# CS 5565

LAB5

(Subset Selection, Ridge and Lasso, PCR and PLS)

Shubhabrata Mukherjee

Id: 16201097

2)a)

Console Terminal Jobs

C:/Users/shubh/Downloads/Shubh_myR_proj/

```
> set.seed(1)
> X=rnorm(100)
> X
  [1] -0.626453811  0.183643324 -0.835628612  1.595280802  0.329507772 -0.820468384  0.487429052  0.738324705
  [9]  0.575781352 -0.305388387  1.511781168  0.389843236 -0.621240581 -2.214699887  1.124930918 -0.044933609
 [17] -0.016190263  0.943836211  0.821221195  0.593901321  0.918977372  0.782136301  0.074564983 -1.989351696
 [25]  0.619825748 -0.056128740 -0.155795507 -1.470752384 -0.478150055  0.417941560  1.358679552 -0.102787727
 [33]  0.387671612 -0.053805041 -1.377059557 -0.414994563 -0.394289954 -0.059313397  1.100025372  0.763175748
 [41] -0.164523596 -0.253361680  0.696963375  0.556663199 -0.688755695 -0.707495157  0.364581962  0.768532925
 [49] -0.112346212  0.881107726  0.398105880 -0.612026393  0.341119691 -1.129363096  1.433023702  1.980399899
 [57] -0.367221476 -1.044134626  0.569719627 -0.135054604  2.401617761 -0.039240003  0.689739362  0.028002159
 [65] -0.743273209  0.188792300 -1.804958629  1.465554862  0.153253338  2.172611670  0.475509529 -0.709946431
 [73]  0.610726353 -0.934097632 -1.253633400  0.291446236 -0.443291873  0.001105352  0.074341324 -0.589520946
 [81] -0.568668733 -0.135178615  1.178086997 -1.523566800  0.593946188  0.332950371  1.063099837 -0.304183924
 [89]  0.370018810  0.267098791 -0.542520031  1.207867806  1.160402616  0.700213650  1.586833455  0.558486426
 [97] -1.276592208 -0.573265414 -1.224612615 -0.473400636
> e = rnorm(100)
> e
  [1] -0.62036668  0.04211587 -0.91092165  0.15802877 -0.65458464  1.76728727  0.71670748  0.91017423
  [9]  0.38418536  1.68217608 -0.63573645 -0.46164473  1.43228224 -0.65069635 -0.20738074 -0.39280793
 [17] -0.31999287 -0.27911330  0.49418833 -0.17733048 -0.50595746  1.34303883 -0.21457941 -0.17955653
 [25] -0.10019074  0.71266631 -0.07356440 -0.03763417 -0.68166048 -0.32427027  0.06016044 -0.58889449
 [33]  0.53149619 -1.51839408  0.30655786 -1.53644982 -0.30097613 -0.52827990 -0.65209478 -0.05689678
 [41] -1.91435943  1.17658331 -1.66497244 -0.46353040 -1.11592011 -0.75081900  2.08716655  0.01739562
 [49] -1.28630053 -1.64060553  0.45018710 -0.01855983 -0.31806837 -0.92936215 -1.48746031 -1.07519230
 [57]  1.00002880 -0.62126669 -1.38442685  1.86929062  0.42510038 -0.23864710  1.05848305  0.88642265
 [65] -0.61924305  2.20610246 -0.25502703 -1.42449465 -0.14439960  0.20753834  2.30797840  0.10580237
 [73]  0.45699881 -0.07715294 -0.33400084 -0.03472603  0.78763961  2.07524501  1.02739244  1.20790840
 [81] -1.23132342  0.98389557  0.21992480 -1.46725003  0.52102274 -0.15875460  1.46458731 -0.76608200
 [89] -0.43021175 -0.92610950 -0.17710396  0.40201178 -0.73174817  0.83037317 -1.20808279 -1.04798441
 [97]  1.44115771 -1.01584747  0.41197471 -0.38107605
```

b)

Console Terminal Jobs

C:/Users/shubh/Downloads/Shubh_myR_proj/

```
> beta0 = 1
> beta1 = 2
> beta2 = 3
> beta3 = 4
> Y = beta0 + beta1*X + beta2*X^2 + beta3*X^3 + e
> Y
  [1]  -0.67933427   1.53535052  -1.82134821  28.22280631   1.47326273   0.93660215   3.86755421   6.63210596
  [9]   4.29386228   2.23726090  24.06486920   2.01096498   1.38857465 -32.81689620  12.53315319   0.52301905
 [17]   0.64839600   8.64421783   7.37518374   3.90654909   7.96993242   7.65637047   0.95288867 -22.77729726
 [25]   4.24452106   1.60915281   0.67253528  -8.21541312  -0.38935112   2.32765431  19.34809474   0.23288207
 [33]   2.99075880  -0.61794227  -6.20391590  -1.13565975   0.13164520   0.36281286  11.50249183   5.99477424
 [41]  -1.18001588   1.79738106   3.54045015   3.26939937  -1.37721812  -1.08070622   4.40893144   6.14210396
 [49]  -0.47879993   6.18685724   2.97424422  -0.03588604   1.87203330  -4.12353968  20.31044089  46.71994372
 [57]   1.47205906  -2.99221837   2.46843319   2.64404721  78.93953450   0.68725054   6.17773043   1.94486716
 [65]  -1.09092447   3.71753080 -16.61264041  21.54134244   1.24696442  60.73449291   5.36739392  -0.23333866
 [73]   4.70858273  -1.58787720  -5.00730201   1.90201200   1.14213787   3.07745938   2.19429841   1.25195482
 [81]  -1.13410227   1.75847750  14.27998151 -10.69696930   4.60534185   1.98735211  12.78731285  -0.20944822
 [89]   1.92321053   0.89833457  -0.01787538  15.24343033  12.87874712   6.07495457  26.50254843   2.70149320
 [97]  -3.54475002  -0.93005486  -3.88431122  -0.07992462
```

c)

C:/Users/shubh/Downloads/Shubh_myR_proj/

```
> library(leaps)
> library(ISLR)
> mydata = data.frame(Y,X)
> myreg=regsubsets(Y ~ poly(X, 10), data = mydata)
> summary(myreg)
Subset selection object
Call: regsubsets.formula(Y ~ poly(X, 10), data = mydata)
10 Variables  (and intercept)
              Forced in Forced out
poly(X, 10)1      FALSE      FALSE
poly(X, 10)2      FALSE      FALSE
poly(X, 10)3      FALSE      FALSE
poly(X, 10)4      FALSE      FALSE
poly(X, 10)5      FALSE      FALSE
poly(X, 10)6      FALSE      FALSE
poly(X, 10)7      FALSE      FALSE
poly(X, 10)8      FALSE      FALSE
poly(X, 10)9      FALSE      FALSE
poly(X, 10)10     FALSE      FALSE
1 subsets of each size up to 8
Selection Algorithm: exhaustive
         poly(X, 10)1 poly(X, 10)2 poly(X, 10)3 poly(X, 10)4 poly(X, 10)5 poly(X, 10)6 poly(X, 10)7
1  ( 1 ) "*"          " "          " "          " "          " "          " "          " "
2  ( 1 ) "*"          " "          "*"          " "          " "          " "          " "
3  ( 1 ) "*"          "*"          "*"          " "          " "          " "          " "
4  ( 1 ) "*"          "*"          "*"          " "          "*"          " "          " "
5  ( 1 ) "*"          "*"          "*"          "*"          "*"          " "          " "
6  ( 1 ) "*"          "*"          "*"          "*"          "*"          " "          " "
7  ( 1 ) "*"          "*"          "*"          "*"          "*"          " "          "*"
8  ( 1 ) "*"          "*"          "*"          "*"          "*"          " "          "*"
         poly(X, 10)8 poly(X, 10)9 poly(X, 10)10
1  ( 1 ) " "          " "          " "
2  ( 1 ) " "          " "          " "
3  ( 1 ) " "          " "          " "
4  ( 1 ) " "          " "          " "
5  ( 1 ) " "          " "          " "
6  ( 1 ) " "          " "          "*"
7  ( 1 ) " "          " "          "*"
8  ( 1 ) " "          "*"          "*"
```

Getting summary with respect to rss, cp, bic:

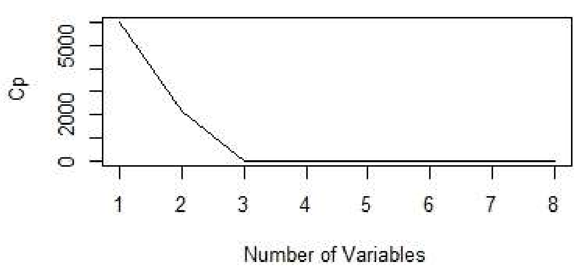
C:/Users/shubh/Downloads/Shubh_myR_proj/

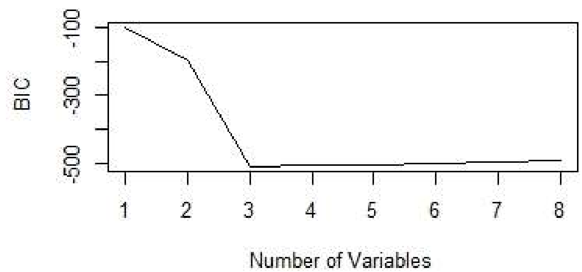
```
> myreg.sum = summary(myreg)
> myreg.sum$rss
[1] 5736.75236 2117.90448   88.95936   86.76840   85.18811   84.28327   84.17453   84.08700
> myreg.sum$cp
[1] 5977.801146 2148.336750    2.185943    1.866261    2.193128    3.235128    5.119994    7.027330
> myreg.sum$bic
[1] -102.2028 -197.2442 -509.6393 -507.5279 -504.7607 -501.2234 -496.7474 -492.2462
```

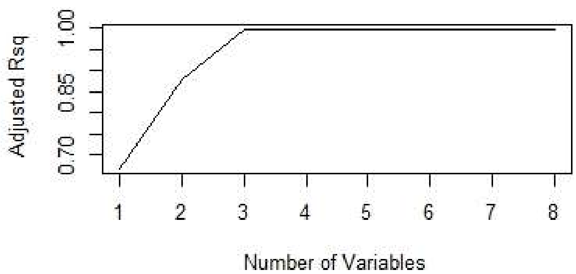
Now plotting them we get:

C:/Users/shubh/Downloads/Shubh_myR_proj/

```
> par(mfrow=c(2,2))
> plot(myreg.sum$cp ,xlab=" Number of Variables ",ylab=" Cp",type="l")
> plot(myreg.sum$bic ,xlab=" Number of Variables ",ylab=" BIC",type="l")
> plot(myreg.sum$adjr2 ,xlab =" Number of Variables ",ylab=" Adjusted Rsq",type="l")
> 
```




So, from the plots we see 3 variable model is the best.

Now we get those coefficients are:

```
C:/Users/shubh/Downloads/Shubh_myR_proj/
> coef(myreg,3)
 (Intercept) poly(X, 10)1 poly(X, 10)2 poly(X, 10)3
    4.454162   108.363598    45.043813    60.156861
>
```

Intercept: 4.454162

Poly(X,10)1: 108.363598

Poly(X,10)2: 45.043813

Poly(X,10)3: 60.156861

d)

Now, comparing this with forward and backward stepwise selection we get:

Firstly forward stepwise model:

```
C:/Users/shubh/Downloads/Shubh_myR_proj/
> myreg.fwd=regsubsets(Y ~ poly(X, 10), data = mydata,method = "forward")
> myreg.fwd.sum = summary(myreg.fwd)
> par(mfrow=c(2,2))
> plot(myreg.fwd.sum$cp ,xlab=" Number of Variables ",ylab=" Cp",type="l")
> plot(myreg.fwd.sum$bic ,xlab=" Number of Variables ",ylab=" BIC",type="l")
> plot(myreg.fwd.sum$adjr2 ,xlab =" Number of Variables ",ylab=" Adjusted Rsq",type="l")
> coef(myreg.fwd,3)
 (Intercept) poly(X, 10)1 poly(X, 10)2 poly(X, 10)3
    4.454162   108.363598    45.043813    60.156861
```

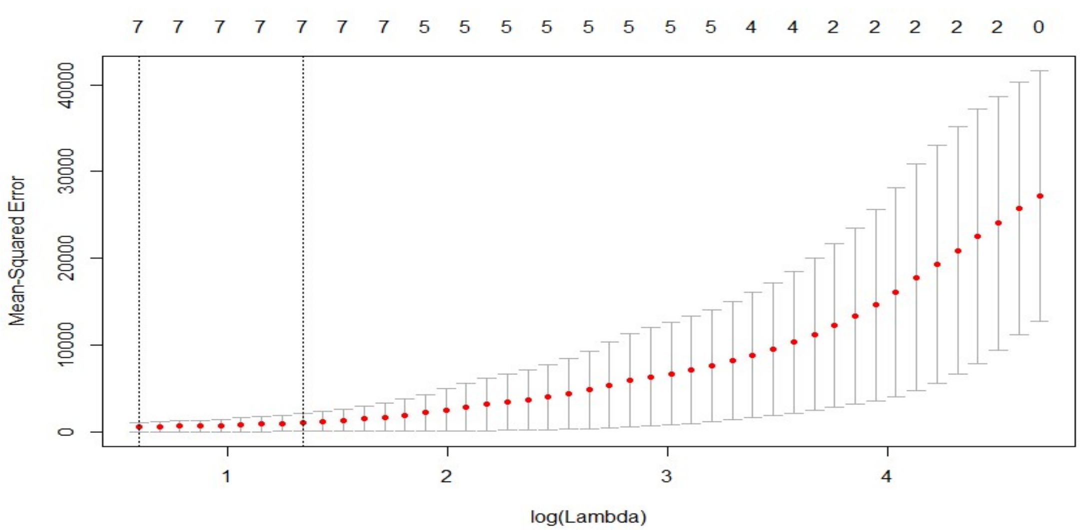
Secondly forward stepwise model:

```
Console  Terminal  Jobs
C:/Users/shubh/Downloads/Shubh_myR_proj/
> myreg.bwd=regsubsets(Y ~ poly(X, 10), data = mydata,method = "backward")
> myreg.bwd.sum = summary(myreg.bwd)
> par(mfrow=c(2,2))
> plot(myreg.bwd.sum$cp ,xlab=" Number of Variables ",ylab=" Cp",type="l")
> plot(myreg.bwd.sum$bic ,xlab=" Number of Variables ",ylab=" BIC",type="l")
> plot(myreg.bwd.sum$adjr2 ,xlab =" Number of Variables ",ylab=" Adjusted Rsq",type="l")
> coef(myreg.bwd,3)
 (Intercept) poly(X, 10)1 poly(X, 10)2 poly(X, 10)3
    4.454162   108.363598    45.043813    60.156861
```

So we see all options gave same results in this case.

e) Now performing LASSO we get:

```
E:/shubh_projects/
> library(glmnet)
> mylasso = model.matrix(Y~poly(X,10),data=newdata)[,-1]
> cv.mylasso=cv.glmnet(mylasso,Y,alpha=1)
> plot(cv.mylasso)
```



Now, checking the performance of lasso we get:

E:/shubh_projects/

```
> lamd=cv.mylasso$lambda.min
> lamd
[1] 1.816716
> fit.mylasso=glmnet(mylasso,Y,alpha=1)
> predict(fit.mylasso,s=lamd,type="coefficients")
11 x 1 sparse Matrix of class "dgCMatrix"
                        1
(Intercept)     -4.484505
poly(X, 10)1  1070.925191
poly(X, 10)2  -410.081152
poly(X, 10)3  1014.779883
poly(X, 10)4  -271.132920
poly(X, 10)5   381.774055
poly(X, 10)6   -39.014695
poly(X, 10)7    41.506398
poly(X, 10)8     .
poly(X, 10)9     .
poly(X, 10)10    .
```

So according to lasso, the best fit model will contain $X$,$X^2$,$X^3$,$X^4$,$X^5$ ,$X^6$ , $X^7$ variables with the above coefficients.

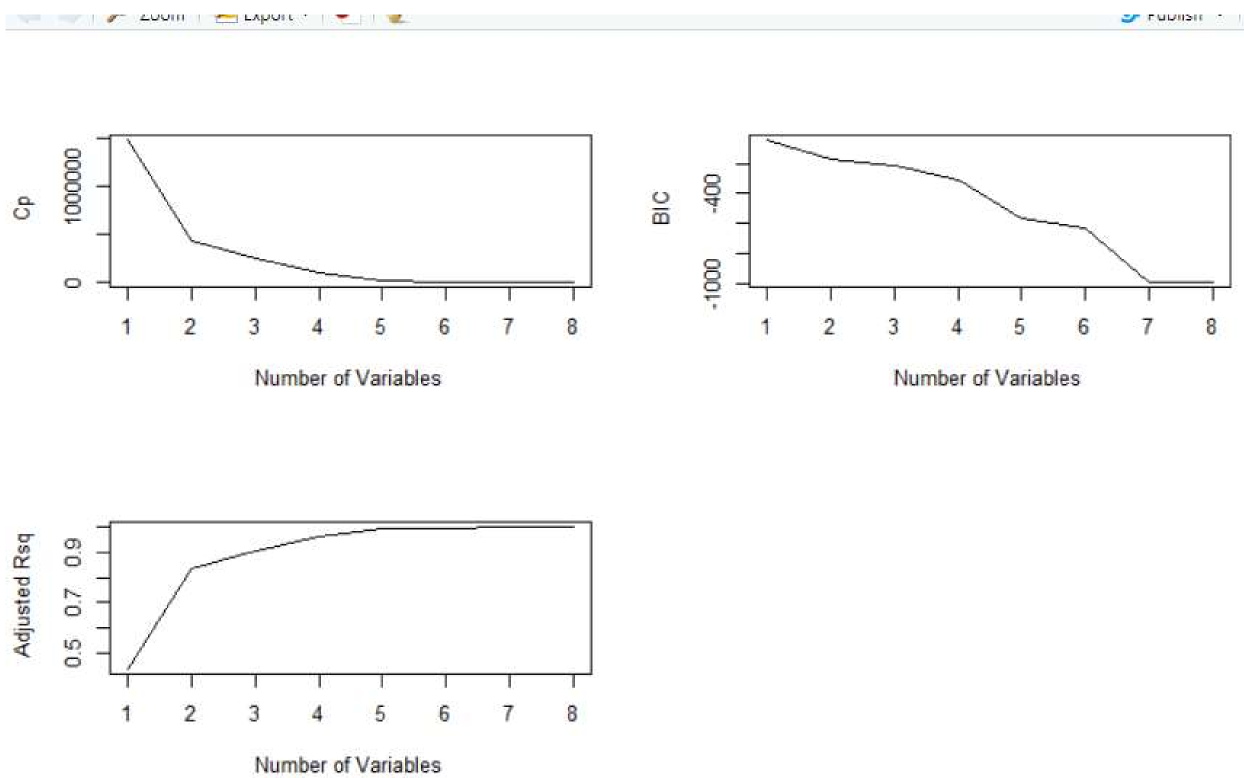
f) Performing the best subset selection, we get:

Console Terminal Jobs

C:/Users/shubh/Downloads/Shubh_myR_proj/

```
> set.seed(2)
> X=rnorm(100)
> #X
> e = rnorm(100)
> #e
> beta0 = 3
> beta7 = 2
> Y = beta0 + beta7*X^7 + e
> #Y
>
> newdata = data.frame(Y,X)
> newreg=regsubsets(Y ~ poly(X, 10), data = newdata)
> newreg.sum = summary(newreg)
> par(mfrow=c(2,2))
> plot(newreg.sum$cp ,xlab=" Number of Variables ",ylab=" Cp",type="l")
> plot(newreg.sum$bic ,xlab=" Number of Variables ",ylab=" BIC",type="l")
> plot(newreg.sum$adjr2 ,xlab =" Number of Variables ",ylab=" Adjusted Rsq",type="l")
```

Comparing $R^2$ , Cp and BIC we get the below curves:




So, this gives optimum result for 7 variable model. Now coefficients, we get:

```
E:/shubh_projects/
> coef(newreg,7)
 (Intercept) poly(X, 10)1 poly(X, 10)2 poly(X, 10)3 poly(X, 10)4 poly(X, 10)5 poly(X, 10)6 poly(X, 10)7
   -4.484505  1089.092347  -428.248307  1032.947039  -289.300075   399.941210   -57.181850    59.673553
>
```

3) a) Dividing college in training and test data set we get:

```
E:/shubh_projects/
> traindata = sample(1:dim(College)[1], dim(College)[1]/2)
> testdata=(-traindata)
> College.traindata = College[traindata,]
> College.traindata = College[testdata,]
```

b) Fitting a linear model using least squares we get:

```
E:/shubh_projects/
> fit.clg = lm(Apps~.,data=College.traindata)
> pred.clg = predict(fit.clg,College.testdata)
> mean((pred.clg - College.testdata$Apps)^2)
[1] 983470
> m1= mean((pred.clg - College.testdata$Apps)^2)
> m1
[1] 983470
```

c) Fitting using Ridge regression we get:

E:/shubh_projects/

```
> train.rid = model.matrix(Apps ~ ., data = College.traindata)
> test.rid = model.matrix(Apps ~ ., data = College.testdata)
> grid = 10 ^ seq(4, -2, length = 100)
> fit.rid = glmnet(train.rid, College.traindata$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
> cv.rid = cv.glmnet(train.rid, College.traindata$Apps, alpha = 0, lambda = grid, thresh = 1e-12)
> lamd.rid = cv.rid$lambda.min
> lamd.rid
[1] 14.17474
>
> pred.rid = predict(fit.rid, s = lamd.rid, newx = test.rid)
> mean((pred.rid - College.testdata$Apps)^2)
[1] 985106.2
> m2= mean((pred.rid - College.testdata$Apps)^2)
> m2
[1] 985106.2
```

E:/shubh_projects/

```
> difmethod = ((m2-m1)/m2)*100
> difmethod
[1] 0.1660945
>
```

So, we see that error reported is very similar by both method.

d) Using Lasso we get:

E:/shubh_projects/

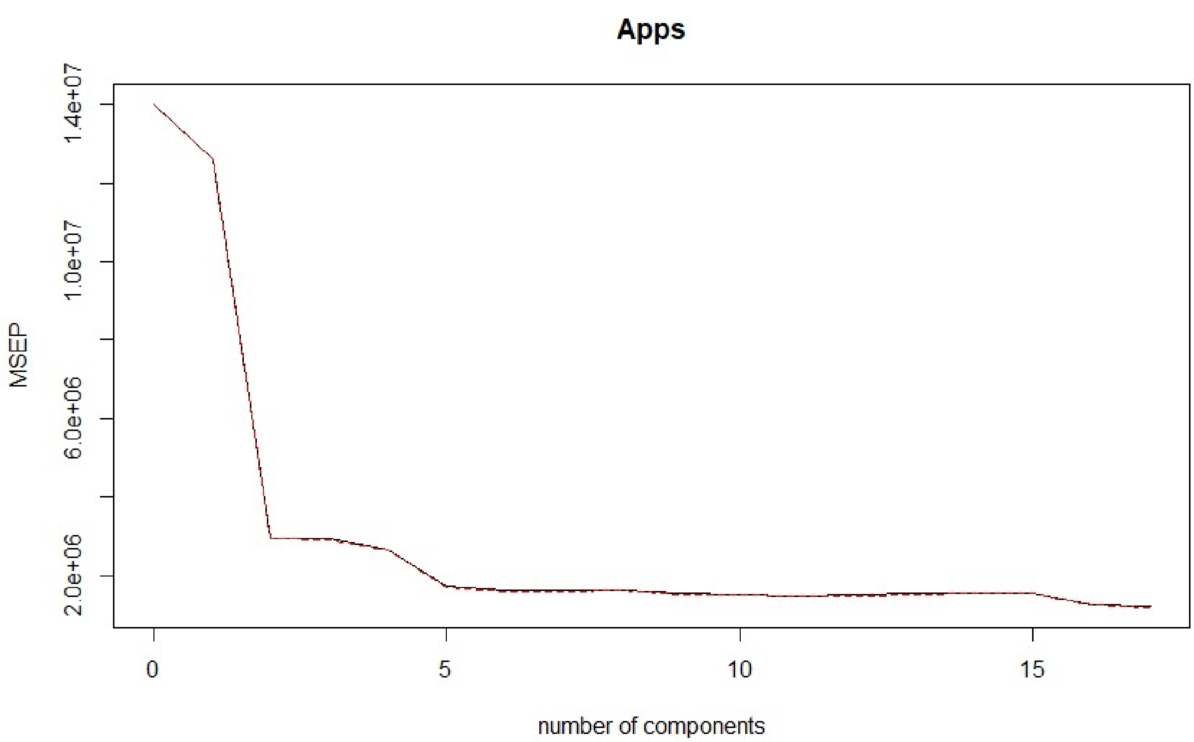
```
> pred.lasso = predict(fit.lasso, s = lamd.lasso, newx = test.rid)
> mean((pred.lasso - College.testdata$Apps)^2)
[1] 1014782
> predict(fit.lasso, s = lamd.lasso, type = "coefficients")
19 x 1 sparse Matrix of class "dgCMatrix"
                       1
(Intercept) -850.22046355
(Intercept)    .
PrivateYes  -594.04804873
Accept         1.17037074
Enroll         .
Top10perc     33.13315726
Top25perc      .
F.Undergrad    0.08392472
P.Undergrad    .
Outstate      -0.01120011
Room.Board     0.16325466
Books          .
Personal       0.02745913
PhD           -9.61493809
Terminal       .
S.F.Ratio      1.30849037
perc.alumni   -8.40210895
Expend         0.05624534
Grad.Rate      6.18801183
```

e) Fitting the PCR model we get:

```
E:/shubh_projects/
> library(pls)
> fit.pcr = pcr(Apps ~ ., data = College.traindata, scale = TRUE, validation = "CV")
> validationplot(fit.pcr, val.type = "MSEP")
> summary(fit.pcr)
Data:   X dimension: 389 17
        Y dimension: 389 1
Fit method: svdpc
Number of components considered: 17

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
CV            3742     3550     1712     1710     1624     1310     1269     1266     1268     1235      1223      1218
adjCV         3742     3553     1711     1709     1626     1304     1262     1262     1264     1233      1220      1214
       12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
CV         1223      1238      1240      1247      1119      1095
adjCV      1219      1234      1236      1242      1113      1090

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
X       31.49    57.21    64.62    70.50    75.76    80.34    84.18    87.52    90.46     93.00     94.99     96.77
Apps    10.05    79.42    79.59    81.97    88.66    89.27    89.27    89.27    89.92     90.33     90.61     90.61
      13 comps  14 comps  15 comps  16 comps  17 comps
X        97.96     98.87     99.41     99.83    100.00
Apps     90.61     90.62     90.63     92.41     92.94
> pred.pcr = predict(fit.pcr, College.testdata, ncomp = 17)
> mean((pred.pcr - College.testdata$Apps)^2)
[1] 983470
> m3=  mean((pred.pcr - College.testdata$Apps)^2)
```



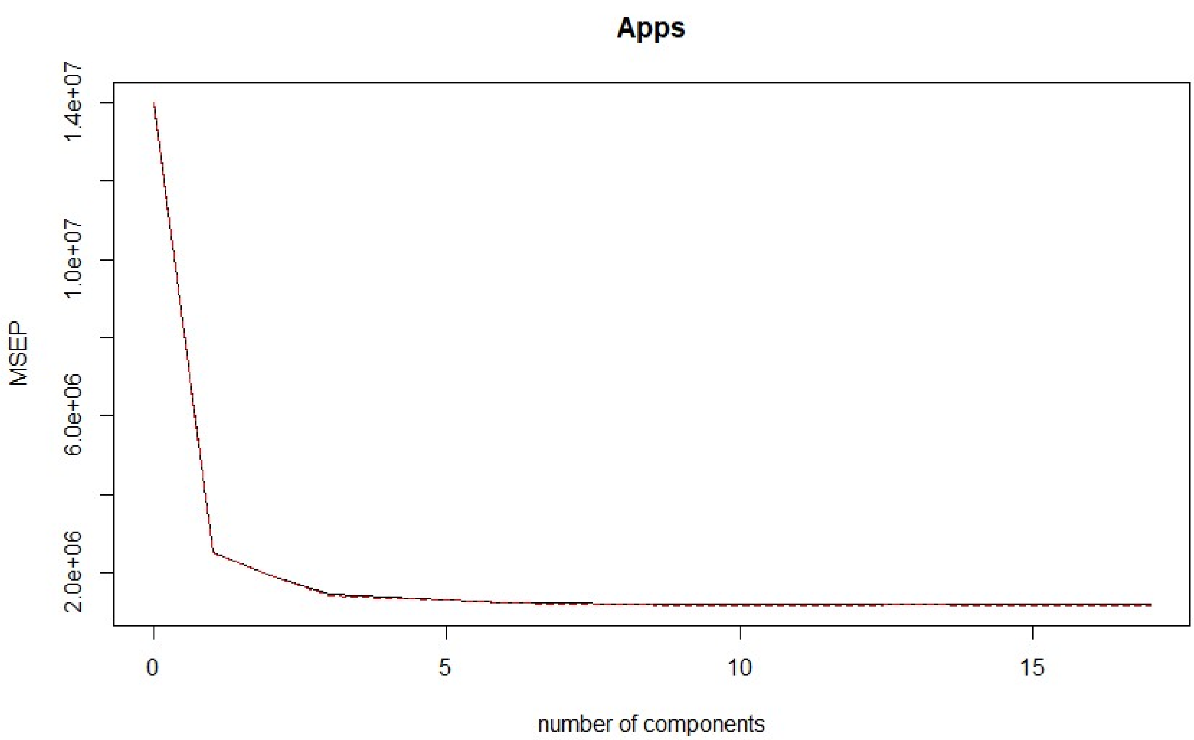

So the minimum MSEP is obtained M=17, The MSE from PCR is similar as least square regression model.

f) Fitting the PLS model we get:

```
E:/shubh_projects/
> fit.pls = plsr(Apps ~ ., data = College.traindata, scale = TRUE, validation = "CV")
> validationplot(fit.pls, val.type = "MSEP")
> summary(fit.pls)
Data:   X dimension: 389 17
        Y dimension: 389 1
Fit method: kernelpls
Number of components considered: 17

VALIDATION: RMSEP
Cross-validated using 10 random segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps
CV            3742     1588     1384     1196     1164     1142     1113     1099     1091     1086      1085      1087
adjCV         3742     1585     1384     1192     1159     1137     1108     1094     1086     1081      1080      1082
       12 comps  13 comps  14 comps  15 comps  16 comps  17 comps
CV         1087      1088      1087      1087      1087      1087
adjCV      1082      1082      1082      1082      1082      1082

TRAINING: % variance explained
      1 comps  2 comps  3 comps  4 comps  5 comps  6 comps  7 comps  8 comps  9 comps  10 comps  11 comps  12 comps
X       26.63    48.92    62.48    66.23    70.91    73.90    78.36    81.46    84.94     86.14     88.31     89.44
Apps    82.95    87.30    90.77    91.52    92.05    92.64    92.80    92.84    92.86     92.93     92.94     92.94
      13 comps  14 comps  15 comps  16 comps  17 comps
X        91.99     93.72     96.32     98.22    100.00
Apps     92.94     92.94     92.94     92.94     92.94
> pred.pls = predict(fit.pls, College.testdata, ncomp = 11)
> mean((pred.pls - College.testdata$Apps)^2)
[1] 983720.9
> m4=  mean((pred.pls - College.testdata$Apps)^2)
```




We see the minimum MSEP is obtained at M=11

The test MSE from PLS is more than least square regression model.

g) Now comparing all the model we get:

```
E:/shubh_projects/
> test.avg = mean(College.testdata$Apps)
> r2.lm=1 - mean((pred.clg - College.testdata$Apps)^2) / mean((test.avg - College.testdata$Apps)^2)
> r2.lm
[1] 0.929389
> r2.ridge= 1 - mean((pred.rid - College.testdata$Apps)^2) / mean((test.avg - College.testdata$Apps)^2)
> r2.ridge
[1] 0.9292715
> r2.lasso= 1 - mean((pred.lasso - College.testdata$Apps)^2) / mean((test.avg - College.testdata$Apps)^2)
> r2.lasso
[1] 0.9271409
> r2.pcr=1 - mean((pred.pcr - College.testdata$Apps)^2) / mean((test.avg - College.testdata$Apps)^2)
> r2.pcr
[1] 0.929389
> r2.pls= 1 - mean((pred.pls - College.testdata$Apps)^2) / mean((test.avg - College.testdata$Apps)^2)
> r2.pls
[1] 0.929371
```

So all models generate very similar R^2 value, only Lasso is a bit less compared to others.

4)a)

Generating a data set with p = 20 features, n = 1, 000 observations, and an associated quantitative response vector generated according to the model we get:

```
E:/shubh_projects/
> b[2] = 0
> b[4] = 0
> b[7] = 0
> b[19] = 0
> set.seed(67)
> eps= rnorm(1000)
> y = x %*% b + eps
```

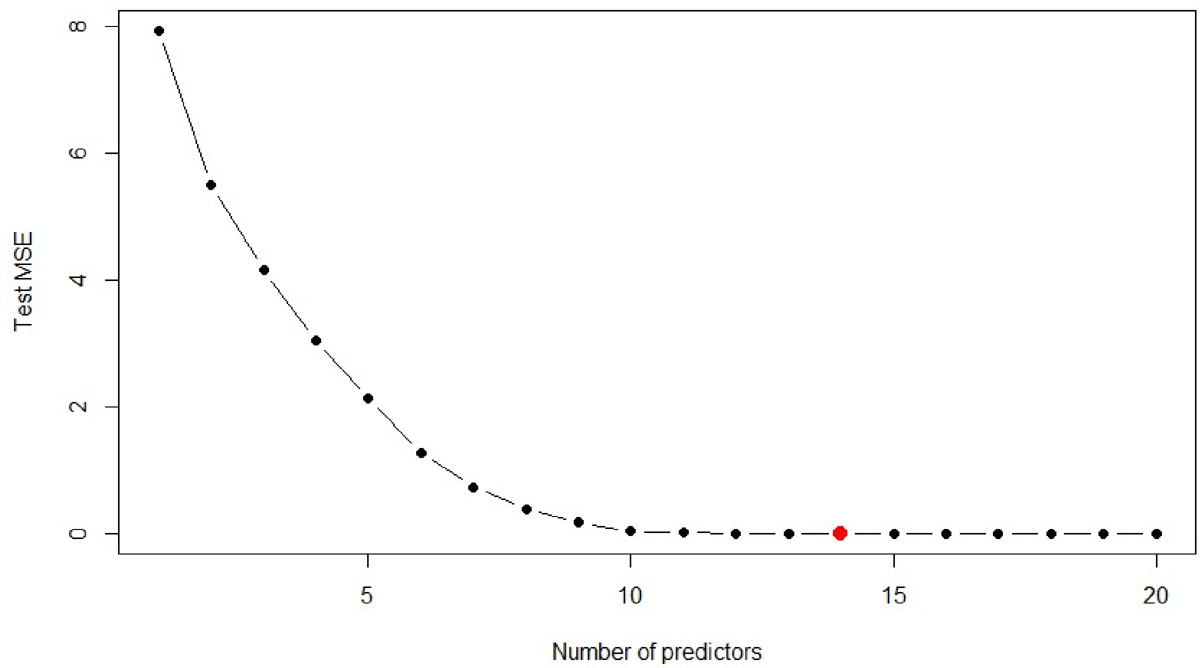
b) Splitting data set into a training set containing 100 observations and test set containing 900 observations we get:

```
>
> train= sample(seq(1000), 100, replace = FALSE)
> test = -train
> x.train = x[train, ]
> x.test = x[test, ]
> y.train = y[train]
> y.test = y[test]
```

c) Performing best subset selection on the training set we get:

```
> data.train = data.frame(y = y.train, x = x.train)
> regfit =regsubsets(y ~ ., data = data.train, nvmax = 20)
> train.mat = model.matrix(y ~ ., data = data.train, nvmax = 20)
> val.errors = rep(NA, 20)
> for (i in 1:20) {
+   coefi = coef(regfit, id = i)
+   pred = train.mat[, names(coefi)] %*% coefi
+   val.errors[i] = mean((pred - y.train)^2)
+ }
> plot(val.errors, xlab = "Number of predictors", ylab = "Training MSE", pch = 19, type = "b")
>
> points(which.min(val.errors), val.errors[which.min(val.errors)], col = "red", cex = 2, pch = 20)
>
```

d) Plotting the test set MSE associated with the best model of each size we see:




e) Test MSE is minimum for 14 number of predictors

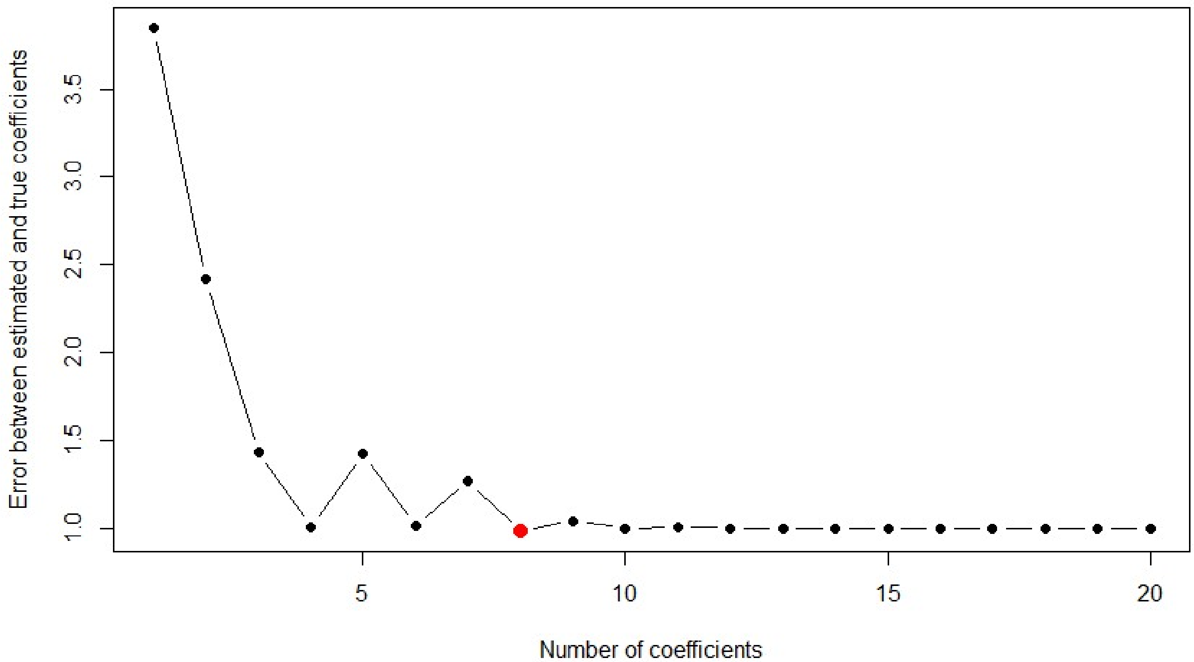
f)

```
> coef(regfit, which.min(val.errors))
  (Intercept)          x.1          x.2          x.3          x.5          x.6          x.8         x.10
 8.179221e-16  2.219402e+00  5.828671e-16 -9.574989e-01 -1.143411e+00 -1.057188e+00  6.083412e-01 -1.537294e+00
         x.12         x.13         x.14         x.16         x.18         x.19         x.20
 4.954311e-01  9.381498e-01 -3.757353e-01 -7.095219e-01  1.315931e-01 -4.093947e-16 -9.853550e-02
```

The best model has eliminated variables corresponding to zero coefficients (b4, b7, b9, b11, b15, b17).

g)

```
>
> val.errors = rep(NA, 20)
> x_cols = colnames(x, do.NULL = FALSE, prefix = "x.")
> for (i in 1:20) {
+   coefi = coef(regfit, id = i)
+   val.errors[i] = sqrt(sum((b[x_cols %in% names(coefi)] - coefi[names(coefi) %in% x_cols])^2) + sum(b[!(x_cols %in% names(c
oefi))])^2)
+ }
> plot(val.errors, xlab = "Number of coefficients", ylab = "Error between estimated and true coefficients", pch = 19, type =
 "b")
> points(which.min(val.errors), val.errors[which.min(val.errors)], col = "red", cex = 2, pch = 20)
>
```




We observe that the error between estimated and true coefficients is minimum for a model with 8 variables. However test error was minimum for model with 14 variables.