In [1]:
```
install.packages('leaps')
```

```
package 'leaps' successfully unpacked and MD5 sums checked

The downloaded binary packages are in
        C:\Users\ashka\AppData\Local\Temp\RtmpkFu7iv\downloaded_packages
```

In [2]:
```
library(caret)
library(leaps)
library(MASS)
library(glmnet)
```

```
Loading required package: lattice
Loading required package: ggplot2
Registered S3 methods overwritten by 'ggplot2':
  method         from
  [.quosures     rlang
  c.quosures     rlang
  print.quosures rlang
Loading required package: Matrix
Loading required package: foreach
Loaded glmnet 2.0-16
```

In [3]:
```
raw_data<-read.table(file = './uscrime.txt',header = TRUE)
```

In [4]:
```
#It is necessary to scale the data as mentioned also in the lecture to optimize the coeffiect range

remove_points <- c("So","Crime")
data_remove_points<-raw_data[, !(names(raw_data) %in% remove_points)]
scaled_data_removed<-scale(data_remove_points)
scaled_data<-cbind(scaled_data_removed,raw_data[,remove_points])
```

In [5]:
```
# The step wise regression is implemented here:
# The range of features is tried to find the optimum model


control_variable <- trainControl(method = "repeatedcv", number = 5, repeats = 5)

step_wise_model <- train(Crime ~., data = scaled_data, method = "leapSeq",tuneGrid = data.frame(nvmax = 1:15
), trControl = control_variable)
```

In [6]:
```
step_wise_model$results
```

nvmax	RMSE	Rsquared	MAE	RMSESD	RsquaredSD	MAESD
1	291.5779	0.5227764	230.8915	80.52834	0.2889516	74.80276
2	308.1104	0.4503600	238.4573	93.82773	0.3121311	71.89600
3	265.5925	0.5740250	209.2237	77.24358	0.2716775	68.41373
4	285.1170	0.5081315	221.7260	61.58699	0.2441184	48.16886
5	276.3819	0.5224353	222.6152	77.18560	0.2469174	62.99584
6	246.7716	0.5991393	193.8343	60.97895	0.2226581	50.10355
7	272.6054	0.5435213	220.7917	59.79278	0.2280076	49.79663
8	280.2471	0.5340670	229.3941	61.54290	0.2449292	51.32562
9	301.7002	0.4500372	240.6683	65.08583	0.2498655	57.50237
10	286.1816	0.5202469	236.0677	54.98126	0.2345198	48.36711
11	289.8436	0.5142951	234.0024	52.93549	0.2307848	45.59564
12	290.2612	0.5073331	234.4481	54.11271	0.2037706	44.13606
13	281.5523	0.5313749	226.9553	49.46382	0.2164079	41.29760
14	279.3022	0.5386282	225.1971	47.90591	0.2026085	36.52641
15	291.6086	0.5022123	235.3427	47.57263	0.2122186	37.83192

In [7]:
```
summary(step_wise_model)
```

```
Subset selection object
15 Variables  (and intercept)
       Forced in Forced out
M          FALSE      FALSE
Ed         FALSE      FALSE
Po1        FALSE      FALSE
Po2        FALSE      FALSE
LF         FALSE      FALSE
M.F        FALSE      FALSE
Pop        FALSE      FALSE
NW         FALSE      FALSE
U1         FALSE      FALSE
U2         FALSE      FALSE
Wealth     FALSE      FALSE
Ineq       FALSE      FALSE
Prob       FALSE      FALSE
Time       FALSE      FALSE
So         FALSE      FALSE
1 subsets of each size up to 6
Selection Algorithm: 'sequential replacement'
         M   Ed  Po1 Po2 LF  M.F Pop NW  U1  U2  Wealth Ineq Prob Time So 
1  ( 1 ) " " " " "*" " " " " " " " " " " " " " " " "    " "  " "  " "  " "
2  ( 1 ) " " " " "*" " " " " " " " " " " " " " " " "    "*"  " "  " "  " "
3  ( 1 ) " " "*" "*" " " " " " " " " " " " " " " " "    "*"  " "  " "  " "
4  ( 1 ) "*" "*" "*" "*" " " " " " " " " " " " " " "    " "  " "  " "  " "
5  ( 1 ) "*" "*" "*" " " " " " " " " " " " " " " " "    "*"  "*"  " "  " "
6  ( 1 ) "*" "*" "*" " " " " " " " " " " " " "*" " "    "*"  "*"  " "  " "
```

In [8]:
```
#After investigating which features are selected by the step_wise
#It is possible to built a linear regression based on the selected factors
linear_regression_model<-lm(formula = Crime ~ M+So+Ed + Po1 , data = scaled_data)
```

In [9]:
```
linear_regression_model
```

```
Call:
lm(formula = Crime ~ M + So + Ed + Po1, data = scaled_data)

Coefficients:
(Intercept)            M           So           Ed          Po1
     855.90       132.78       144.49        92.05       314.42
```

Lasso Regression

In [10]:
```
# The following developed data set will be used for the Lasso Regression
head(scaled_data)
```

M	Ed	Po1	Po2	LF	M.F	Pop	NW	U1	U2	Wealth
0.9886930	-1.3085099	-0.9085105	-0.8666988	-1.2667456	-1.12060499	-0.09500679	1.943738564	0.69510600	0.8313680	-1.3616094
0.3521372	0.6580587	0.6056737	0.5280852	0.5396568	0.98341752	-0.62033844	0.008483424	0.02950365	0.2393332	0.3276683
0.2725678	-1.4872888	-1.3459415	-1.2958632	-0.6976051	-0.47582390	-0.48900552	1.146296747	-0.08143007	-0.1158877	-2.1492481
-0.2048491	1.3731746	2.1535064	2.1732150	0.3911854	0.37257228	3.16204944	-0.205464381	0.36230482	0.5945541	1.5298536
0.1929983	1.3731746	0.8075649	0.7426673	0.7376187	0.06714965	-0.48900552	-0.691709391	-0.24783066	-1.6551781	0.5453053
-1.3983912	0.3898903	1.1104017	1.2433590	-0.3511718	-0.64550313	-0.30513945	-0.555560788	-0.63609870	-0.5895155	1.6956723

In [11]:
```
# The Lasso model will be developed in the following:

lasso_model = cv.glmnet(x = as.matrix(scaled_data[,-16]),
                        y = as.matrix(scaled_data$Crime),
                        alpha = 1, nfolds = 5,
                        type.measure = "mse", family = "gaussian", standardize = F)
```

In [12]:
```
#The summary of the lasso model is presented in the following:

coef(lasso_model, s = lasso_model$lambda.min)
```

```
16 x 1 sparse Matrix of class "dgCMatrix"
                    1
(Intercept) 905.08511
M            87.38594
Ed          123.07830
Po1         308.82944
Po2           .
LF            .
M.F          52.10363
Pop           .
NW           11.13820
U1          -28.85703
U2           62.03082
Wealth        .
Ineq        187.18594
Prob        -77.09109
Time          .
So            .
```

In [13]:
```
# Regression model for the lasso will be presented in the following:
lasso_linear_model <- lm(Crime ~ M + Ed + Po1 + M.F + NW + U1 + U2 + Ineq + Prob, data = scaled_data)
```

In [14]:
```
# Testing the trained lasso model after developing a linear regression
summary(lasso_linear_model)
```

```
Call:
lm(formula = Crime ~ M + Ed + Po1 + M.F + NW + U1 + U2 + Ineq +
    Prob, data = scaled_data)

Residuals:
   Min     1Q Median     3Q    Max
-439.2 -102.2   -6.3  124.1  476.6

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   905.09      28.87  31.352  < 2e-16 ***
M             111.23      46.83   2.375 0.022820 *
Ed            203.63      60.12   3.387 0.001687 **
Po1           297.89      52.08   5.719 1.51e-06 ***
M.F            68.74      41.63   1.651 0.107134
NW             16.55      53.15   0.311 0.757222
U1           -109.46      60.94  -1.796 0.080609 .
U2            156.94      62.09   2.528 0.015889 *
Ineq          236.70      61.95   3.821 0.000492 ***
Prob          -89.99      36.28  -2.481 0.017791 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 197.9 on 37 degrees of freedom
Multiple R-squared:  0.7894,    Adjusted R-squared:  0.7381
F-statistic: 15.41 on 9 and 37 DF,  p-value: 4.881e-10
```

Elastic Net

In [17]:
```
#It is important to find the alpha value to be able to develop the Elastic net

mse_list <- numeric()

search_alpha <- function(num_value, scaled_data){
    alpha <- num_value
    elastic_net <- cv.glmnet(x=as.matrix(scaled_data[,-16]),
                            y=as.matrix(scaled_data[,16]),
                            alpha = alpha,
                            nfolds=5,
                            type.measure="mse",
                            family="gaussian",
                            standardize=FALSE)

        mse_list <<- cbind(mse_list, c(alpha, min(elastic_net$cvm),elastic_net$lambda.min))
}
```

In [18]:
```
for (i in seq(.01,1,by = .01)){search_alpha(i,scaled_data)}
```

In [19]:
```
minIndex <- which.min(mse_list[2,])
```

In [20]:
```
# The alpha value will be used as part of the model for the development.

mse_list[1, minIndex]
```

0.8

In [21]:
```
elastic_net_final <- cv.glmnet(x=as.matrix(scaled_data[,-16]),
                            y=as.matrix(scaled_data[,16]),
                            alpha = 0.26,
                            nfolds=5,
                            type.measure="mse",
                            family="gaussian",
                            standardize=FALSE)
```

In [22]:
```
coef(elastic_net_final, s = elastic_net_final$lambda.min)

# The linear regression model will be develop in the following section
elastic_linear_regression <- lm(Crime ~ ., data = scaled_data[, -4])
```

```
16 x 1 sparse Matrix of class "dgCMatrix"
                     1
(Intercept) 900.219667
M            93.139696
Ed          138.787373
Po1         199.220825
Po2          78.485074
LF             .
M.F          66.718527
Pop          -2.658487
NW           27.340856
U1          -64.139269
U2           99.200515
Wealth       39.484214
Ineq        200.575065
Prob        -85.446760
Time           .
So           14.292230
```

In [23]:
```
# The summary of the elastic linear regression will be presented in the following:

summary(elastic_linear_regression)
```

```
Call:
lm(formula = Crime ~ ., data = scaled_data[, -4])

Residuals:
    Min      1Q  Median      3Q     Max 
-442.55 -116.46    8.86  118.26  473.49 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  903.155     58.889  15.336 2.66e-16 ***
M            112.934     52.244   2.162 0.038232 *  
Ed           198.350     68.044   2.915 0.006445 ** 
Po1          286.864     71.091   4.035 0.000317 ***
LF           -11.321     56.896  -0.199 0.843538    
M.F           53.684     59.798   0.898 0.376026    
Pop          -29.833     48.950  -0.609 0.546523    
NW            25.149     63.619   0.395 0.695239    
U1           -97.649     75.332  -1.296 0.204164    
U2           143.034     69.378   2.062 0.047441 *  
Wealth        87.540     99.662   0.878 0.386292    
Ineq         290.076     90.023   3.222 0.002921 ** 
Prob         -97.432     49.655  -1.962 0.058484 .  
Time          -7.991     47.425  -0.168 0.867251    
So             5.669    148.100   0.038 0.969705    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 208.6 on 32 degrees of freedom
Multiple R-squared:  0.7976,	Adjusted R-squared:  0.709 
F-statistic: 9.006 on 14 and 32 DF,  p-value: 1.673e-07
```

In [ ]: