STATISTICAL DATA MINING

HOMEWORK 2

NAME: SHREERAM G S

UB PERSON NUMBER: 50413349

1) sol: install packages of ISLR(for data sets), leaps(performing search for subset of the variables in x for predicting y in linear regression) ,glmnet (for fitting linear models)

```
install.packages('ISLR')
install.packages('leaps')
install.packages('glmnet')
```

Result of installing the packages

```
Console  Terminal  Jobs
R 4.1.1 · C:/Users/Sriram/Desktop/R projects/
> install.packages('ISLR')
Error in install.packages : Updating loaded packages

Restarting R session...

> install.packages("ISLR")
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/Sriram/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/ISLR_1.4.zip'
Content type 'application/zip' length 2923814 bytes (2.8 MB)
downloaded 2.8 MB

package 'ISLR' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Sriram\AppData\Local\Temp\RtmpklZKH2\downloaded_packages
>
> install.packages('leaps')
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/Sriram/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/leaps_3.1.zip'
Content type 'application/zip' length 103032 bytes (100 KB)
downloaded 100 KB

package 'leaps' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Sriram\AppData\Local\Temp\RtmpklZKH2\downloaded_packages
> install.packages('glmnet')
WARNING: Rtools is required to build R packages but is not currently installed. Please download and install the appropriate version of Rtools before proceeding:

https://cran.rstudio.com/bin/windows/Rtools/
Installing package into 'C:/Users/Sriram/Documents/R/win-library/4.1'
(as 'lib' is unspecified)
also installing the dependencies 'iterators', 'foreach', 'shape'

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/iterators_1.0.13.zip'
Content type 'application/zip' length 343145 bytes (335 KB)
downloaded 335 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/foreach_1.5.1.zip'
Content type 'application/zip' length 146044 bytes (142 KB)
downloaded 142 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/shape_1.4.6.zip'
Content type 'application/zip' length 789300 bytes (770 KB)
downloaded 770 KB

trying URL 'https://cran.rstudio.com/bin/windows/contrib/4.1/glmnet_4.1-2.zip'
Content type 'application/zip' length 2291869 bytes (2.2 MB)
downloaded 2.2 MB
```

```
package 'iterators' successfully unpacked and MD5 sums checked
package 'foreach' successfully unpacked and MD5 sums checked
package 'shape' successfully unpacked and MD5 sums checked
package 'glmnet' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\Sriram\AppData\Local\Temp\RtmpklZKH2\downloaded_packages
>
```

Loading cereals data set

```
R 4.1.1 · C:/Users/Sriram/Desktop/R projects/
> library(ISLR)
> library(leaps)
> library(glmnet)
> data(College)
> set.seed(1)
> var1 <- read.delim("C:/Users/Sriram/Downloads/cereal.csv",sep = ",")
> var1$name<- NULL
> var1
   mfr type calories protein fat sodium fiber carbo sugars potass vitamins shelf weight cups   rating
1    N    C       70       4   1    130  10.0   5.0       6     280       25     3   1.00 0.33 68.40297
2    Q    C      120       3   5     15   2.0   8.0       8     135        0     3   1.00 1.00 33.98368
3    K    C       70       4   1    260   9.0   7.0       5     320       25     3   1.00 0.33 59.42551
4    K    C       50       4   0    140  14.0   8.0       0     330       25     3   1.00 0.50 93.70491
5    R    C      110       2   2    200   1.0  14.0       8      -1       25     3   1.00 0.75 34.38484
6    G    C      110       2   2    180   1.5  10.5      10      70       25     1   1.00 0.75 29.50954
7    K    C      110       2   0    125   1.0  11.0      14      30       25     2   1.00 1.00 33.17409
8    G    C      130       3   2    210   2.0  18.0       8     100       25     3   1.33 0.75 37.03856
9    R    C       90       2   1    200   4.0  15.0       6     125       25     1   1.00 0.67 49.12025
10   P    C       90       3   0    210   5.0  13.0       5     190       25     3   1.00 0.67 53.31381
11   Q    C      120       1   2    220   0.0  12.0      12      35       25     2   1.00 0.75 18.04285
12   G    C      110       6   2    290   2.0  17.0       1     105       25     1   1.00 1.25 50.76500
13   G    C      120       1   3    210   0.0  13.0       9      45       25     2   1.00 0.75 19.82357
14   G    C      110       3   2    140   2.0  13.0       7     105       25     3   1.00 0.50 40.40021
15   G    C      110       1   1    180   0.0  12.0      13      55       25     2   1.00 1.00 22.73645
16   R    C      110       2   0    280   0.0  22.0       3      25       25     1   1.00 1.00 41.44502
17   K    C      100       2   0    290   1.0  21.0       2      35       25     1   1.00 1.00 45.86332
18   K    C      110       1   0     90   1.0  13.0      12      20       25     2   1.00 1.00 35.78279
19   G    C      110       1   1    180   0.0  12.0      13      65       25     2   1.00 1.00 22.39651
20   K    C      110       3   3    140   4.0  10.0       7     160       25     3   1.00 0.50 40.44877
21   N    H      100       3   0     80   1.0  21.0       0      -1        0     2   1.00 1.00 64.53382
22   K    C      110       2   0    220   1.0  21.0       3      30       25     3   1.00 1.00 46.89564
23   G    C      100       2   1    140   2.0  11.0      10     120       25     3   1.00 0.75 36.17620
24   R    C      100       2   0    190   1.0  18.0       5      80       25     3   1.00 0.75 44.33086
25   K    C      110       2   1    125   1.0  11.0      13      30       25     2   1.00 1.00 32.20758
26   K    C      110       1   0    200   1.0  14.0      11      25       25     1   1.00 0.75 31.43597
27   K    C      100       3   0      0   3.0  14.0       7     100       25     2   1.00 0.80 58.34514
28   P    C      120       3   2    160   5.0  12.0      10     200       25     3   1.25 0.67 40.91705
29   K    C      120       3   0    240   5.0  14.0      12     190       25     3   1.33 0.67 41.01549
30   P    C      110       1   1    135   0.0  13.0      12      25       25     2   1.00 0.75 28.02576
31   P    C      100       2   0     45   0.0  11.0      15      40       25     1   1.00 0.88 35.25244
32   G    C      110       1   1    280   0.0  15.0       9      45       25     2   1.00 0.75 23.80404
33   P    C      100       3   1    140   3.0  15.0       5      85       25     3   1.00 0.88 52.07690
34   P    C      110       3   0    170   3.0  17.0       3      90       25     3   1.00 0.25 53.37101
35   P    C      120       3   3     75   3.0  13.0       4     100       25     3   1.00 0.33 45.81172
36   Q    C      120       1   2    220   1.0  12.0      11      45       25     2   1.00 1.00 21.87129
37   G    C      110       3   1    250   1.5  11.5      10      90       25     1   1.00 0.75 31.07222
38   P    C      110       1   0    180   0.0  14.0      11      35       25     1   1.00 1.33 28.74241
39   K    C      110       2   1    170   1.0  17.0       6      60      100     3   1.00 1.00 36.52368
40   K    C      140       3   1    170   2.0  20.0       9      95      100     3   1.30 0.75 36.47151
41   G    C      110       2   1    260   0.0  21.0       3      40       25     2   1.00 1.50 39.24111
42   Q    C      100       4   2    150   2.0  12.0       6      95       25     2   1.00 0.67 45.32807
43   G    C      110       2   1    180   0.0  12.0      12      55       25     2   1.00 1.00 26.73451
44   A    H      100       4   1      0   0.0  16.0       3      95       25     2   1.00 1.00 54.85092
45   R    C      150       4   3     95   3.0  16.0      11     170       25     3   1.00 1.00 37.13686
46   R    C      150       4   3    150   3.0  16.0      11     170       25     3   1.00 1.00 34.13976
47   K    C      160       3   2    150   3.0  17.0      13     160       25     3   1.50 0.67 30.31335
48   G    C      100       2   1    220   2.0  15.0       6      90       25     1   1.00 1.00 40.10596
```

a) Now we have to split the data into test and train into partitions of 20% and 80% respectively.

- First create a sample space which will be later used to divide the data sets into 80:20 ratio of train and test

```
smp_size <- floor(0.8 * nrow(var1))
smp_size
```

- Next we use the sample space to make a training and a test data set partition

```
#here we are setting the seed to make our partition reproducible
set.seed(123)
train_ind <- sample(seq_len(nrow(var1)), size = smp_size)

train <-var1[train_ind, ]
test <- var1[-train_ind, ]
test
train
```

The result of this will give us the following data sets:

- Test data set

```
> test
   mfr type calories protein fat sodium fiber carbo sugars potass vitamins shelf weight cups   rating
2    Q    C      120       3   5     15   2.0   8.0      8    135        0     3   1.00 1.00 33.98368
3    K    C       70       4   1    260   9.0   7.0      5    320       25     3   1.00 0.33 59.42551
4    K    C       50       4   0    140  14.0   8.0      0    330       25     3   1.00 0.50 93.70491
11   Q    C      120       1   2    220   0.0  12.0     12     35       25     2   1.00 0.75 18.04285
22   K    C      110       2   0    220   1.0  21.0      3     30       25     3   1.00 1.00 46.89564
24   R    C      100       2   0    190   1.0  18.0      5     80       25     3   1.00 0.75 44.33086
33   P    C      100       3   1    140   3.0  15.0      5     85       25     3   1.00 0.88 52.07690
35   P    C      120       3   3     75   3.0  13.0      4    100       25     3   1.00 0.33 45.81172
37   G    C      110       3   1    250   1.5  11.5     10     90       25     1   1.00 0.75 31.07222
40   K    C      140       3   1    170   2.0  20.0      9     95      100     3   1.30 0.75 36.47151
46   R    C      150       4   3    150   3.0  16.0     11    170       25     3   1.00 1.00 34.13976
47   K    C      160       3   2    150   3.0  17.0     13    160       25     3   1.50 0.67 30.31335
58   Q    H      100       5   2      0   2.7  -1.0     -1    110        0     1   1.00 0.67 50.82839
59   K    C      120       3   1    210   5.0  14.0     12    240       25     2   1.33 0.75 39.25920
60   G    C      100       3   2    140   2.5  10.5      8    140       25     3   1.00 0.50 39.70340
61   K    C       90       2   0      0   2.0  15.0      6    110       25     3   1.00 0.50 55.33314
66   N    C       90       3   0      0   3.0  20.0      0    120        0     1   1.00 0.67 72.80179
68   K    C      110       6   0    230   1.0  16.0      3     55       25     1   1.00 1.00 53.13132
72   G    C      100       3   1    200   3.0  16.0      3    110      100     3   1.00 1.00 46.65884
77   G    C      110       2   1    200   1.0  16.0      8     60       25     1   1.00 0.75 36.18756
```

- Training data set

```
> train
   mfr type calories protein fat sodium fiber carbo sugars potass vitamins shelf weight cups   rating
31   P    C      100       2   0     45   0.0  11.0     15     40       25     1   1.00 0.88 35.25244
51   K    C       90       3   0    170   3.0  18.0      2     90       25     3   1.00 1.00 59.64284
14   G    C      110       3   2    140   2.0  13.0      7    105       25     3   1.00 0.50 40.40021
67   K    C      110       2   1     70   1.0   9.0     15     40       25     2   1.00 0.75 31.23005
42   Q    C      100       4   2    150   2.0  12.0      6     95       25     2   1.00 0.67 45.32807
50   K    C      140       3   2    220   3.0  21.0      7    130       25     3   1.33 0.67 40.69232
43   G    C      110       2   1    180   0.0  12.0     12     55       25     2   1.00 1.00 26.73451
75   R    C      100       3   1    230   3.0  17.0      3    115       25     1   1.00 0.67 49.78744
25   K    C      110       2   1    125   1.0  11.0     13     30       25     2   1.00 1.00 32.20758
57   Q    C      100       4   1    135   2.0  14.0      6    110       25     3   1.00 0.50 49.51187
9    R    C       90       2   1    200   4.0  15.0      6    125       25     1   1.00 0.67 49.12025
26   K    C      110       1   0    200   1.0  14.0     11     25       25     1   1.00 0.75 31.43597
7    K    C      110       2   0    125   1.0  11.0     14     30       25     2   1.00 1.00 33.17409
73   G    C      110       2   1    250   0.0  21.0      3     60       25     3   1.00 0.75 39.10617
74   G    C      110       1   1    140   0.0  13.0     12     25       25     2   1.00 1.00 27.75330
62   R    C      110       1   0    240   0.0  23.0      2     30       25     1   1.00 1.13 41.99893
19   G    C      110       1   1    180   0.0  12.0     13     65       25     2   1.00 1.00 22.39651
36   Q    C      120       1   2    220   1.0  12.0     11     45       25     2   1.00 1.00 21.87129
70   G    C      110       2   1    200   0.0  21.0      3     35      100     3   1.00 1.00 38.83975
17   K    C      100       2   0    290   1.0  21.0      2     35       25     1   1.00 1.00 45.86332
71   G    C      140       3   1    190   4.0  15.0     14    230      100     3   1.50 1.00 28.59278
39   K    C      110       2   1    170   1.0  17.0      6     60      100     3   1.00 1.00 36.52368
53   P    C      120       3   1    200   6.0  11.0     14    260       25     3   1.33 0.67 37.84059
12   G    C      110       6   2    290   2.0  17.0      1    105       25     1   1.00 1.25 50.76500
15   G    C      110       1   1    180   0.0  12.0     13     55       25     2   1.00 1.00 22.73645
32   G    C      110       1   1    280   0.0  15.0      9     45       25     2   1.00 0.75 23.80404
64   N    C       80       2   0      0   3.0  16.0      0     95        0     1   0.83 1.00 68.23588
45   R    C      150       4   3     95   3.0  16.0     11    170       25     3   1.00 1.00 37.13686
65   N    C       90       3   0      0   4.0  19.0      0    140        0     1   1.00 0.67 74.47295
63   K    C      110       2   0    290   0.0  22.0      3     35       25     1   1.00 1.00 40.56016
41   G    C      110       2   1    260   0.0  21.0      3     40       25     2   1.00 1.50 39.24111
10   P    C       90       3   0    210   5.0  13.0      5    190       25     3   1.00 0.67 53.31381
23   G    C      100       2   1    140   2.0  11.0     10    120       25     3   1.00 0.75 36.17620
27   K    C      100       3   0      0   3.0  14.0      7    100       25     2   1.00 0.80 58.34514
49   K    C      120       2   1    190   0.0  15.0      9     40       25     2   1.00 0.67 29.92429
44   A    H      100       4   1      0   0.0  16.0      3     95       25     2   1.00 1.00 54.85092
52   G    C      130       3   2    170   1.5  13.5     10    120       25     3   1.25 0.50 30.45084
38   P    C      110       1   0    180   0.0  14.0     11     35       25     1   1.00 1.33 28.74241
69   N    C       90       2   0     15   3.0  15.0      5     90       25     2   1.00 1.00 59.36399
34   P    C      110       3   0    170   3.0  17.0      3     90       25     3   1.00 0.25 53.37101
29   K    C      120       3   0    240   5.0  14.0     12    190       25     3   1.33 0.67 41.01549
5    R    C      110       2   2    200   1.0  14.0      8     -1       25     3   1.00 0.75 34.38484
8    G    C      130       3   2    210   2.0  18.0      8    100       25     3   1.33 0.75 37.03856
54   K    C      100       3   0    320   1.0  20.0      3     45      100     3   1.00 1.00 41.50354
13   G    C      120       1   3    210   0.0  13.0      9     45       25     2   1.00 0.75 19.82357
18   K    C      110       1   0     90   1.0  13.0     12     20       25     2   1.00 1.00 35.78279
1    N    C       70       4   1    130  10.0   5.0      6    280       25     3   1.00 0.33 68.40297
56   Q    C       50       2   0      0   1.0  10.0      0     50        0     3   0.50 1.00 63.00565
76   G    C      100       3   1    200   3.0  17.0      3    110       25     1   1.00 1.00 51.59219
30   P    C      110       1   1    135   0.0  13.0     12     25       25     2   1.00 0.75 28.02576
6    G    C      110       2   2    180   1.5  10.5     10     70       25     1   1.00 0.75 29.50954
21   N    H      100       3   0     80   1.0  21.0      0     -1        0     2   1.00 1.00 64.53382
55   Q    C       50       1   0      0   0.0  13.0      0     15        0     3   0.50 1.00 60.75611
48   G    C      100       2   1    220   2.0  15.0      6     90       25     1   1.00 1.00 40.10596
28   P    C      120       3   2    160   5.0  12.0     10    200       25     3   1.25 0.67 40.91705
```

Now we shall apply a linear regression model to the training data set using logistic regression

```
library(caret)
model=lm(rating~.,data=train)
summary(model)
```

Library caret used for necessary predict function and lm() function is used for logistic regression. The following commands generates:

```
> library(caret)
> model=lm(rating~.,data=train)
> summary(model)

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

Residuals:
       Min         1Q     Median         3Q        Max
-5.138e-07 -1.794e-07  0.000e+00  1.945e-07  5.682e-07

Coefficients:
             Estimate Std. Error    t value Pr(>|t|)
(Intercept)  5.493e+01  6.149e-07  8.933e+07   <2e-16 ***
mfrG         1.442e-07  5.521e-07  2.610e-01    0.795
mfrK        -2.605e-08  5.934e-07 -4.400e-02    0.965
mfrN        -8.845e-08  5.463e-07 -1.620e-01    0.872
mfrP         2.796e-07  5.711e-07  4.900e-01    0.627
mfrQ         2.062e-08  5.527e-07  3.700e-02    0.970
mfrR        -1.363e-07  6.089e-07 -2.240e-01    0.824
typeH        2.121e-07  3.895e-07  5.450e-01    0.589
calories    -2.227e-01  9.328e-09 -2.388e+07   <2e-16 ***
protein      3.273e+00  6.621e-08  4.944e+07   <2e-16 ***
fat         -1.691e+00  1.025e-07 -1.651e+07   <2e-16 ***
sodium      -5.449e-02  7.919e-10 -6.882e+07   <2e-16 ***
fiber        3.443e+00  9.685e-08  3.555e+07   <2e-16 ***
carbo        1.092e+00  5.174e-08  2.111e+07   <2e-16 ***
sugars      -7.249e-01  4.684e-08 -1.547e+07   <2e-16 ***
potass      -3.399e-02  2.385e-09 -1.425e+07   <2e-16 ***
vitamins    -5.121e-02  2.393e-09 -2.140e+07   <2e-16 ***
shelf       -1.055e-07  6.707e-08 -1.573e+00    0.124
weight      -4.906e-07  8.524e-07 -5.760e-01    0.568
cups         1.694e-07  2.377e-07  7.130e-01    0.481
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.944e-07 on 37 degrees of freedom
Multiple R-squared:      1,     Adjusted R-squared:      1
F-statistic: 5.776e+15 on 19 and 37 DF,  p-value: < 2.2e-16

> |
```

Now we are fitting model to the test set and checking accuracy.

```
#here we are fitting training model onto the test set
pred=predict(model,newdata=test)
#Now using mean squared error, we are Calculating the Accuracy
MSE=mean((test$rating-pred)^2)
#Printing MSE
print(MSE)
```

```
> #here we are fitting training model onto the test set
> pred=predict(model,newdata=test)
> #Now using mean squared error, we are Calculating the Accuracy
> MSE=mean((test$rating-pred)^2)
> #Printing MSE
> print(MSE)
[1] 2.730089e-13
> |
```

b) With the data in (a) performing forwards subset selection

```
> library(olsrr)
> forward <- lm(rating~sodium+protein+fat,data=train)
> forward_fit <- ols_step_forward_p(forward, penter = 0.2)
> forward_fit

                        Selection Summary
--------------------------------------------------------------------------
        Variable                    Adj.
Step    Entered     R-Square    R-Square     C(p)        AIC         RMSE
--------------------------------------------------------------------------
   1    protein       0.2343      0.2203    63.8714    444.2205    11.5071
   2    fat           0.5335      0.5162    20.2017    417.9737     9.0644
   3    sodium        0.6527      0.6331     4.0000    403.1459     7.8939
--------------------------------------------------------------------------
> |
```

c) With the data in (a) performing exhaustive subset selection

```
> exhaustive <-lm(rating~sodium+protein+fat,data=train)
> exhaustive_fit<- ols_step_best_subset(exhaustive, details = TRUE)
> exhaustive_fit
     Best Subsets Regression
---------------------------------
Model Index    Predictors
---------------------------------
     1         protein
     2         protein fat
     3         sodium protein fat
---------------------------------

                                                    Subsets Regression Summary
-----------------------------------------------------------------------------------------------------------------------------------------------------
                        Adj.        Pred
Model    R-Square     R-Square     R-Square      C(p)        AIC         SBIC        SBC         MSEP         FPE         HSP         APC
-----------------------------------------------------------------------------------------------------------------------------------------------------
  1        0.2343       0.2203       0.1714    63.8714    444.2205    279.8875    450.3497    7547.6716    137.0583    2.4521    0.8214
  2        0.5335       0.5162       0.4772    20.2017    417.9737    254.9384    426.1459    4685.0364     86.4880    1.5503    0.5184
  3        0.6527       0.6331       0.5996     4.0000    403.1459    241.9793    413.3612    3554.4380     66.6864    1.1983    0.3997
-----------------------------------------------------------------------------------------------------------------------------------------------------
AIC: Akaike Information Criteria
 SBIC: Sawa's Bayesian Information Criteria
 SBC: Schwarz Bayesian Criteria
 MSEP: Estimated error of prediction, assuming multivariate normality
 FPE: Final Prediction Error
 HSP: Hocking's Sp
 APC: Amemiya Prediction Criteria
```

d) The exhaustive subset selection is better than linear regression model. This is known by comparing the MSE values of models respectively. The linear regression model yields an MSE value of 2.730089e-13 for, whereas exhaustive model yields MSE value of protein 7547.671. Although the MSE value states that exhaustive is the best model, it may not be the correct/right model for all scenarios.

2) Reading the RData files

```
#Loading the Train file
#(here var 1 and var2 shall be used as variables to initially input files, and then put into another variable considering the 4's and 7's, respectively)

var1 <- get(load("zip.train.RData"))

#here we are considering only 4's and 7's
var2 <- which(var1[, 1] == 4 | var1[, 1] == 7)
x.train <- var1[var2, -1]
y.train <- var1[var2, 1] == 7

#Loading the Test file similarly as above
var1 <- get(load("zip.test.RData"))
var2 <- which(var1[, 1] == 4 | var1[, 1] == 7)
x.test <- var1[var2, -1]
y.test <- var1[var2, 1] == 7
```

Dimensions of the training and testing variables are:

```
> dim(x.train)
[1] 1297  256
> dim(y.train)
NULL
>
> dim(x.test)
[1] 347 256
> dim(y.test)
NULL
```

Now we shall apply a linear regression model for the training data:

```
> model.train <- lm(y.train ~ x.train)
> summary(model.train)

Call:
lm(formula = y.train ~ x.train)

Residuals:
     Min       1Q   Median       3Q      Max
-0.49961 -0.08647 -0.00440  0.08748  0.91455

Coefficients: (3 not defined because of singularities)
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  3.357e-01  8.162e+00   0.041 0.967197
x.train1     3.670e-02  9.406e-02   0.390 0.696494
x.train2     3.699e-02  4.638e-02   0.798 0.425342
x.train3    -1.029e-02  3.372e-02  -0.305 0.760392
x.train4     1.040e-02  2.358e-02   0.441 0.659373
x.train5     5.608e-02  1.962e-02   2.859 0.004340 **
x.train6    -2.408e-02  1.824e-02  -1.320 0.187129
x.train7     3.724e-02  1.832e-02   2.033 0.042343 *
x.train8     1.522e-02  1.958e-02   0.778 0.436981
x.train9     3.774e-02  2.103e-02   1.795 0.072970 .
x.train10    2.220e-02  1.948e-02   1.140 0.254722
x.train11   -8.571e-03  1.580e-02  -0.543 0.587511
x.train12    8.793e-03  1.455e-02   0.604 0.545820
x.train13    1.642e-03  1.619e-02   0.101 0.919272
x.train14   -3.211e-03  2.114e-02  -0.152 0.879268
x.train15    2.211e-02  3.209e-02   0.689 0.491057
x.train16    5.467e-02  6.866e-02   0.796 0.426045
x.train17    2.709e-02  6.378e-02   0.425 0.671044
x.train18   -5.426e-02  3.885e-02  -1.397 0.162792
x.train19    3.484e-02  2.785e-02   1.251 0.211260
x.train20   -2.648e-02  2.100e-02  -1.261 0.207578
x.train21    1.440e-02  1.840e-02   0.782 0.434113
x.train22    2.602e-02  1.866e-02   1.394 0.163485
x.train23    2.265e-02  2.041e-02   1.110 0.267269
x.train24    4.372e-02  2.277e-02   1.920 0.055154 .
x.train25    5.119e-02  2.514e-02   2.036 0.041987 *
x.train26    3.299e-02  2.349e-02   1.405 0.160464
x.train27    3.804e-02  1.876e-02   2.028 0.042786 *
x.train28   -3.346e-03  1.836e-02  -0.182 0.855445
x.train29    1.153e-02  1.903e-02   0.606 0.544709
x.train30   -2.805e-03  2.292e-02  -0.122 0.902609
x.train31   -1.677e-02  3.187e-02  -0.526 0.598752
x.train32   -7.620e-02  5.143e-02  -1.482 0.138705
x.train33    1.421e-01  5.037e-02   2.821 0.004875 **
x.train34   -8.079e-02  3.430e-02  -2.356 0.018681 *
x.train35   -1.012e-04  2.302e-02  -0.004 0.996496
x.train36    1.528e-02  1.888e-02   0.810 0.418365
x.train37    9.154e-03  1.697e-02   0.539 0.589722
x.train38    8.717e-05  1.668e-02   0.005 0.995830
x.train39   -2.682e-03  1.914e-02  -0.140 0.888572
x.train40    1.518e-02  2.381e-02   0.638 0.523855
x.train41    6.505e-02  2.603e-02   2.499 0.012624 *
x.train42   -2.034e-02  2.523e-02  -0.806 0.420253
x.train43    7.778e-03  2.202e-02   0.353 0.723999
x.train44   -6.285e-03  2.167e-02  -0.290 0.771883
x.train153   5.009e-03  1.994e-02   0.251 0.801735
x.train154  -1.802e-02  2.126e-02  -0.847 0.397008
x.train155   3.425e-03  2.067e-02   0.166 0.868431
x.train156  -3.320e-02  2.175e-02  -1.527 0.127160
x.train157   2.889e-02  2.591e-02   1.115 0.264958
x.train158  -3.896e-02  3.415e-02  -1.141 0.254189
x.train159   7.376e-02  4.917e-02   1.500 0.133912
x.train160  -6.390e-02  6.717e-02  -0.951 0.341676
x.train161  -2.584e-01  1.372e-01  -1.884 0.059868 .
x.train162   8.573e-02  6.350e-02   1.350 0.177250
x.train163   2.125e-03  4.614e-02   0.046 0.963270
x.train164  -2.385e-02  3.699e-02  -0.645 0.519265
x.train165   1.277e-02  3.782e-02   0.338 0.735743
x.train166  -3.088e-02  3.646e-02  -0.847 0.397230
x.train167   9.259e-04  3.131e-02   0.030 0.976416
x.train168   3.269e-03  2.508e-02   0.130 0.896331
x.train169  -8.577e-02  2.343e-02  -3.661 0.000264 ***
x.train170   3.290e-02  2.313e-02   1.422 0.155193
x.train171  -1.434e-03  2.268e-02  -0.063 0.949597
x.train172  -1.462e-02  2.634e-02  -0.555 0.578951
x.train173  -2.415e-02  3.708e-02  -0.651 0.514986
x.train174   1.298e-01  5.669e-02   2.289 0.022266 *
x.train175  -1.644e-01  6.754e-02  -2.435 0.015077 *
x.train176   7.273e-02  9.305e-02   0.782 0.434615
x.train177   6.156e-01  3.240e-01   1.900 0.057721 .
x.train178   3.787e-02  1.154e-01   0.328 0.742890
x.train179  -1.206e-01  7.987e-02  -1.510 0.131437
x.train180  -2.419e-02  5.140e-02  -0.471 0.638019
x.train181   4.034e-02  4.482e-02   0.900 0.368291
x.train182  -2.790e-02  4.320e-02  -0.646 0.518558
x.train183  -5.892e-03  3.316e-02  -0.178 0.859025
x.train184  -1.093e-02  2.773e-02  -0.394 0.693657
x.train185   9.522e-03  2.857e-02   0.333 0.738953
x.train186  -8.143e-02  2.524e-02  -3.226 0.001293 **
x.train187  -5.001e-02  2.703e-02  -1.850 0.064623 .
x.train188   2.899e-02  3.851e-02   0.753 0.451772
x.train189   5.386e-02  5.934e-02   0.908 0.364242
x.train190  -1.576e-01  8.862e-02  -1.778 0.075649 .
x.train191   1.683e-01  1.161e-01   1.450 0.147262
x.train192  -3.810e-02  1.644e-01  -0.232 0.816785
x.train193  -1.023e+00  1.228e+00  -0.833 0.405200
x.train194  -8.714e-02  3.764e-01  -0.231 0.816977
x.train195   1.913e-01  1.463e-01   1.308 0.191310
x.train196  -5.280e-02  9.816e-02  -0.538 0.590728
x.train197  -2.308e-02  6.403e-02  -0.360 0.718554
x.train198  -7.051e-03  4.769e-02  -0.148 0.882484
x.train199  -1.814e-02  3.266e-02  -0.555 0.578755
 [ reached getOption("max.print") -- omitted 57 rows ]
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’

Residual standard error: 0.1598 on 1043 degrees of freedom
Multiple R-squared:  0.9179,    Adjusted R-squared:  0.898
F-statistic: 46.08 on 253 and 1043 DF,  p-value: < 2.2e-16
```

Now we shall perform Knn Classification with k values:

```
#knn with k values
# Classification by k-nearest neighbors
library(class)
z <- c(1, 3, 5, 7, 15)
z_error <- rep(NA, length(k))
for (i in 1:length(k)) {
  y <- knn(x.train, x.test, y.train, k[i])
  z_error[i] <- mean(y != y.test)
}
z_error
```

```
> #knn with k values
> # Classification by k-nearest neighbors
> library(class)
> z <- c(1, 3, 5, 7, 15)
> z_error <- rep(NA, length(k))
> for (i in 1:length(k)) {
+   y <- knn(x.train, x.test, y.train, k[i])
+   z_error[i] <- mean(y != y.test)
+ }
> z_error
[1] 0.02305476 0.02017291 0.02017291 0.02593660 0.02593660
>
```

classification using linear regression:

```
> # Classification by linear regression
> l_model <- lm(y.train ~ x.train)
> y <- (cbind(1, x.test) %*% l_model$coef) >= 0.5
> l_model_error <- mean(y != y.test)
> l_model_error
[1] NA
```

Comparing which method performed better:

```
> # Compare results of linear regression model and knn w.r.t k values
> compared_error <- matrix(c(L.error, k.error), ncol = 1)
> colnames(compared_error) <- c("errors")
> rownames(compared_error) <- c("Linear Regression model", paste("k-NN considering k =", k))
> compared_error
                            errors
Linear Regression model         NA
k-NN considering k = 1  0.02305476
k-NN considering k = 3  0.02017291
k-NN considering k = 5  0.02017291
k-NN considering k = 7  0.02593660
k-NN considering k = 15 0.02593660
>
```

3) a) normalizing and splitting the data set

```
> # normalizing and creating equal partitions of training and test data set
> College[, -1] <- apply(College[, -1], 2, scale)
> #calculating training data set size
> train.size <-  dim(College)[1] / 2
> train <-  sample(1:dim(College)[1], train.size)
> #the remaning data set other than the one included in training(50% each)
> test <-  -train
> College_train_dataset <-  College[train, ]
> College_test_dataset <-  College[test, ]
>
```

Fitting a linear model using least squares on the training set

```
> linear_model <-  lm(Apps ~ . , data = College_train_dataset)
> model_predictor <-  predict(linear_model, College_test_dataset)
> mean((College_test_dataset[, "Apps"] - lm.pred)^2)
[1] 0.08587343
>
```

The test error obtained is 0.08587343.

b) Fitting a ridge regression model on the training set, with λ chosen by cross-validation.

```
> library(glmnet)
> #creating training and test matrices and generating a grid in order to find lamda
> train_matrix <-  model.matrix(Apps ~ . -1 , data = College_train_dataset)
> test_matrix <-  model.matrix(Apps ~ . -1, data = College_test_dataset)
> grid <-  10 ^ seq(4, -2, length = 100)
> m <-  cv.glmnet(train_matrix, College_train_dataset[, "Apps"],
+                         alpha = 0, lambda = grid, thresh = 1e-12)
> l <-  m$lambda.min
> l
[1] 0.01321941
```

Therefore the best value of lambda is 0.01

```
> ridge_predictors <-  predict(m, newx = test_matrix, s = l)
> mean((College_test_dataset[, "Apps"] - ridge_predictors)^2)
[1] 0.09364753
```

The test error obtained is 0.09364753

c)

```
> las <-  cv.glmnet(train_matrix, College_train_dataset[, "Apps"],
+                          alpha = 1, lambda = grid, thresh = 1e-12)
> l <-  las$lambda.min
> l
[1] 0.01
>
> las_predictor <-  predict(las, newx = test_matrix, s = l)
> mean((College_test_dataset[, "Apps"] - las_predictor)^2)
[1] 0.08993697
>
```

The test error obtained using lasso model's lambda (0 .01) is 0.08993697.

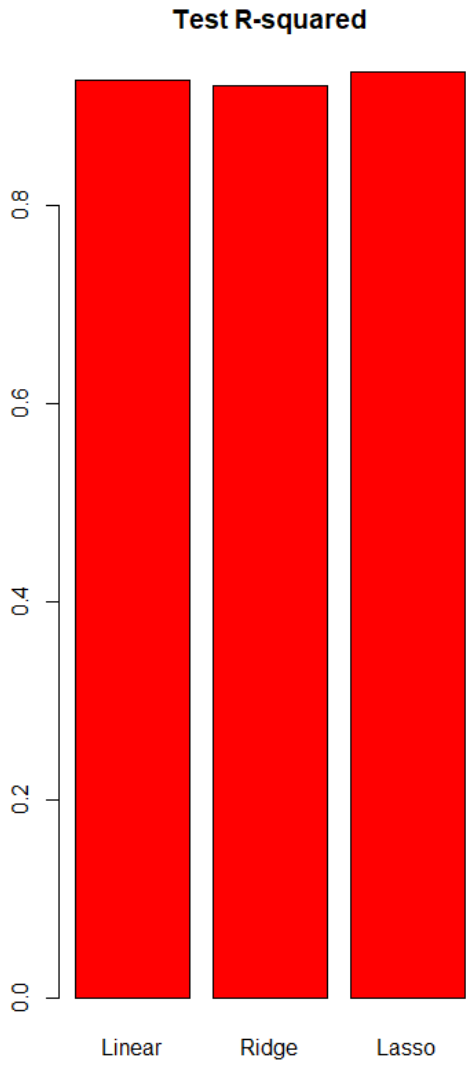
The non-zero coefficient estimates are :

```
> las <-  glmnet(model.matrix(Apps ~ . -1, data = College),
+                College[, "Apps"], alpha = 1)
> predict(las, s = l, type = "coefficients")
19 x 1 sparse Matrix of class "dgCMatrix"
                      s1
(Intercept) -2.483323e-02
PrivateNo    9.101612e-02
PrivateYes  -1.362175e-13
Accept       8.827830e-01
Enroll       .
Top10perc    1.285778e-01
Top25perc    .
F.Undergrad  .
P.Undergrad  .
Outstate    -3.693941e-02
Room.Board   2.682937e-02
Books        .
Personal     .
PhD         -1.307949e-02
Terminal    -1.016626e-02
S.F.Ratio    .
perc.alumni -1.794075e-03
Expend       8.228831e-02
Grad.Rate    1.271356e-02
>
```

d) We can make R squared for all models and plot to compare the shared trends between the colleges. This is done as follows:

```
> test_average <-  mean(College_test_dataset[, "Apps"])
>
> l_test_rsqr <-  1 - mean((College_test_dataset[, "Apps"] - model_predictor)^2) /
+   mean((College_test_dataset[, "Apps"] - test_average)^2)
>
> ridge_test_rsqr <-  1 - mean((College_test_dataset[, "Apps"] - ridge_predictors)^2)/
+   mean((College_test_dataset[, "Apps"] - test_average)^2)
>
> lasso_test_rsqr <-  1 - mean((College_test_dataset[, "Apps"] - las_predictor)^2) /
+   mean((College_test_dataset[, "Apps"] - test_average)^2)
>
> barplot(c(l_test_rsqr, ridge_test_rsqr, lasso_test_rsqr),
+         col = "red", names.arg = c("OLS", "Ridge", "Lasso"),
+         main = "Test R-squared")
>
```

We get the following bar plot from comparing these models:



From this we can observe they all shared a similar R2(r squared) value which is approximately 0.9.