Code

Regularization Techniques

Sushma Vijayeendra Karjol

Regularization Techniques

Instructor : Dr. Vladimir Shapiro

Introduction

In this module, we will be using the glmnet package to perform the Ridge Regression and the Lasso Regression. To fit the ridge regression models and lasso models the main function of this package, cv. glmnet() or cross-validation glmnet is used. And, this function has quite a different syntax from other model-fitting functions. A college dataset from the ISLR is used for the study. This study aims to know which model fits well to know the predictions of variable - 'Graduation rate' for private and non-private colleges using the rest of the variables as dependents. This is divided into three parts. In the first part, the data set is regularised using the Ridge method, where lambda. min and lambda.1se values are estimated. Then the methods are visually represented and compared. The best fit of the Ridge regression model is set with the train and test dataset. The second part includes the same set of procedures are followed but with LASSO regression. Coefficients of variables are compared for the study to know which one would be the best fit. The third part involves the analysis and discussions.

College Dataset from the library ISLR is a assigned to M4_College

Hide

```
library(ISLR)
attach(College)

M4_College <- College[-1]

# first column has the categorical dataset which is deleted for the study.
```

1) Split the data into a train and test set – refer to the ALY6015_Feature_Selection_R.pdf document for information on how to split a dataset.

The dataset is split into training and testing sets. It is the important part of evaluation data mining models. Typically , a large portion of dataset is allocated for training and smaller for testing.

Hide

```
library (caret)
```

```
Loading required package: lattice
Loading required package: ggplot2
Registered S3 method overwritten by 'data.table':
  method           from
  print.data.table
```

Hide

```
library(lattice)
library(ggplot2)

# spliiting the dataset for th study
set.seed(1520)
trainIndex <- createDataPartition(M4_College$Apps,p=0.7,list=FALSE,times=1)
M4College_train <- M4_College[ trainIndex,]
M4College_test <- M4_College[-trainIndex,]
```

```
The dataset is split into - training and test sets. They differ in size because of th
e  way the samples are being taken.
  Train dataset is used to fit the model and the test is conducted to  test if the pr
ediction fits the dataset.
```

Hide

```
# provides details of the variables

summary(M4College_train)
```

```
      Apps            Accept          Enroll         Top10perc       Top25perc     
 Min.   :   81   Min.   :   72   Min.   :  35.0   Min.   : 1.00   Min.   :  9.00  
 1st Qu.:  776   1st Qu.:  614   1st Qu.: 244.0   1st Qu.:15.00   1st Qu.: 40.00  
 Median : 1558   Median : 1097   Median : 430.0   Median :24.00   Median : 55.00  
 Mean   : 3004   Mean   : 2015   Mean   : 773.4   Mean   :27.42   Mean   : 55.73  
 3rd Qu.: 3624   3rd Qu.: 2424   3rd Qu.: 913.0   3rd Qu.:35.00   3rd Qu.: 69.00  
 Max.   :48094   Max.   :26330   Max.   :6180.0   Max.   :96.00   Max.   :100.00  
  F.Undergrad     P.Undergrad         Outstate       Room.Board       Books       
 Min.   :  139   Min.   :    1.0   Min.   : 2340   Min.   :1880   Min.   :  96.0  
 1st Qu.:  979   1st Qu.:   86.0   1st Qu.: 7380   1st Qu.:3603   1st Qu.: 470.0  
 Median : 1707   Median :  341.0   Median :10100   Median :4230   Median : 500.0  
 Mean   : 3666   Mean   :  869.6   Mean   :10484   Mean   :4368   Mean   : 552.7  
 3rd Qu.: 4296   3rd Qu.:  963.0   3rd Qu.:12950   3rd Qu.:5050   3rd Qu.: 600.0  
 Max.   :28938   Max.   :21836.0   Max.   :20100   Max.   :8124   Max.   :2340.0  
    Personal         PhD            Terminal        S.F.Ratio      perc.alumni   
 Min.   : 300   Min.   :  8.00   Min.   : 24.00   Min.   : 2.50   Min.   : 0.00  
 1st Qu.: 882   1st Qu.: 62.00   1st Qu.: 71.00   1st Qu.:11.50   1st Qu.:13.00  
 Median :1200   Median : 75.00   Median : 82.00   Median :13.60   Median :21.00  
 Mean   :1362   Mean   : 72.38   Mean   : 79.49   Mean   :14.04   Mean   :22.71  
 3rd Qu.:1702   3rd Qu.: 85.00   3rd Qu.: 92.00   3rd Qu.:16.30   3rd Qu.:31.00  
 Max.   :6800   Max.   :100.00   Max.   :100.00   Max.   :39.80   Max.   :60.00  
     Expend        Grad.Rate    
 Min.   : 3186   Min.   : 10.0  
 1st Qu.: 6817   1st Qu.: 53.0  
 Median : 8377   Median : 66.0  
 Mean   : 9612   Mean   : 65.6  
 3rd Qu.:10872   3rd Qu.: 78.0  
 Max.   :45702   Max.   :118.0  
```

Hide

```
# dimension of the dataset
dim(M4College_train)
```

```
[1] 545  17
```

Summary determines the descriptive statistics of the all the variables in the dataset.

2) Use the glmnet function to estimate the lambda.min and lambda.1se values. Compare and discuss the values.

```
The glmnet() function conatins alpha argument that provides the model that is fit. If
alpha is  0 then a ridge regression model is fit, and if alpha is  1 then a lasso mod
el is fit. First fit Ridge Regression model:
```

Hide

```
library(glmnet)
```

Hide

```
library(dplyr)
```

```

Attaching package: ‘dplyr’

The following objects are masked from ‘package:stats’:

    filter, lag

The following objects are masked from ‘package:base’:

    intersect, setdiff, setequal, union
```

Hide

```
library(foreach)
#install.packages("Matrix")
library(Matrix)

train.mat = model.matrix(Grad.Rate~., data=M4College_train)[,-1]

lambda_seq <- 10 ^ seq(2,-2,by = -.1)

mod.ridge = cv.glmnet(x = train.mat, y = M4College_train$Grad.Rate, alpha = 0,lambda=
lambda_seq )
mod.ridge
```

```

Call:  cv.glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda_seq,
alpha = 0)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min  1.995    18     172 15.77      16
1se 25.119     7     186 17.27      16
```

Hide

```
cat("\n lambda.best \n")
```

```

 lambda.best
```

Hide

```
lambda.best = mod.ridge$lambda.min
lambda.best
```

```
[1] 1.995262
```

Hide

```
cat("\n lambda.lse \n")
```

```

 lambda.lse
```

Hide

```
lambda.1se = mod.ridge$lambda.1se
lambda.1se
```

```
[1] 25.11886
```

To predict the ‘Grad rate variable’ train data set is regularized using Ridge method by bringing the co-efficient of all the variables which have a lesser impact on the Graduation rate, are brought to zero. Based on which lambda. min and lambda. lse is calculated. lambda. min is considered as best as it has the least value with 1.995. And, lambda.1se with 25.119 is the largest value of the lambda which has an error of 17.27, usually one standard error away from the minimum. The penalty term lambda regularizes the coefficients to reduce the complexity in the model by shrinking the multi-collinearity.

Hide

```
test.mat <- model.matrix(Grad.Rate~., data = M4College_test)[,-1]
y = M4College_test$Grad.Rate

alpha_predict <- predict(mod.ridge, s = mod.ridge$lambda.min , newx = test.mat)
```

since we would compare Ridge model with Lasso in the later part , for consistency we will be using lamba.1se in both the cases.

Hide

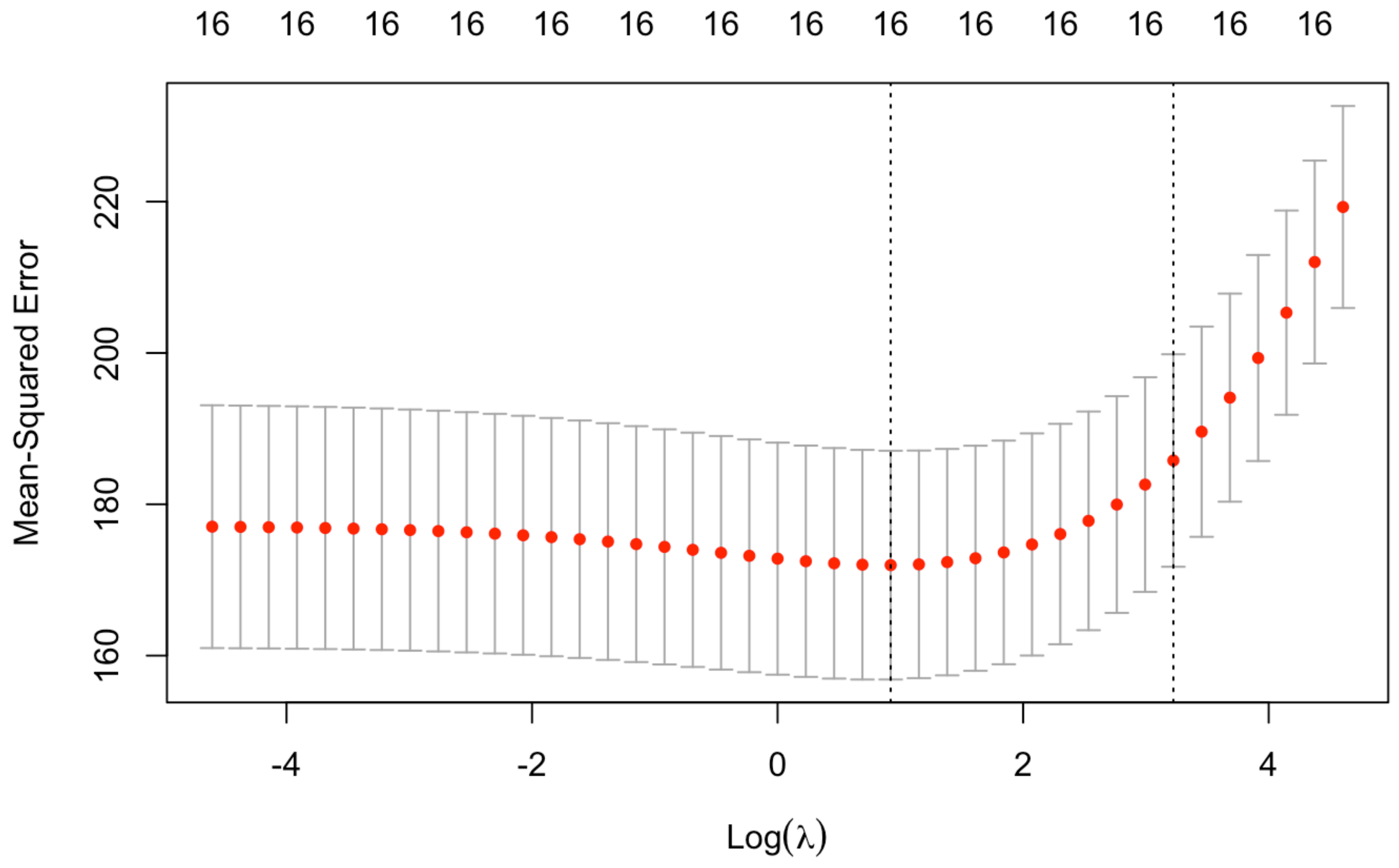
```
#glmnet(x, y, alpha = 0.2, weights = wts, nlambda = 20)

fit <- glmnet(train.mat, M4College_train$Grad.Rate , alpha = 0)
```

3)Plot the results from the glmnet function, provide an interpretation. What does this plot tell us?

Hide

```
plot(mod.ridge)
```



In Ridge regression we estimate the coefficients of multiple-regression models where the independent variables are highly correlated. The least square values are unbiased , the variances are large and are far away from the true values. Here , Ridge minimizes the residual sum of squares .

4) Fit a Ridge regression model against the training set and report on the coefficients. Is there anything interesting?

Hide

```
# Rebuilding the model with best lamda value identified
Ridge_best <- glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda.be
st)
Ridge_best
```

```


Call:  glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda.best)

  Df  %Dev Lambda
1  6 39.14  1.995
```

Hide

```
# table of co-efficent values for each variable
coef(Ridge_best)
```

```
17 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept) 42.7809929441
Apps         .
Accept       .
Enroll       .
Top10perc    0.0589106769
Top25perc    0.0851650613
F.Undergrad  .
P.Undergrad -0.0004624633
Outstate     0.0013289825
Room.Board   .
Books        .
Personal    -0.0002377431
PhD          .
Terminal     .
S.F.Ratio    .
perc.alumni  0.1432326606
Expend       .
```

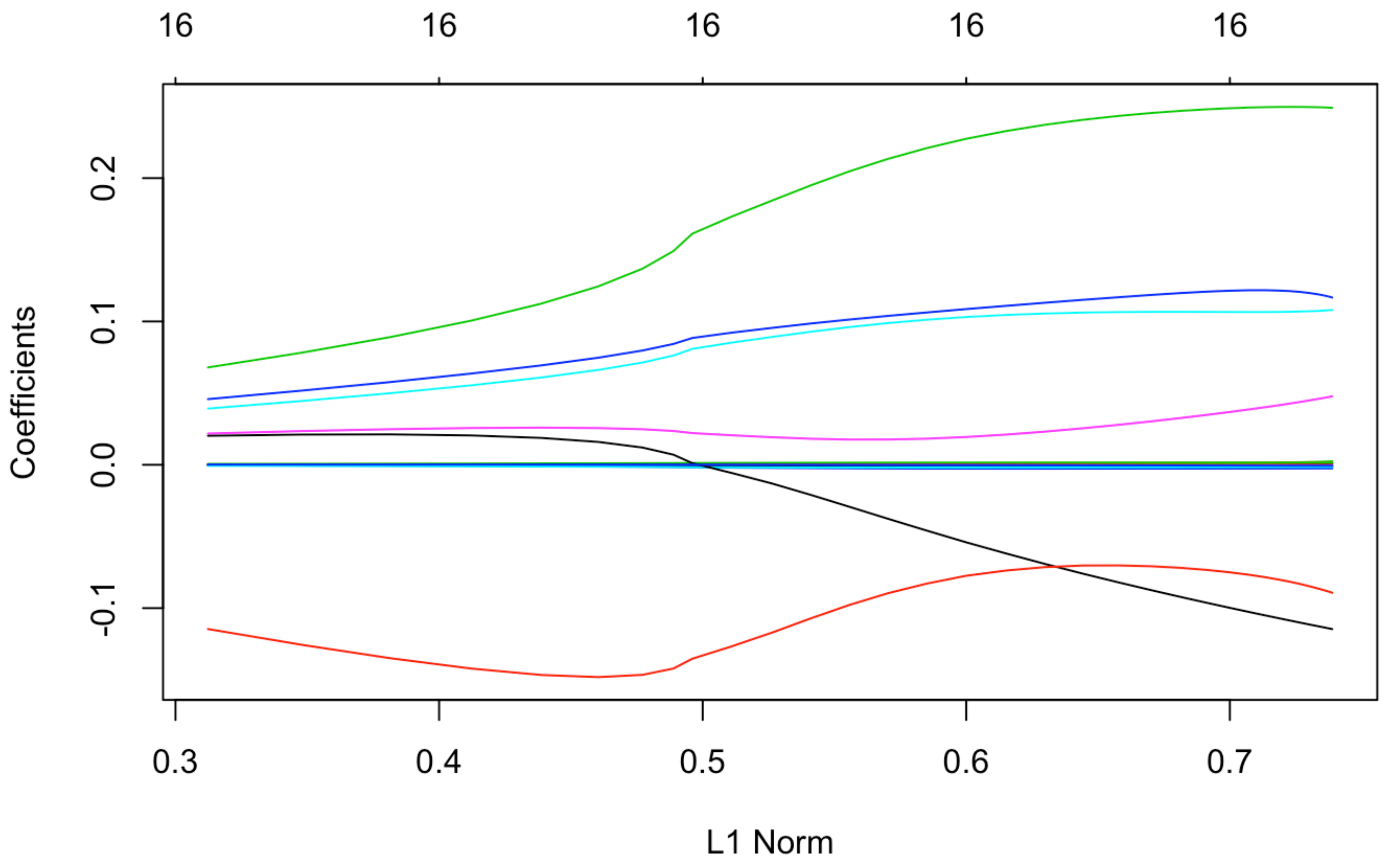
Hide

```
?glmnet()
```

```
The co-efficient of Apps,Accept,Enroll,Top10per,F.Undergrad,Terminal, Room.Board,Book
s, Personal,S.F.Ratio approach to zero at a stage stating their lower impact on Grad
rate . Further more, the variables Top10perc, Top25perc ,Outstate, perc.alumni have g
reater impact on increasing  'Grad rate' in all the Colleges as the coefficients of t
hese variables don't tur to zero.
```

Hide

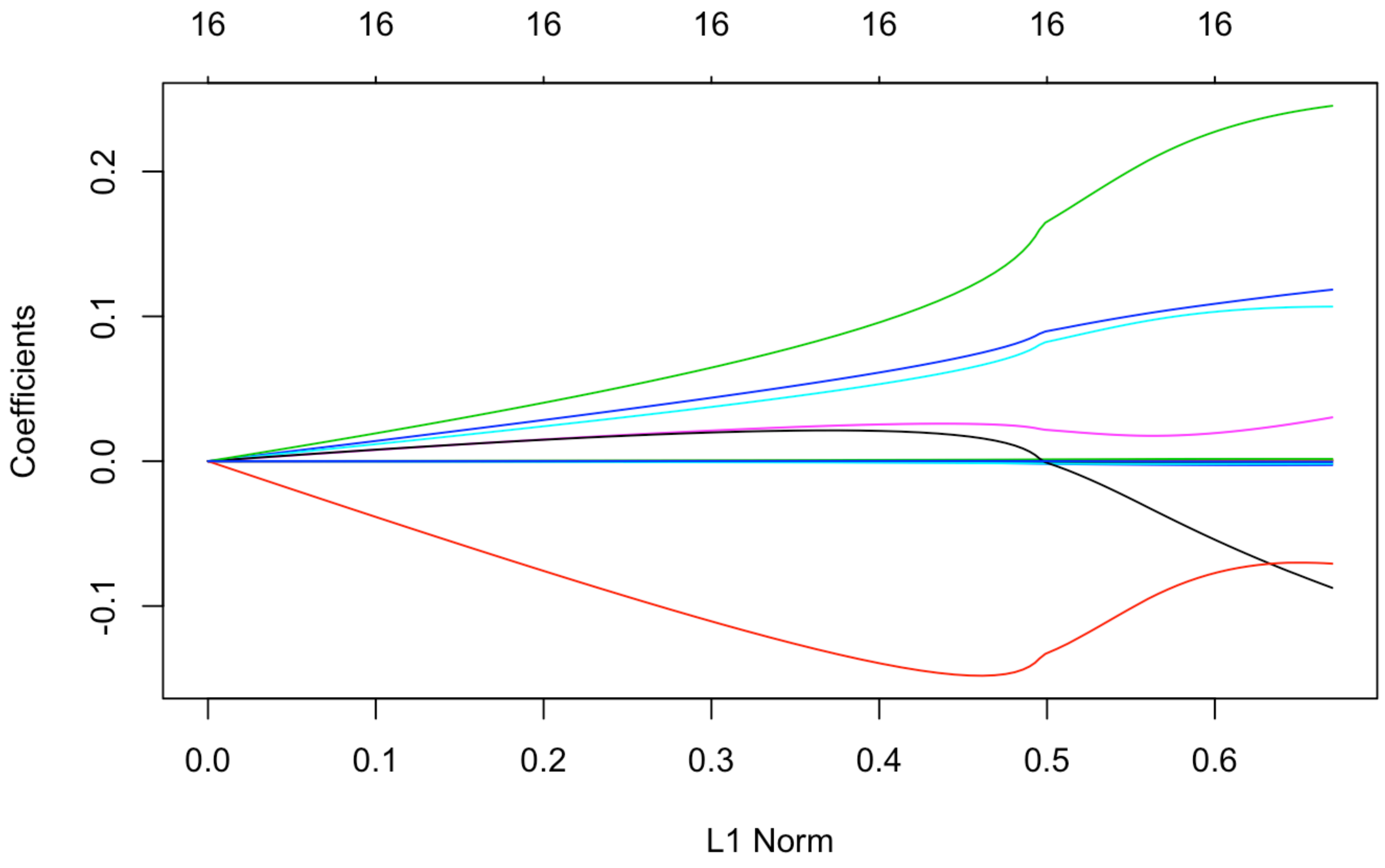
```
plot(mod.ridge$glmnet.fit)
```



Hide

```
NA
NA
```

```
The above plot designates the whole path of variables as they narrow towards zero. As
lambda values raise the coefficients approach to zero. Vice-versa when the coefficien
ts are unregularised when the lambda is approaching zero. The ridge regression here h
as penalized the coefficients such that the least efficient variables are in the esti
mation to shrink. This indicates that several variables have no impact on the 'gradua
tion rate'. There are two co-efficient that have negative values. The faster the vari
ables are shrinking they have a lesser role to play and can increase the chances of '
Grad rate'.
```



Hide

```
# creating matrix for test dataset.
test.mat = model.matrix(Grad.Rate~., data=M4College_test)
```

5) Determine the performance of the fit model against the training set by calculating the root mean square error (RMSE). sqrt(mean((actual - predicted)^2))

Hide

```
# RMSE calculaton for train data set

Ridge_best # model created using best lmada value
```

```
Call:  glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda.best)

  Df  %Dev Lambda
1  6 39.14  1.995
```

Hide

```
alpha_predictTrain <- predict(Ridge_best, s = lambda.best , newx = train.mat)

predict <- alpha_predictTrain

cat ("\n Root mean square value of Ridge regression against train dataset \n")
```

```
 Root mean square value of Ridge regression against train dataset
```

Hide

```
sqrt(mean((M4College_train$Grad.Rate - predict)^2))
```

```
[1] 13.43388
```

```
The root mean square value of Ridge Regression  method is 13.6257.
```

6) Determine the performance of the fit model against the test set by calculating the root mean square error (RMSE). Is your model overfit?

Hide

```
# matrix for train dataset
test.mat = model.matrix(Grad.Rate~., data = M4College_test)[,-1]


y = M4College_test$Grad.Rate

fit <- glmnet(x = test.mat, y = M4College_test$Grad.Rate)

alpha_predictTest <- predict(Ridge_best, s = lambda.best , newx = test.mat)

actualTest <- mod.ridge
predictTest <- alpha_predictTest

cat ("\n Root mean square value of Ridge regression against test dataset \n")
```

```

 Root mean square value of Ridge regression against test dataset
```

Hide

```
sqrt(mean((M4College_test$Grad.Rate - predictTest)^2))
```

```
[1] 13.37291
```

In comparison with the RMSE of test dataset is 13.52 while train dataset is 13.62. Its difficult to conclude that the model is overfit or underfit as the RMSE values are almost equal

7) Use the cv.glmnet function to estimate the min and lambda.1se values. Compare and discuss the values.

Hide

```
lambda_seqLasso <- 10^seq(2, -2, by = -.1)

mod.lasso = cv.glmnet(train.mat, M4College_train$Grad.Rate, alpha=1 ,lambda = lambda_
seqLasso)
mod.lasso
```

```
Call:  cv.glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda_seqLas
so,      alpha = 1)

Measure: Mean-Squared Error

    Lambda Index Measure    SE Nonzero
min 0.3162    26   172.1 15.48      11
1se 1.9953    18   185.8 14.70       6
```

Hide

```
lambda.bestLasso = mod.lasso$lambda.min
lambda.bestLasso
```

```
[1] 0.3162278
```

Hide

```
lambda.1se = mod.lasso$lambda.1se
lambda.1se
```

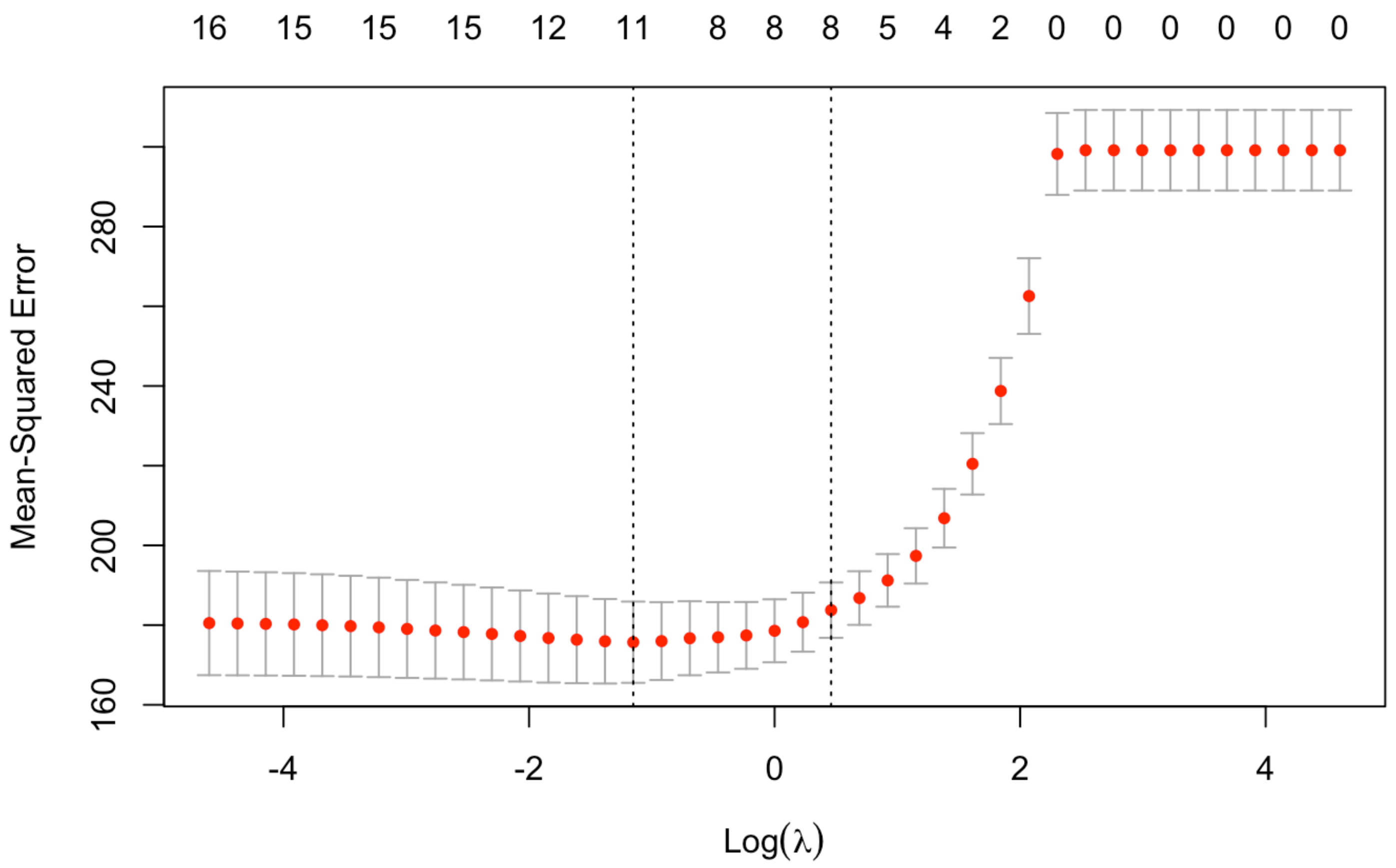
```
[1] 1.995262
```

```
 The minimum Labda is 0.2512 with Standard error of 8.959 which includes 12 variables
. Lambda 1se is 1.2589 with Standard error of 7.668 and with only  8 variables.
```

8) Plot the results from the glmnet function, provide an interpretation. What does this plot tell us?

Hide

```
plot(mod.lasso)
```



```
     The above graph is the cross validation curve with red dotted line. The plot al
so has the lower and upper standard deviation curves along with error bars( lambda se
quence). There are two vertical dotted lines which indicate lamda.min and lambda.1se.
Lamda.min provides minimum mean of cross validation error. And, lambda.1se provides
corss -validaton error one standard deviation away from the labda.min.
```

9) Fit a LASSO regression model against the training set and report on the coefficients. Do any coefficients reduce to zero? If so, which ones?

Hide

```
lasso_best <- glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda.be
stLasso)
lasso_best
```

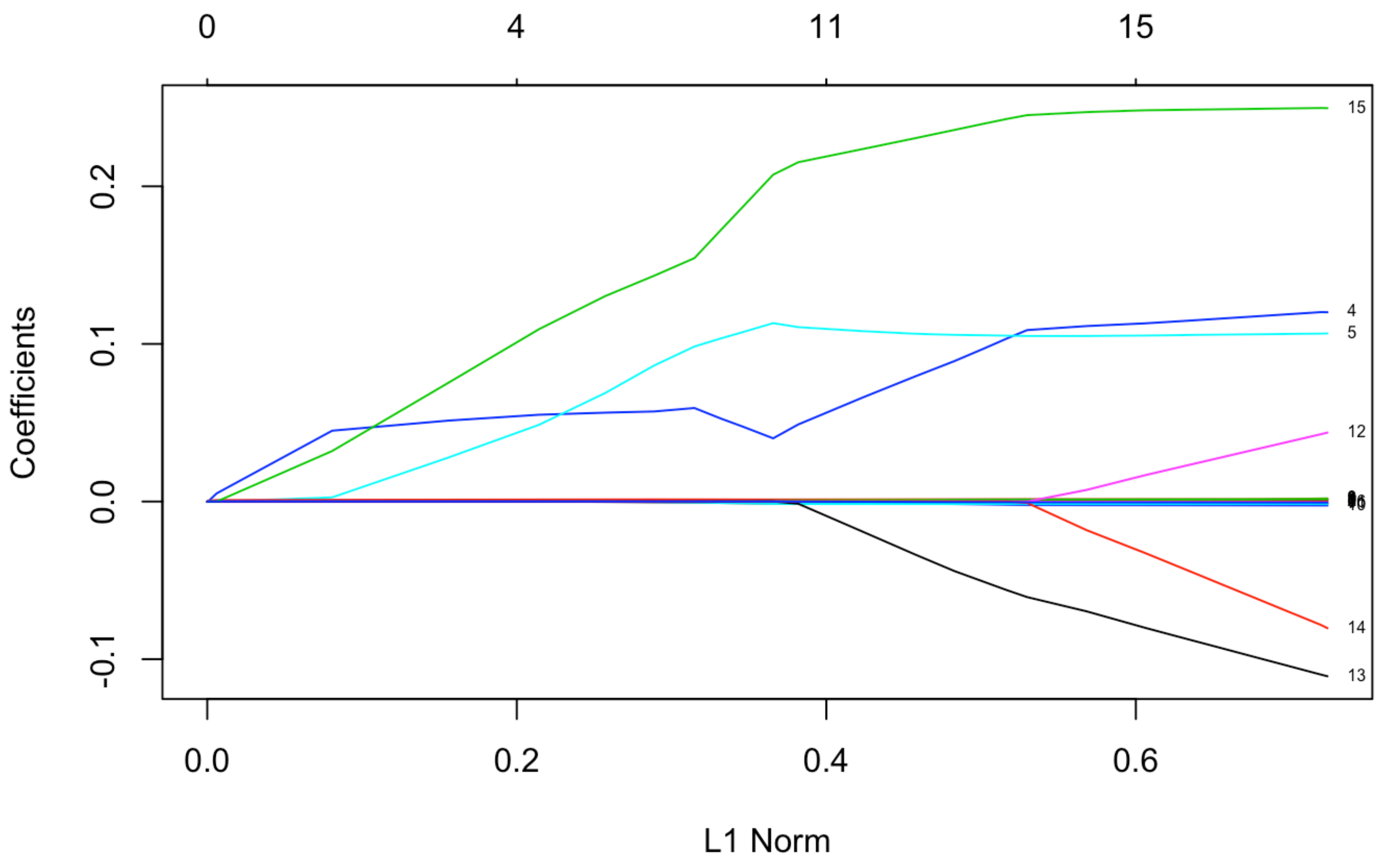
```
Call:  glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda.bestLasso
)

  Df  %Dev Lambda
1 12 45.15 0.2512
```

Hide

```
#predict(mod.lasso, s = mod.lasso$lambda.1se , newx =)
#predict(mod.lasso, s=lambda.best, type="coefficients")

plot(mod.lasso$glmnet.fit , label = TRUE)
```



```
The above graph measure the multi-linear complexity of the model. This is graphically
explained against train data set.There are several co-efficient values that are blowi
ng-up at the end of the path. Along side coefficient of few variables are suppressed
to zero.
```

Hide

```
coef(lasso_best)
```

```
17 x 1 sparse Matrix of class "dgCMatrix"
                       s0
(Intercept) 40.8651778076
Apps         0.0005533179
Accept       .
Enroll       .
Top10perc    0.0798603795
Top25perc    0.1060521441
F.Undergrad  .
P.Undergrad -0.0014723046
Outstate     0.0013854095
Room.Board   0.0010241267
Books       -0.0012158800
Personal    -0.0015175287
PhD          .
Terminal    -0.0333919968
S.F.Ratio    .
perc.alumni  0.2305534682
Expend      -0.0002690844
```

10) Determine the performance of the fit model against the training set by calculating the root mean square error (RMSE). sqrt(mean((actual - predicted)^2))

Hide

```
# calcualting RMSE

lasso_best <- glmnet(x = train.mat, y = M4College_train$Grad.Rate, lambda = lambda.be
stLasso)

train.mat <- model.matrix(Grad.Rate~., data = M4College_test)[,-1]
y = M4College_test$Grad.Rate

alpha_predictL <- predict(lasso_best, s = mod.lasso$lambda.min , newx = train.mat)

actualTrain <- mod.lasso
predictTrain <- alpha_predictTest

cat("\n Root mean square error of train dataset \n")
```

```

 Root mean square error of train dataset
```

Hide

```
sqrt(mean((M4College_test$Grad.Rate - predictTrain)^2))
```

```
[1] 13.37291
```

11) Determine the performance of the fit model against the test set by calculating the root mean square error (RMSE). Is your model overfit?

Hide

```
cat("\n Root mean square error of test dataset \n")
```

```

 Root mean square error of test dataset
```

Hide

```
sqrt(mean((M4College_test$Grad.Rate - predictTest)^2))
```

```
[1] 13.37291
```

The RMSE for train dataset and test data set is equivalent with 13.52. Through this it can be said that the model is good fit. However its diffcilut to conclude that it is good fit for any of the independent data sets.

12) Which model performed better and why? Is that what you expected?

The aim of using these two models is to shrink and regularize the co-efficient. This improves the prediction accuracy and decreases the variance. Altogether interpretability can also be improved.

```
In ridge regression, a penalty is tuning the parameter called lambda was chosen from
the cross-validation. It is making the model fit small by decreasing the residual sum
of squares also by including a shrinkage penalty. Thus the coefficients that have lar
ge values are penalized. And as the lambda is growing, the bias is not changed, howev
er, variance is getting dropped. Ridge is not filtering any variables but selecting a
ll the variables in the final model. The RMSE value for the trained dataset is 13.625
7 and the test data set is 13.52526.

On the other hand, the lasso is shrinking the coefficient either approximately to zer
o or setting the variables directly to zero when the lambda value is big. But lasso d
oes the variable selection. A combination of shrinkage along with a selection of vari
ables makes it a better model than ridge when both the models have. The RMSE value fo
r the trained dataset is 13.52526 and the test data set is 13.52526.
```

In comparison, since lasso filters and selects few variables, better results were expected. On the other hand, Ridge considers all the variables in reducing the coefficients. However, we observed stable RMSE values for both models.

13) Refer to the ALY6015_Feature_Selection_R.pdf document for how to perform stepwise selection and then fit a model. Did this model perform better or as well as Ridge regression or LASSO? Which method do you prefer and why?

Hide

```
library(leaps)
library(ISLR)
library(dplyr)
```

Regression subset function or Regsubset() from the leaps package identifies different
models with different sizes. We need to only specify the ‘nmax’ values which inputs the maximum number of variables to be included in the model. Among these best variables are quantified using RSS.

Hide

```
reg.summary
```

```
Subset selection object
Call: regsubsets.formula(Grad.Rate ~ ., data = M4_College, nvmax = 19)
16 Variables  (and intercept)
            Forced in Forced out
Apps            FALSE      FALSE
Accept          FALSE      FALSE
Enroll          FALSE      FALSE
```

```
Top10perc       FALSE      FALSE
Top25perc       FALSE      FALSE
F.Undergrad     FALSE      FALSE
P.Undergrad     FALSE      FALSE
Outstate        FALSE      FALSE
Room.Board      FALSE      FALSE
Books           FALSE      FALSE
Personal        FALSE      FALSE
PhD             FALSE      FALSE
Terminal        FALSE      FALSE
S.F.Ratio       FALSE      FALSE
perc.alumni     FALSE      FALSE
Expend          FALSE      FALSE
1 subsets of each size up to 16
Selection Algorithm: exhaustive
          Apps Accept Enroll Top10perc Top25perc F.Undergrad P.Undergrad Outstate
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 ) "*"  "*"    "*"    "*"       "*"       "*"         "*"         "*"
          Room.Board Books Personal PhD Terminal S.F.Ratio perc.alumni Expend
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 ) "*"          "*"    "*"       "*" "*"       "*"        "*"          "*"
```

The summary function sets the variables that provides the best models. In the above results the good variables are represented by the asterisk. From the above we can conclude that a good model can be created using one variable to max of sixteen variables. Among them Outstate, Top25perc ,perc.alumni , Apps are some of the important variables in an ordinal form.

Hide

```
#fit_subset <- lm(Grad.Rate ~ Outstate + Top25perc + P.Undergrad + perc.alumni + Room
.Board + + Apps)

fit_subset <- step(lm(Grad.Rate ~. , data =  M4College_train), direction = 'both', st
eps = 16 , trace = FALSE)

Subset_lm_Train <- predict.lm(fit_subset, newdata = M4College_train)



library(mltools)
SubRMSE_train<- mltools::rmse(preds  = Subset_lm_Train,
              actuals = M4College_train$Grad.Rate,
              weights = 1,
              na.rm   = FALSE)
cat("\n Root mean square error of best fit of train dataset")
```

```

 Root mean square error of best fit of train dataset
```

Hide

```
SubRMSE_train
```

```
[1] 12.71618
```

Hide

```
Subset_lm_Test <- predict.lm(fit_subset, newdata = M4College_test)

cat("\n Root mean square error of best fit of test dataset")
```

```

 Root mean square error of best fit of test dataset
```

Hide

```
SubRMSE_test <- mltools::rmse(preds = Subset_lm_Test,
              actuals = M4College_test$Grad.Rate,
              weights = 1,
              na.rm   = FALSE)



SubRMSE_test
```

```
[1] 12.76898
```

Hide

```
 In a comparison of all the three methods used, Lasso and subset used a lesser number
of variables to form a good model. The outState variable is not given a good amount o
f importance in Ridge and lasso but the Subset regression mentions that a good model
can be created using only Outstate Variable with 'Grad Rate'. However, practically it
may not be correct to predict and improve business based only on one variable. On the
other hand, Lasso gives more importance to 'Pct. new students from top 25% of H.S. cl
ass' and 'Pct. alumni who donate'. These are the area of concentration that all the i
nstitutions need improvement to increase the Graduation rate.
```

Conclusion

```
 A variety of models fit using various methods such as: Ridge, Lasso and  best subset
s  We found that the performance from best to worst was:
```

1. Linear model : Train - 12.71618 Test - 12.76898
2. Lasso model : Train - 13.37291 Test - 13.37291
3. Ridge model : Train - 13.43388 Test - 13.37291

   From the above results, we can conclude that Linear has the least mean square and results in the best model inclusive of all the 16 variables that impact the Graduation rate of the education institution. Thus to improve the Graduation rate, the institution can concentrate on all the areas.

Reference

1. Best Subsets Regression Essentials in R Kassambara & Sfd http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/155-best-subsets-regression-essentials-in-r/ (http://www.sthda.com/english/articles/37-model-selection-essentials-in-r/155-best-subsets-

regression-essentials-in-r/)

2. Deepika Singh Singh https://www.pluralsight.com/guides/linear-lasso-and-ridge-regression-with-r (https://www.pluralsight.com/guides/linear-lasso-and-ridge-regression-with-r)
3. Ridge and Lasso Regression Models http://wavedatalab.github.io/machinelearningwithr/post4.html (http://wavedatalab.github.io/machinelearningwithr/post4.html)