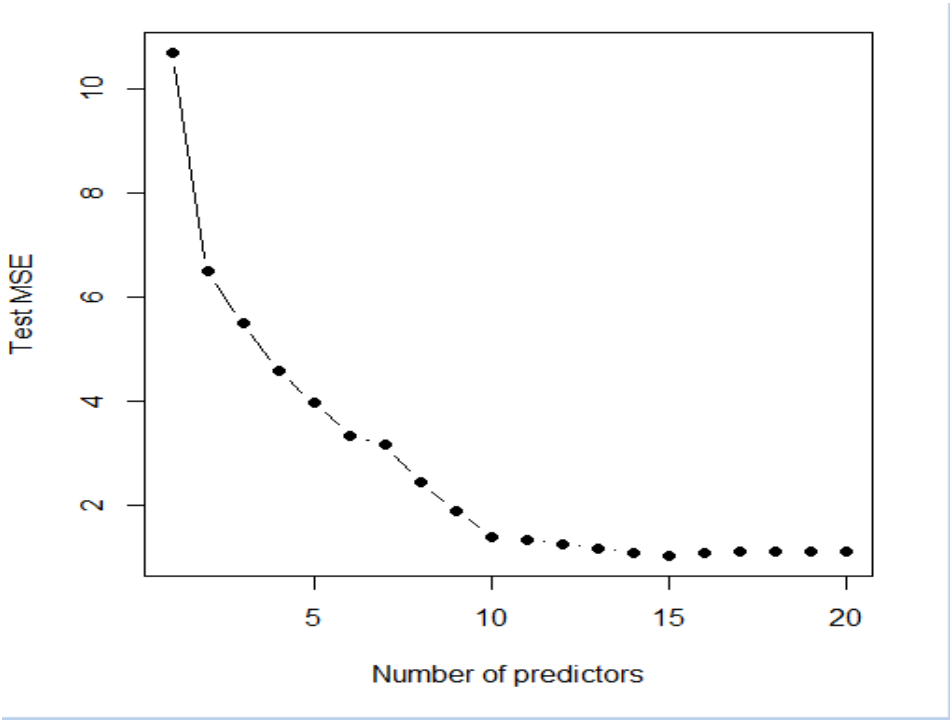
In this problem, we need to generate a dataset with 20 features, 1000 observations and the model Y= $X\beta + e$, where β has some random elements that are exactly zero.

After data is generated and divided into training set and testing set, performed best subset selection on the training and testing data and plotted the MSE associated with the best model of each size.

Here is the result for training set:



The testing set result:



The test error decreases till 10 predictors and then stops decreasing.

```
> which.min(val.errors)
[1] 15
> coef(regfit.full,which.min(val.errors))
(Intercept)         x.1         x.3         x.5         x.6         x.8         x.9        x.10        x.11
-0.00397492  0.38841654 -0.75946924  0.93010711 -0.19732357  0.71540881  1.87283059  0.73441435  0.83109671
       x.12        x.13        x.14        x.16        x.17        x.18        x.20
 0.75887948 -0.30728504 -0.64592799 -0.36304012  0.31466069  1.55082110 -0.89532248
>
```

The MSE minimum on the testing set is for 15-variable model. I used 2, 4, 7, 14 and 19 variable with zero value. From the above result, we can say that the best model caught all zeroed out coefficients.