1.) In this question almost every model follows the same pattern. We will predict the line using training data for different models. Then we will predict the y hat value using test data and the respective line formed by the models. And finally we calculate the Mean Square Error using the Y-true value and Y-hat value

(a)

Splitted the data set into training set and test set.

The dimension of the training set is:

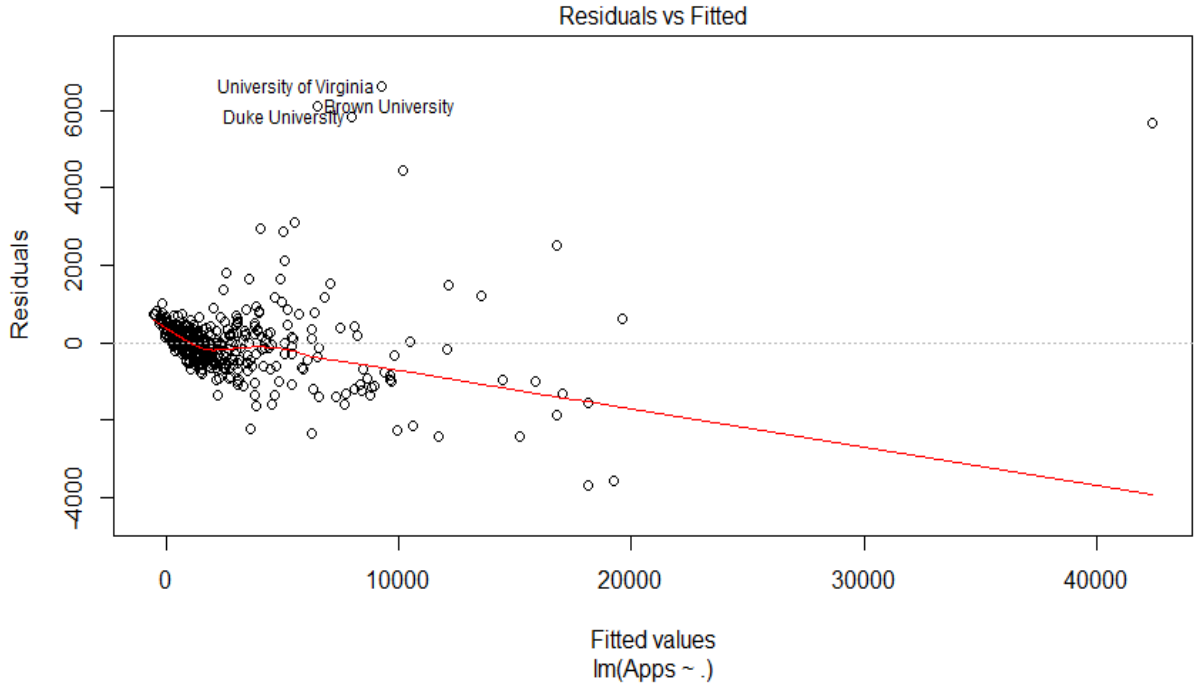
388 X 18

The dimension of the test set is:

389 X 18

Then, fitted a linear model using least squares on the training set and predicted the y^(y-hat) value using the test data as well as linear model:

The plot of the fitted line using least squares:



now, the Mean Square Error(MSE) i.e. test error is predicted; MSE= mean([(ytrue)-(yhat)]^2)

The least square error is: 1255144

(b) Ridge:

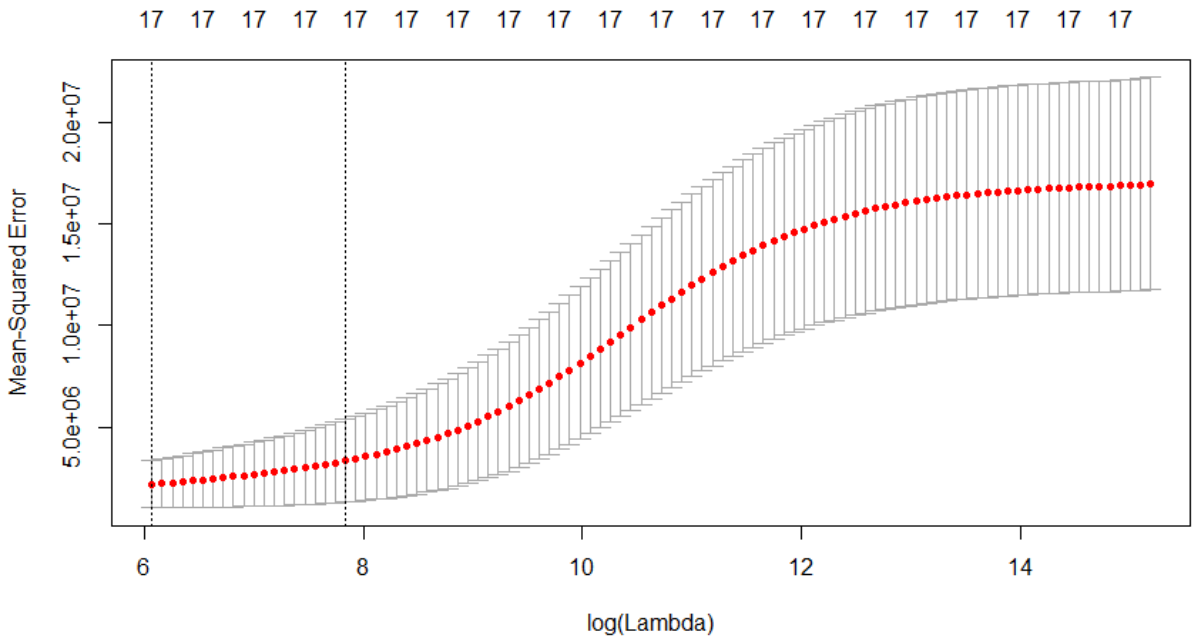
In this first convert the training data and test data into matrix form. since, cross validation as well as ridge takes input in the form of a matrix.

To find the best lambda for the model we will choose cross validation method:

cv.out<- cv.glmnet(train.matrixmod, c.trainingdata$Apps, alpha = 0).

Alpha=0 refers to ridge model.

The plot of the cross validation is:



We will choose the 'lambda.min' as the best value of lambda.

The optimal lambda is: 432.0172

Then, fitted a line using Ridge model on the training set.

Now, predicted the y^(y-hat) value using the test data, ridge model & best lambda.

now we will find the Mean Square error; MSE= mean([(ytrue)-(yhat)]^2)

ridge_error <- mean((c.testingdata[, "Apps"] - ridge.pred)^2)

The Ridge error is: 1233049

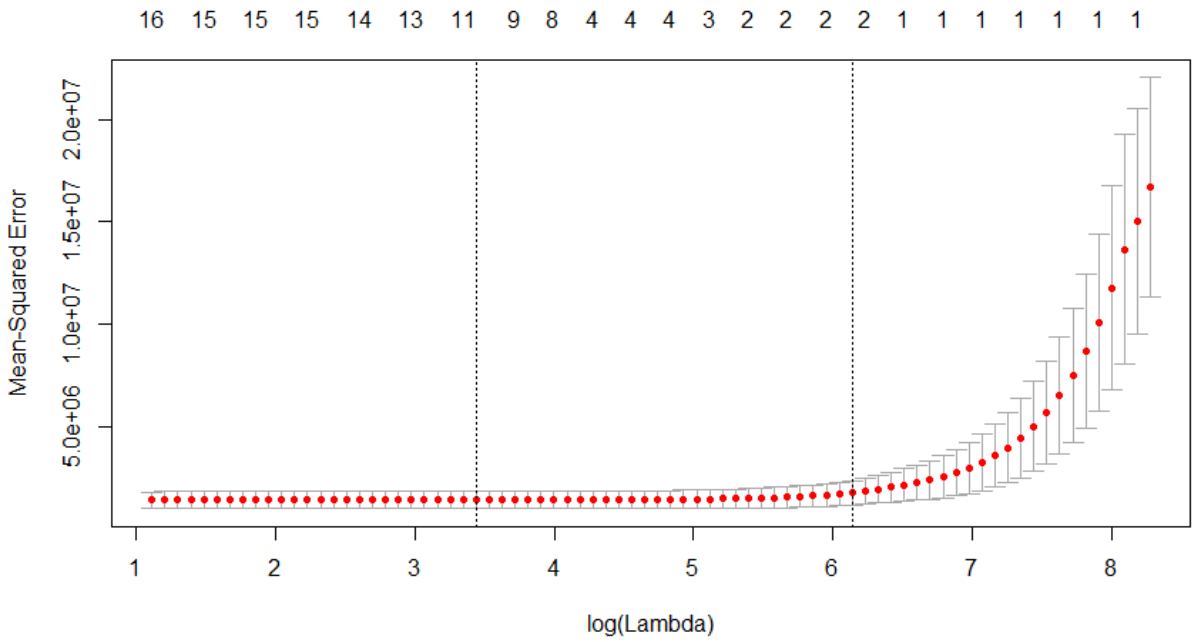
(d)Lasso:

To find the best lambda for the model we will choose cross validation method:

cv.out<- cv.glmnet(train.matrixmod, c.trainingdata$Apps, alpha = 1).

Alpha=1 refers to Lasso model.

The plot for cross validation is:



We will choose the ‘lambda.min’ as the best value of lambda.

The optimal lambda is: 31.19513

Then, fitted a line using Lasso model on the training set.

Now, predicted the y^(y-hat) value using the test data, Lasso model & best lambda.

now we will find the Mean Square error; MSE= mean([(ytrue)-(yhat)]^2)

The Lasso error is: 1290466

In Lasso we will take L1-norm. so, some of the coefficients gets reduced to zero.

The non-zero coefficients can be found by:

coeff_obs <- predict(lasso.mod, s=bestlam, type="coefficients")

PrivateYes	-472.46789708
Accept	1.57731006
Enroll	-0.38143329
Top10perc	30.13162107
Top25perc	0
F.Undergrad	0
P.Undergrad	0
Outstate	-0.01360515
Room.Board	0.05074282
Books	0.01918840
Personal	0
PhD	-3.36270366
Terminal	-3.37235597
S.F.Ratio	0
perc.alumni	0
Expend	0.02389514
Grad.Rate	0.92224715

(e) Principal Component Regression(PCR):

The PCR fit on the training set with k choosen by cross validation is find out.

The value of K selected by cross validation is 10 given by: summary(pcr.fit)

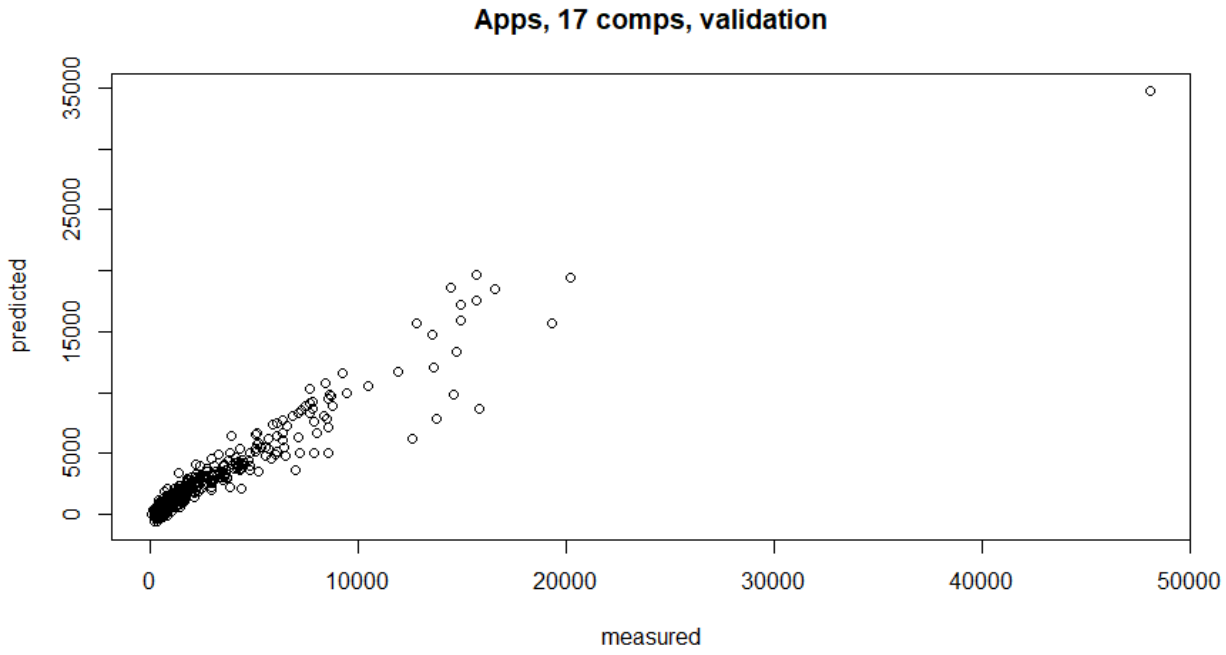
Now, predicted the y^(y-hat) value using the test data, PCR & k value.

now we will find the Mean Square error; MSE= mean([(ytrue)-(yhat)]^2)

The PCR error is: 1879927

Value of K: 10

The plot of the PCR fit is:



(f)Partial Least squares(PLS):

The PLS fit on the training set with k choosen by cross validation is find out.

The value of K selected by cross validation is 10 given by: summary(pls.fit)

Now, predicted the y^(y-hat) value using the test data, PLS & k value:

now we will find the Mean Square error; MSE= mean([(ytrue)-(yhat)]^2)

The PLS error is: 1251509

(g) The least square error is: 1255144

The Ridge error is: 1233049

The Lasso error is: 1290466

The PCR error is: 1879927

The PLS error is: 1251509

The least is the error, the better a model is predicting. Out of the above five model results, PCR is not performing well as it is having the highest error.

For, the other 4 models there isn't much difference among the test errors. Out of all the 5 models the order of preference is: Ridge>PLS>Least Square>Lasso>PCR

2.) OLS:

After performing OLS on the training data and start predicting the Y-hat values we will observe that some predictions are –ve, some are +ve and some are in between 0 and 1.

So, to classify the predictions between 0 and 1 (as y target contains only 2 classifiers) we define a class.

If y-hat<0 – define it as 0 classifier

y-hat>0 – define it as 1 classifier

If y-hat is between 0 and 1, then make it a 0 classifier. If y-hat is close to 0 or make it a 1 classifier if y-hat is close to 1.

Now after classifying them if you observe them

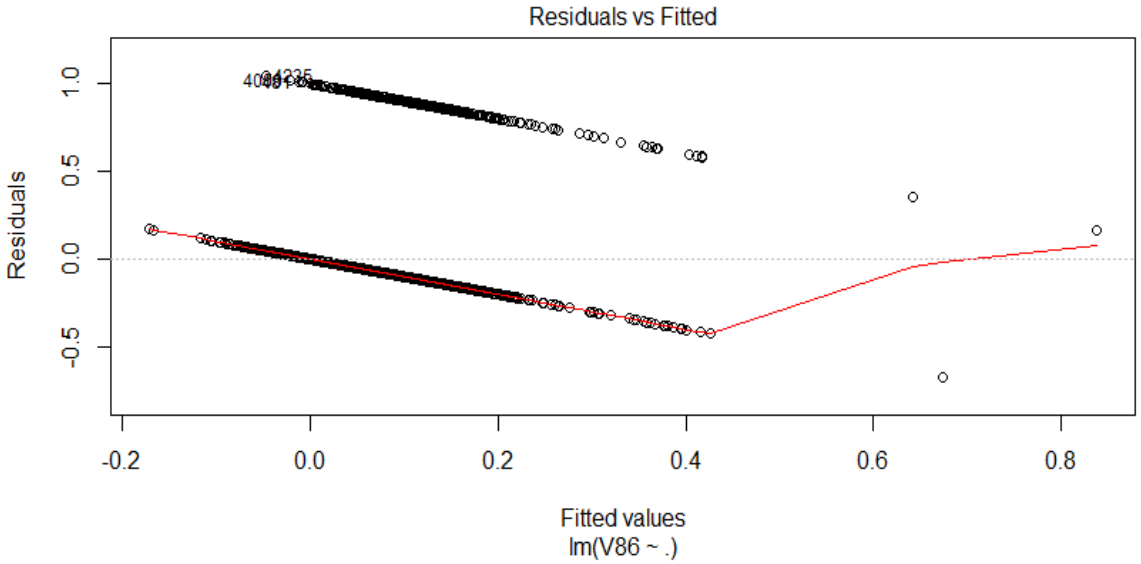
table(predictions): 0`s – 3997; 1`s - 3

table(Y_target$V1): 0`s – 3762; 1`s - 238

we can find that the predictions are not good. So, OLS it not a good model to predict those who will be interested in buying insurance policy.

OLS_error: `0.05975`

The plot of OLS:



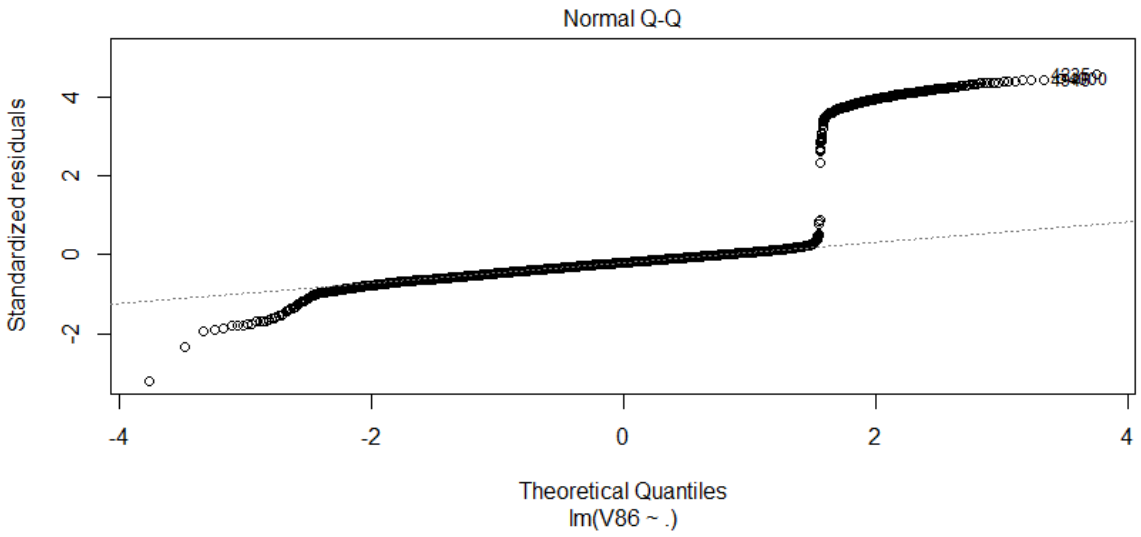
FORWARD SELECTION:

For, forward selection there is no particular method to find the prediction. So, we predict manually by writing code for that.

For a forward selection as there are 85 features we will be getting 85 error values. The minimum error of all of them is: min(test.error.fwd) - 0.05385551

The Position of the least value is: 23

The plot of forward subset selection is:



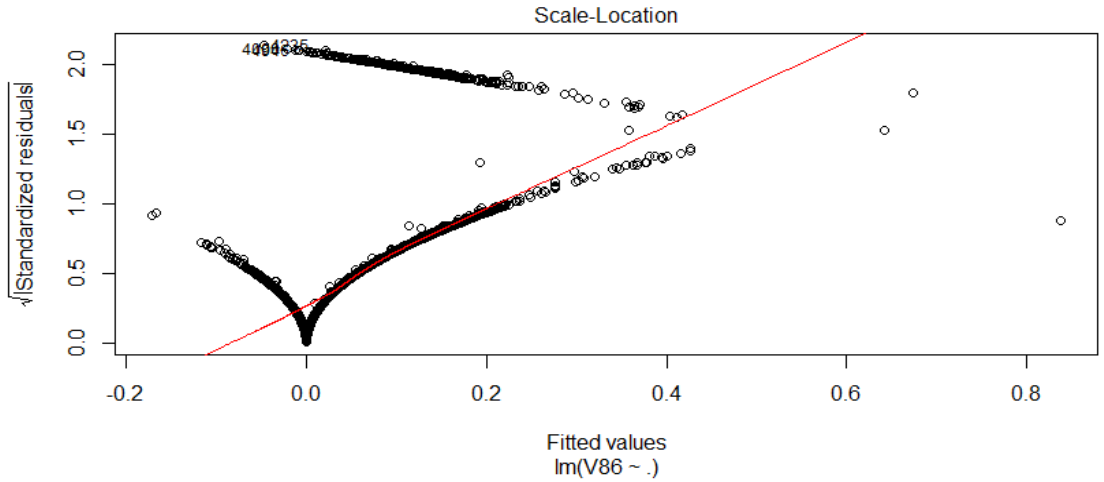
BACKWARD SELECTION:

Even for backward selection there is no particular model/ function to calculate the predictions. We do them manually.

Also, For a backward selection there are 85 features we will be getting 85 error values. The minimum error of all of them is: min(test.error.bwd) - 0.05383966

The Position of the least value is: 29

The plot of Backward selection is:



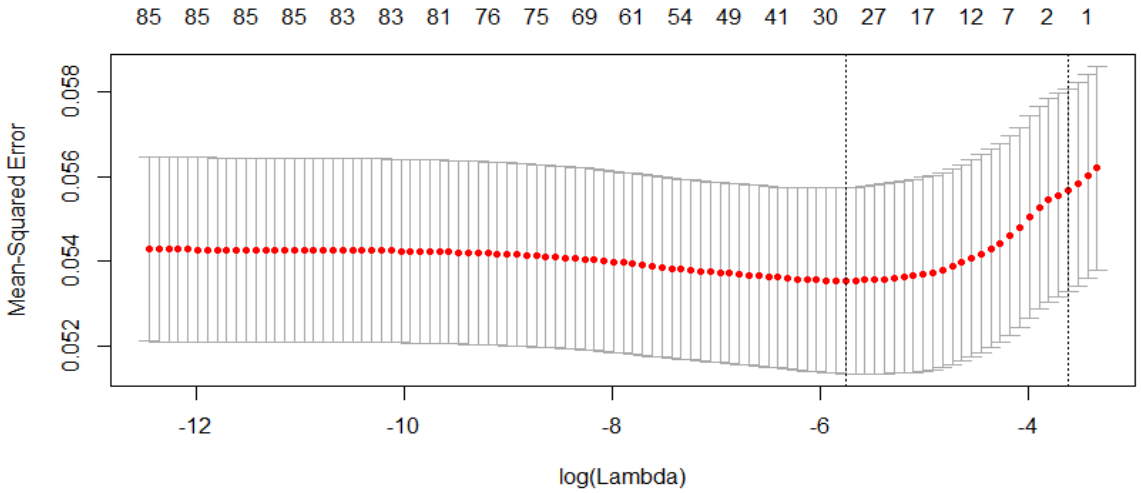
LASSO:

To find the best lambda for the model we will choose cross validation method:

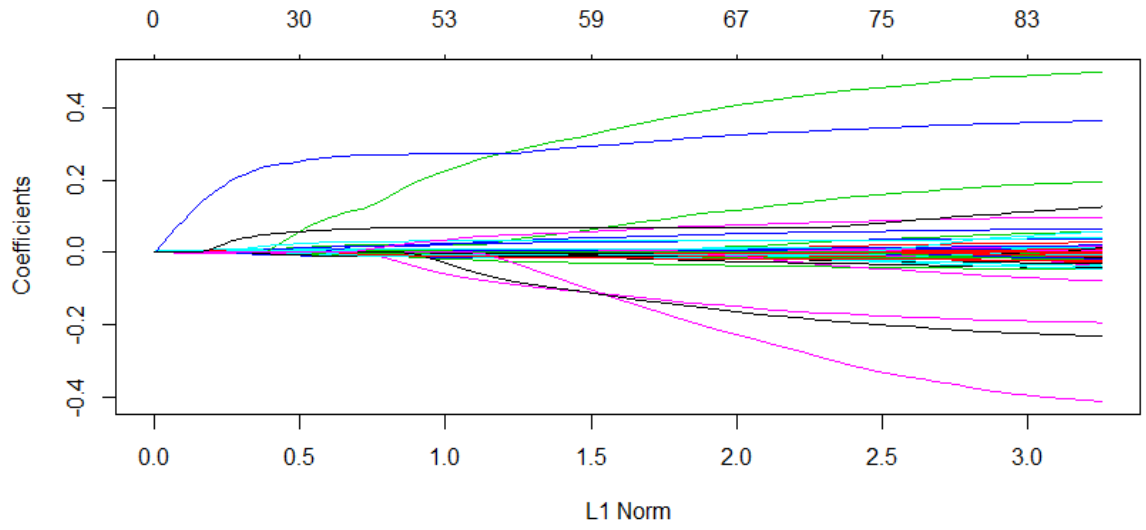
cv.out.lasso<- cv.glmnet(train.mat.mod[,1:85], traindata$V86, alpha = 1)

Alpha=1 refers to Lasso model.

The plot for cross validation is:



The plot of Lasso Model for training data:



We will choose the 'lambda.min' as the best value of lambda.

The optimal lambda is: `0.00290187`

Then, fitted a line using Lasso model on the training set.

Now, predicted the y^(y-hat) value using the test data, Lasso model & best lambda.

now we will find the Mean Square error; MSE= mean([(ytrue)-(yhat)]^2)

The Lasso error is: `0.05975`

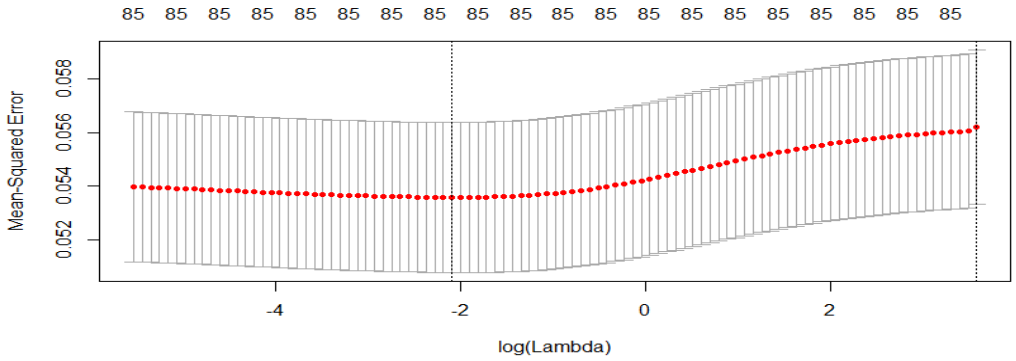
RIDGE:

To find the best lambda for the model we will choose cross validation method:

cv.out.ridge<- cv.glmnet(train.mat.mod[,1:85], traindata$V86, alpha = 0)

Alpha=0 refers to ridge model.

The plot for cross validation is:



We will choose the 'lambda.min' as the best value of lambda.

The optimal lambda is: `0.1227271`

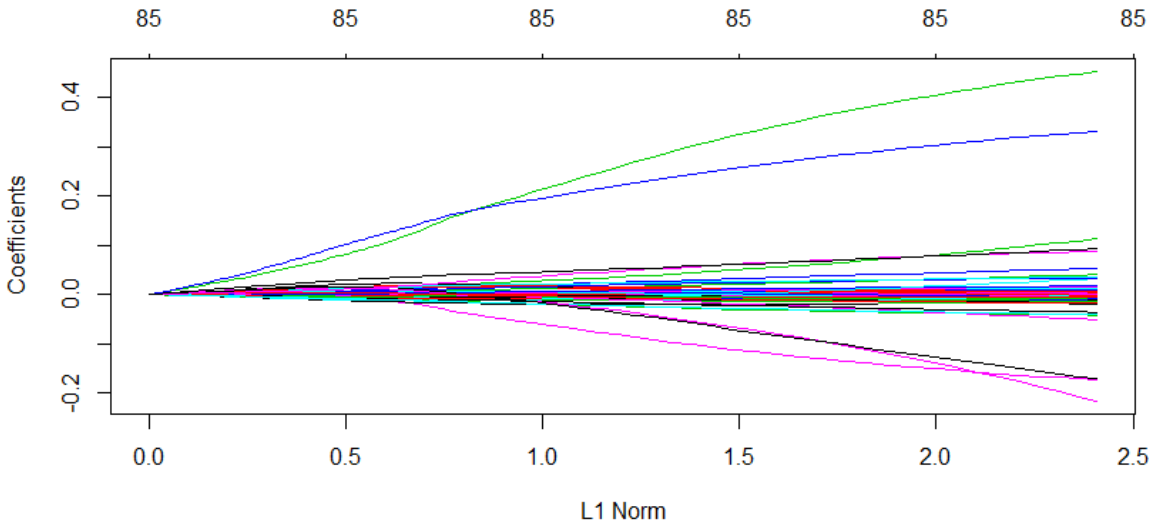
Then, fitted a line using Ridge model on the training set.

Now, predicted the y^(y-hat) value using the test data, Ridge model & best lambda.

now we will find the Mean Square error; MSE= mean([(ytrue)-(yhat)]^2)

The Ridge error is: `0.05925`

The plot of RIDGE Model for training data:



Summary:

OLS Error is: 0.05975

Forward Error is: 0.05385551

Backward Error is: 0.05383966

The Lasso error is: 0.05975
The Ridge error is: 0.05925

By, looking at errors we can infer that there isn't much difference among the test errors between different models. Also, we can infer that no model is better enough to predict this analysis.

The order of preference is: Backward>Forward >Ridge>Lasso >OLS.

No model is good enough to predict who will be interested in buying a caravan insurance policy.

3.)

First generated a dataset with p = 20 features, n = 1, 000 observations according to the given model using "rnorm".

Equated some of the coefficients to zero as specified in the question.

Splitted the data set into a training set containing 100 observations and a test set containing 900 observations.

dim(train_data_mod): 100 X 21

dim(test_data_mod): 900 X 21

performed both forward and backward subset selection on training data as well as test data.

Forward subset selection:

Minimum test error for forward subset selection: 1.20316
Position of the Test error element: 14

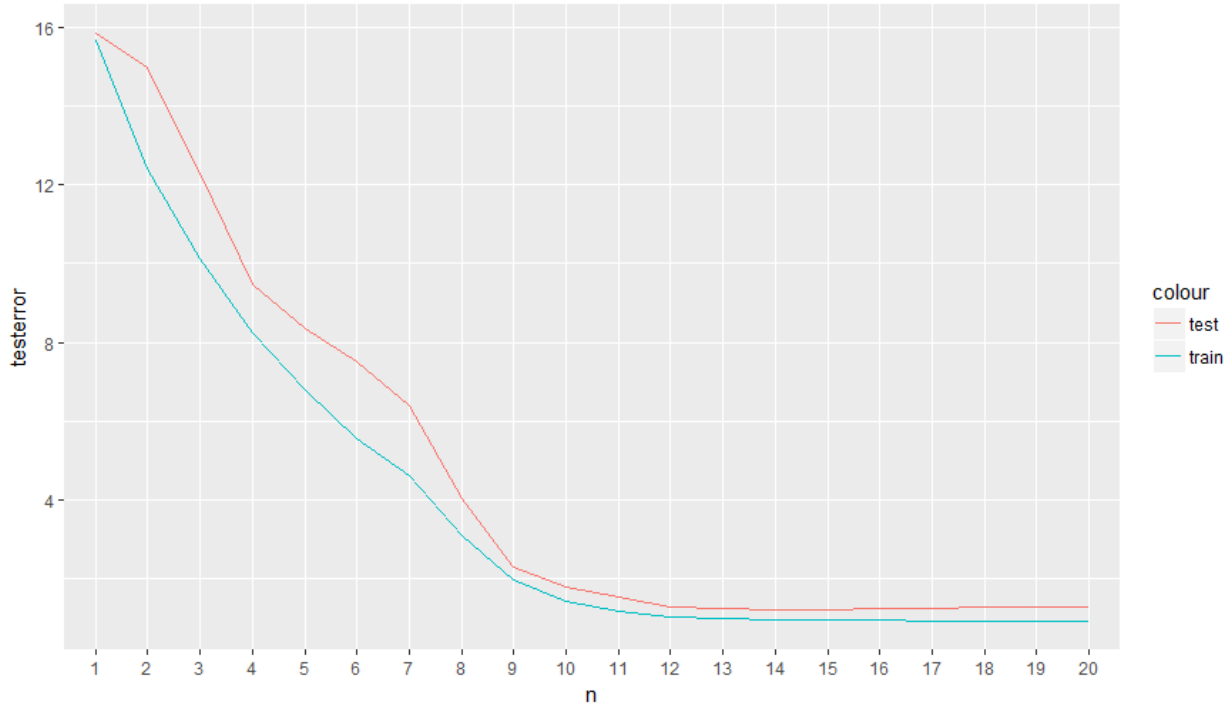
Minimum train error for forward subset selection: 0.9341086
Position of the train error element: 20

Backward Subset selection:

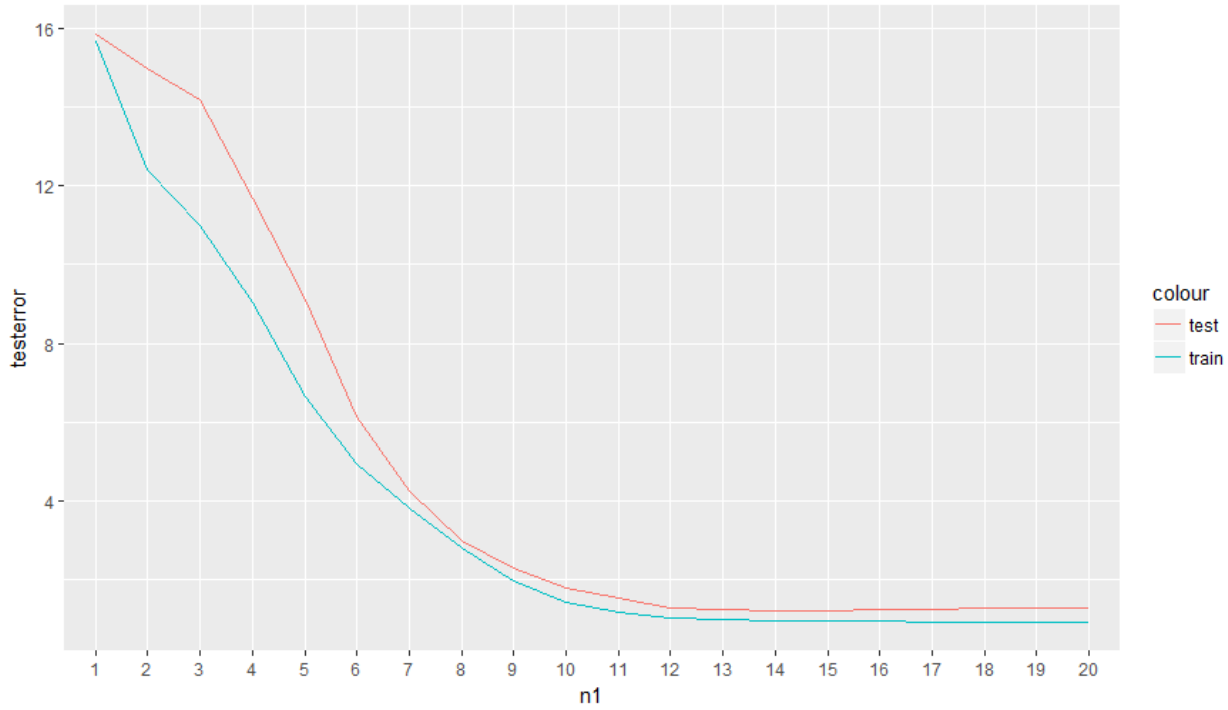
Minimum test error for backward subset selection: 1.20316
Position of the Test error element: 14

Minimum train error for backward subset selection: 0.9341086
Position of the train error element: 20

Plot of training error as well as test error for forward subset selection:



Plot of training error as well as test error for Backward subset selection:



The test set MSE take on its minimum value at position: 14

From above we can observe that both forward and backward subset selection are generating same results for the given data. So, both forward as well as backward are good for enough for the prediction.

The value of coefficients can be observed by:

coef(regfit.forward, id=14)

coef(regfit.backward, id=14)