Lasso Regression

Implementing Lasso Regression and Studying the Results

Introduction

When the value of coefficients corresponding to the variables in a polynomial equation becomes too high, the dependent variables y become too sensitive to those variables. Even a small change in the value of those variables which have high coefficients value will make a big change in dependent variable y.

That is why lasso regression comes into picture, it penalizes those coefficients that have higher value by including a adding a new variables called lambda to the old ordinary least square equation. This makes the regression model more general and predictable.

I am going to take example of diabetes dataset in order to follow this exercises and this dataset is inbuilt in R environment in lars package. The dataset has three elements to it and they are called x, y and x2. The each one of these elements have dimensions of itself.

X has 442 rows and 10 columns

X2 has 442 rows and 64 columns

Y has length of 442

X has attributes like age, sex, bmi, glucose level etc. that are used to predict the diabetes level that is stored in y.

X2 also have many different attributes related to human health and they are used to predict the progression of diseases that is y.

We are going to see how lasso regression can help us utilize only those columns out of x and x2 that are helping us in prediction value of y. It is also a form of regression but it understands which columns hold more value and those who does not have much value will be removed.

1.

First check the structure of the diabetes data frame. Then assign all three elements of data to variables x, x2 and y for sake of simplicity. Then we check the dimension of the all three datasets.

```
#1
df <- diabetes #loading datset
str(df) # checking stuctrue
x <- df$x  # assigning x element to variables x
y <- df$y  #assigning y element to variables y
x2 <- df$x2 #assigning x2 element to variables x2
dim(x)  # cheking dimension of variable x
dim(x2) # cheking dimension of variable x2
length(y) # cheking length of variable y
```

```
> str(df)
'data.frame': 442 obs. of  3 variables:
 $ x : AsIs [1:442, 1:10] 0.038075.... -0.00188.... 0.085298.... -0.08906.... 0.005383.... ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : NULL
  .. ..$ : chr  "age" "sex" "bmi" "map" ...
 $ y : num  151 75 141 206 135 97 138 63 110 310 ...
 $ x2: AsIs [1:442, 1:64] 0.038075.... -0.00188.... 0.085298.... -0.08906.... 0.005383.... ...
  ..- attr(*, ".Names")= chr  "age" "age" "age" "age" "age" ...
  ..- attr(*, "dimnames")=List of 2
  .. ..$ : chr  "1" "2" "3" "4" ...
  .. ..$ : chr  "age" "sex" "bmi" "map" ...
> dim(x)
[1] 442  10
> dim(x2)
[1] 442  64
> length(y)
[1] 442
```

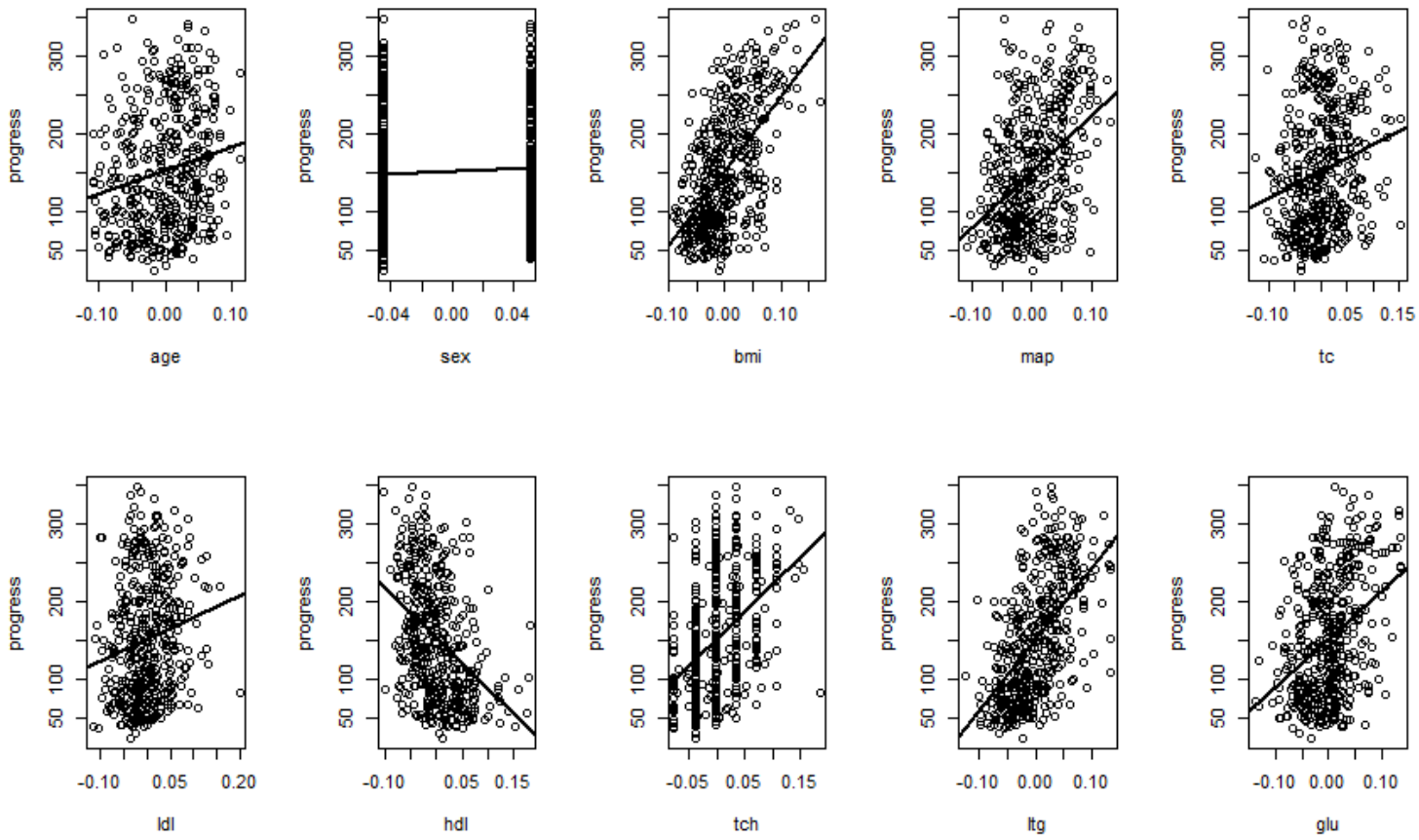
The y varies has numeric datatype. Whereas x and x2 have special data type called AsIs. That is why I decided to separate them out of data frame and make individualized.

2.

Since there are 10 columns in x, meaning 10 different attributes. To understand the relationship between those attributes and the values in y, I have drawn 10 plots using a for-loop that iterates ten times. And, I have also added a straight regression line defining the relationship between attributes and y.

```
#2
par(mfrow=c(2,5)) # changing screen to show 10 graphs at a time.

# creating a for loop to iteriate over all the columns of x with respect to y.
for (i in 1:ncol(x)) {
  plot(x[,i],y,xlab = colnames(x)[i],ylab="progress")
  abline(lm(y~x[,i]))
}
dev.off()   # clsoing the screen to show single plot.
```



I see that almost all of the factors such as age, body mass index, blood pressure and all other six blood serum except hdl are making a direct impact in increasing the chances of diseases. Increase in hdl serum will decrease the chances of diseases.

3.

We saw the liner relation between the predictors in x on dependent variables y, but we need to build a liner model to see which variables are really important and have a significant impact on predicting y.

```
#3

model <- lm(y~x)  #creating a liner model by regressing y on predictors on y
summary(model) # checking the summary of model
```

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-155.829  -38.534   -0.227   37.806  151.355

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  152.133      2.576  59.061  < 2e-16 ***
xage         -10.012     59.749  -0.168 0.867000
xsex        -239.819     61.222  -3.917 0.000104 ***
xbmi         519.840     66.534   7.813 4.30e-14 ***
xmap         324.390     65.422   4.958 1.02e-06 ***
xtc         -792.184    416.684  -1.901 0.057947 .
xldl         476.746    339.035   1.406 0.160389
xhdl         101.045    212.533   0.475 0.634721
xtch         177.064    161.476   1.097 0.273456
xltg         751.279    171.902   4.370 1.56e-05 ***
xglu          67.625     65.984   1.025 0.305998
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 54.15 on 431 degrees of freedom
Multiple R-squared:  0.5177, Adjusted R-squared:  0.5066
F-statistic: 46.27 on 10 and 431 DF,  p-value: < 2.2e-16
```

The three starts signifies that those variables have high significance level in predicting the value of y even if there are other variables that are also having a direct relation with y. In other words, the model knows which variables have more evidence and which variables have less evidence that they are making an impact on value of y.

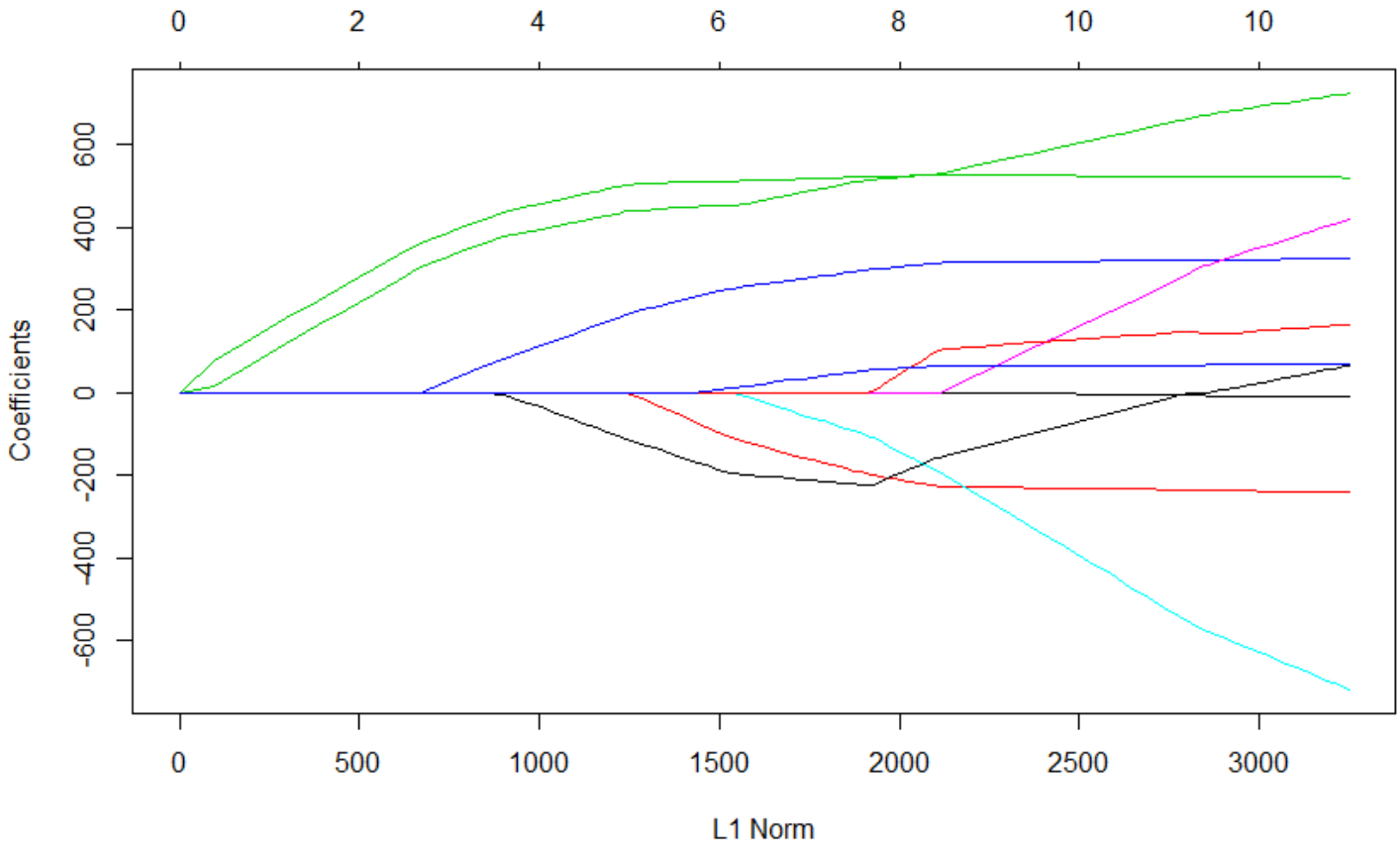
Gender, body mass index, blood pressure and xltg serum are higly significant variables in predicting chances of having a diseases.

4.

But if we have a lot of attributes, then choosing one over the other becomes very hard in liner regression. That if why we use lasso. Below I have used glmnet function keeping alpha equal to 1 for lasso (0 for ridge). And plotting the result.

```
#lasso

lasso <- glmnet(x,y,family = "gaussian",alpha = 1) # running a lasso regression
plot(lasso) # plotting the result of lasso regression
```

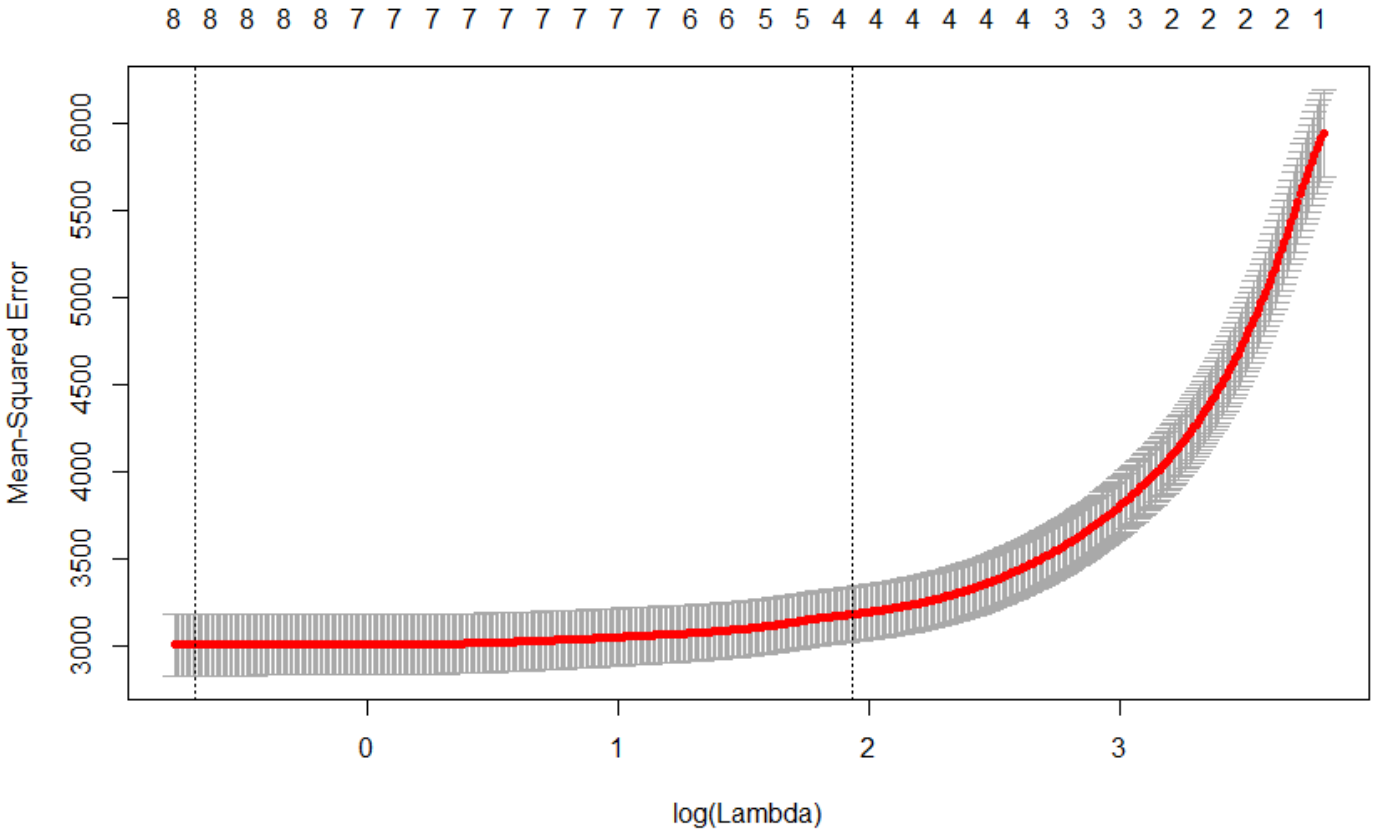


L1 norm is the absolute difference between the predicted value and the actual value. In other words, it is the cost function with absolute value. We can see in the graph that top line represents the coefficients used in the equation at the moment, as we move from left to right it keeps increasing. In other words, initially we have 0 variables and that is why 0 coefficients are included with zero cost function. But when we have 2 variables the cost function rises to be around 500 and the value of coefficients has also increases a little bit. If we include all the 10 variables then the cost function will be highest and model will be equivalent to OLS model.

5.

But what value of lambda should we choose so that the tradeoff between the number of coefficients used and the l1 norm is perfect. For that, we run cross validation which runs model many different times using different set of training value and present the result in the form of a graph that looks like this. Keeping nlambda equal to 1000 means we want to run cross validation 1000 times.

```
#5
cv <- cv.glmnet(x,y,alpha=1,nlambda=1000) #using cross validation
plot(cv) # ploting result of cross validation
cv$lambda.min # checking the minimum value possible of lambda
```



Here we can see that as the value of log lambda is increasing the mean-square error remains constant for a while and then increased instantly to a very high value. Meaning, if increase the value of lambda such that log of lambda is somewhere near 2, then it will be using 4 most important variables and at the same time the cost function will also be minimum.

Also, we have found out the minimum value of lambda from cross validation. Which is shown below.

```
> cv$lambda.min # checking the minimum value possible of lambda
[1] 0.5021116
```

At this value of lambda, the log lambda will be less than 0 and 8 variables will be used.

6.

To see all which 8 variables are present at minimum value of lambda, we can look at the beta matrix.

```
#6
fit <- glmnet(x,y,family="gaussian",lambda = cv$lambda.min) #running lasso at minimum value of lambda
fit$beta # extracting beta matrix
```

```
> fit$beta
10 x 1 sparse Matrix of class "dgCMatrix"
            s0
age    .
sex -216.32420
bmi  524.99789
map  308.27384
tc  -160.63313
ldl    .
hdl -180.75656
tch   65.74169
ltg  524.23124
glu   61.10076
>
```

It looks like two variables has been removed from the equation at minimum value of lambda. They are age and ldl. Meaning, these two variables had least amount of significance.

7.

But remove just a few variables will not be the solution of high dimensionality, we need to reduce the beta matrix further more.

```
#7
fit <- glmnet(x,y,family="gaussian",lambda = cv$lambda.1se) # running lasso at 1 standard deviation value of lambda
fit$beta # extarcting beta matrix at that value.
```

By choosing the value of lambda that is one standard deviation away from the minimum value of lambda will reduce the number of variables used in equation and at the same time will keep the cost function to the minimum.

The second vertical line in the cross validation graph above represents the value of lambda one standard deviation away from minimum value.

```
> fit$beta # extarcting beta matrix at that value.
10 x 1 sparse Matrix of class "dgCMatrix"
            s0
age    .
sex    .
bmi  500.1103
map  182.4389
tc     .
ldl    .
hdl -105.0984
tch    .
ltg  434.5747
glu    .
```

Now, only body mass index, blood pressure, ltg serum and hdl serum are left as the most significant variables in predicting the possibility of having a diseases (y).

8.

Till now, we had a very simple table expressing only a few variables to predict y. Now, let us take example of x2 and run all those steps again in determining how it chooses only most important factors out of 64 factors.

```
#8
model_2 <- lm(y~x2) # runing liner model on x2
summary(model_2) # summary of liner model
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)   152.133      2.532  60.086  < 2e-16 ***
x2age          50.721     65.513   0.774   0.4393
x2sex        -267.344     65.270  -4.096 5.15e-05 ***
x2bmi         460.721     84.601   5.446 9.32e-08 ***
x2map         342.933     72.447   4.734 3.13e-06 ***
x2tc        -3599.542  60575.187  -0.059   0.9526
x2ldl        3028.281  53238.699   0.057   0.9547
x2hdl        1103.047  22636.179   0.049   0.9612
x2tch          74.937    275.807   0.272   0.7860
x2ltg        1828.210  19914.504   0.092   0.9269
x2glu          62.754     70.398   0.891   0.3733
x2age^2        67.691     69.470   0.974   0.3305
x2bmi^2        45.849     83.288   0.550   0.5823
x2map^2        -8.460     71.652  -0.118   0.9061
x2tc^2       6668.449   7059.159   0.945   0.3454
x2ldl^2      3583.174   5326.148   0.673   0.5015
x2hdl^2      1731.821   1590.574   1.089   0.2769
x2tch^2       773.374    606.967   1.274   0.2034
x2ltg^2      1451.581   1730.103   0.839   0.4020
x2glu^2       114.149     94.122   1.213   0.2260
x2age:sex     148.678     73.407   2.025   0.0435 *
x2age:bmi     -18.052     79.620  -0.227   0.8208
x2age:map      18.534     76.303   0.243   0.8082
x2age:tc     -158.891    617.109  -0.257   0.7970
x2age:ldl     -67.285    494.527  -0.136   0.8918
x2age:hdl     209.245    280.614   0.746   0.4563
x2age:tch     184.960    210.330   0.879   0.3798
x2age:ltg     124.667    223.765   0.557   0.5778
x2age:glu      62.575     80.377   0.779   0.4367
x2sex:bmi      64.612     77.902   0.829   0.4074
x2sex:map      88.472     74.744   1.184   0.2373
x2sex:tc      433.598    590.709   0.734   0.4634
x2sex:ldl    -352.823    468.951  -0.752   0.4523
x2sex:hdl    -124.731    273.870  -0.455   0.6491
x2sex:tch    -131.223    199.714  -0.657   0.5115
x2sex:ltg    -118.995    226.493  -0.525   0.5996
x2sex:glu      45.758     73.650   0.621   0.5348
x2bmi:map     154.720     86.340   1.792   0.0739 .
x2bmi:tc     -302.045    667.930  -0.452   0.6514
x2bmi:ldl     241.540    561.026   0.431   0.6671
x2bmi:hdl     121.942    329.884   0.370   0.7118
x2bmi:tch     -33.445    230.836  -0.145   0.8849
x2bmi:ltg     114.673    255.987   0.448   0.6544
x2bmi:glu      23.377     91.037   0.257   0.7975
x2map:tc      478.303    682.264   0.701   0.4837
x2map:ldl    -326.740    574.317  -0.569   0.5697
x2map:hdl    -187.305    309.589  -0.605   0.5455
x2map:tch     -58.294    198.601  -0.294   0.7693
x2map:ltg    -154.795    271.966  -0.569   0.5696
x2map:glu    -133.476     91.314  -1.462   0.1447
x2tc:ldl    -9313.775  11771.220  -0.791   0.4293
x2tc:hdl    -3932.025   3816.572  -1.030   0.3036
x2tc:tch    -2205.910   1761.843  -1.252   0.2113
x2tc:ltg    -3801.442  13166.091  -0.289   0.7729
```

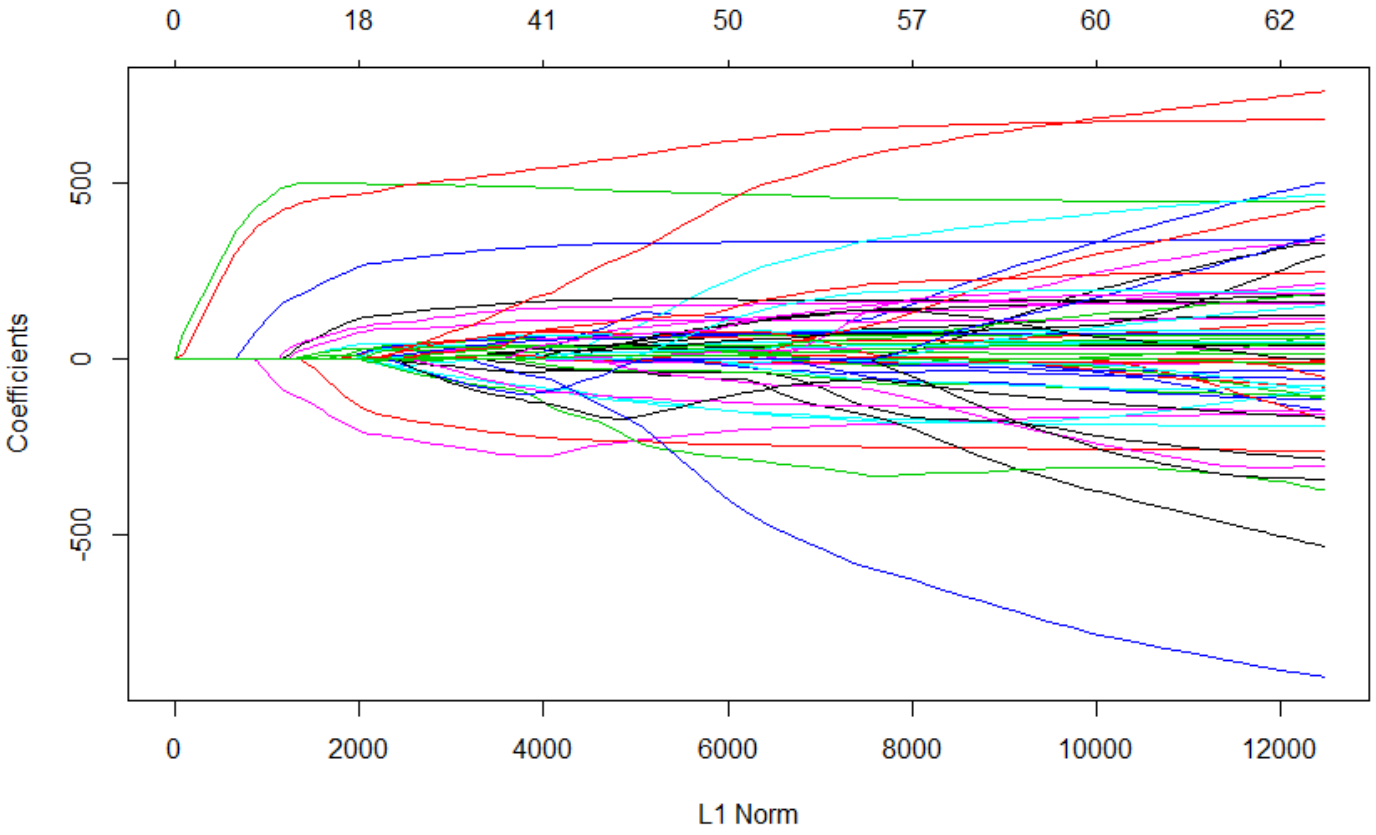
It looks like that of all the variables present in x2, only a few have high significance in determining the value of y. But, all other values are just making the model more complex and over fit to the data points. That is why we need to determine only significant variables using lasso.

9.

I am using lasso regression by regressing y on variables present in x2.

```
#9
lasso_2 <- glmnet(x2,y,family="gaussian",alpha=1) # running lasso on to regress y on x2
plot(lasso_2) # plotting lasso result.
```

I got this result.

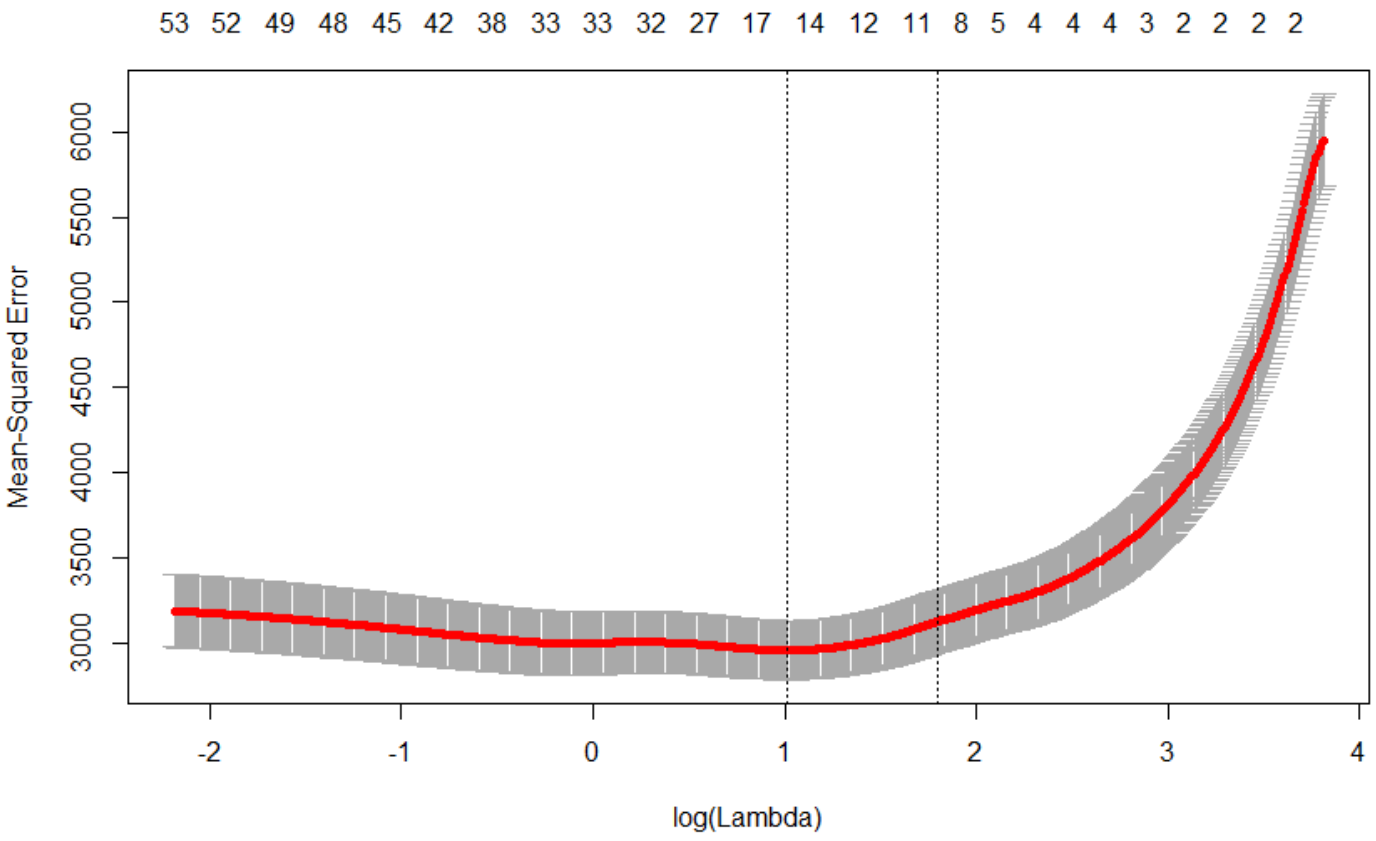


As we move from left to right, we see that initially it is just using a few variables represented by green, red and blue line. These variables have the most significance in determining value of y. However, as we move further ahead, the model starts adopting more variables as shown on the top line of graph. Also, the l1 norm starts increasing which means that absolute value of cost function is increasing.

10.

To determine appropriate value of lambda in this case, we run cross validation using the following code. nlambda again is 1000 means the cross validation will be done 1000 times.

```
#10
cv_2 <- cv.glmnet(x2,y,nlambda=1000)  #running cross validation
plot(cv_2) #ploting result of cross validation
cv_2$lambda.min #extarcting minimum value of lambda
```



Looks like the minimum value of log lambda possible is approximately 1 having around 15 variables, but log lambda that is one standard deviation away have around 10 variables and the mean-square error is till minimized.

The minimum value of lambda corresponding to log lambda value of 1 is shown below.

```
> cv_2$lambda.min
[1] 2.764019
```

11.

We need to check the beta matrix at minimum value of lambda and see how many variables has been removed.

```
#11
fit_2 <- glmnet(x2,y,family="gaussian",alpha=1, lambda = cv_2$lambda.min) #running lasso at minimum value of lambda
fit_2$beta # extracting beta matrix at minimum value of lambda
```

```
> fit_2$beta # extractin
64 x 1 sparse Matrix of
                s0
age        .
sex     -124.499681
bmi      501.109431
map      258.772983
tc         .
ldl        .
hdl     -195.756851
tch        .
ltg      469.113796
glu       22.681423
age^2     13.826199
bmi^2     41.316296
map^2      .
tc^2       .
ldl^2      .
hdl^2      .
tch^2      .
ltg^2      .
glu^2     73.994155
age:sex  112.557783
age:bmi    .
age:map   30.427604
age:tc     .
age:ldl    .
age:hdl    .
age:tch    .
age:ltg   11.134109
age:glu   10.202090
sex:bmi    .
sex:map    4.833878
sex:tc     .
sex:ldl    .
sex:hdl    .
sex:tch    .
sex:ltg    .
sex:glu    .
bmi:map   88.749467
bmi:tc     .
bmi:ldl    .
bmi:hdl    .
bmi:tch    .
bmi:ltg    .
bmi:glu    .
map:tc     .
map:ldl    .
map:hdl    .
```

It looks like that there are still many variables used in this model even though most of them has been penalized to zero at minimum value of lambda.

12.

Let's run lasso on x2 with lambda value that falls within one standard deviation with minimum value of lambda and see how many variables has been penalized.

```
#12
fit_2 <- glmnet(x2,y,family="gaussian",alpha=1, lambda = cv_2$lambda.1se) #running lasso at 1 sd value of lambda
fit_2$beta # extracting beta matrix at 1 sd value of lambda
```

```
                  s0
age         .
sex      -122.544294
bmi       501.199236
map       257.648157
tc          .
ldl         .
hdl      -194.485307
tch         .
ltg       468.898327
glu        21.949426
age^2      12.827328
bmi^2      40.892109
map^2       .
tc^2        .
ldl^2       .
hdl^2       .
tch^2       .
ltg^2       .
glu^2      73.318550
age:sex   111.780115
age:bmi     .
age:map    30.368699
age:tc      .
age:ldl     .
age:hdl     .
age:tch     .
age:ltg    10.725072
age:glu    10.426084
sex:bmi     .
sex:map     4.054816
sex:tc      .
sex:ldl     .
sex:hdl     .
sex:tch     .
sex:ltg     .
sex:glu     .
bmi:map    88.262202
bmi:tc      .
bmi:ldl     .
bmi:hdl     .
bmi:tch     .
bmi:ltg     .
bmi:glu     .
map:tc      .
map:ldl     .
map:hdl     .
map:tch     .
map:ltg     .
map:glu     .
```

Looks like the model that is using lambda value one standard deviation away has improve in comparison with lasso used with minimum value of lambda. Less number of variables are used now and most of them has been penalized keeping the cost to its minimum.

References

"Diabetes data". *North Carolina State University*. https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

Sajjad, B. (12, June 2017). "Lasso Regression in R exercises". *Rexercises.* https://www.r-exercises.com/2017/06/12/lasso-regression-in-r-exercises/?utm_source=rss&utm_medium=rss&utm_campaign=lasso-regression-in-r-exercises