***BOSTON HOUSING – LASSO, RIDGE AND ELASTIC NET REGRESSION.***

Abstract

This DOCUMENT focuses on the regularization techniques which are hyper tuning techniques in other words. These techniques are used to decrease the errors in models and avoid overfitting of the parameters. I have used the Boston Housing dataset and built Ridge and Lasso regularization models to display these techniques.

Keywords: Lasso, Ridge, Linear, Logistic, Regression, Boston, Housing.

[Title Here, up to 12 Words, on One to Two Lines]

First, we need to install the packages needed and activate them.

*Code:*

*install.packages("glmnet")*

*install.packages("caret")*

*install.packages("mlbench")*

*install.packages("psych")*

*library(glmnet)*

*library(caret)*

*library(mlbench)*

*library(psych)*

*?BostonHousing*

*data("BostonHousing")*

*mydata <- BostonHousing*

I have stored the Boston Housing dataset in mydata so every time we do not have to write Boston Housing.

*Code:*

*mydata <- BostonHousing*

Following Code shows the exploration and understanding of he data set.

*Code:*

*nrow(mydata)*

*ncol(mydata)*

*str(mydata)*

Output:

> nrow(mydata)

[1] 506

> ncol(mydata)

[1] 14

> str(mydata)

'data.frame': 506 obs. of 14 variables:

$ crim : num 0.00632 0.02731 0.02729 0.03237 0.06905 ...

$ zn : num 18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...

$ indus : num 2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...

$ chas : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...

$ nox : num 0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...

$ rm : num 6.58 6.42 7.18 7 7.15 ...

$ age : num 65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...

$ dis : num 4.09 4.97 4.97 6.06 6.06 ...

$ rad : num 1 2 2 3 3 3 5 5 5 5 ...

$ tax : num 296 242 242 222 222 222 311 311 311 311 ...

$ ptratio: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...

$ b : num 397 397 393 395 397 ...

$ lstat : num 4.98 9.14 4.03 2.94 5.33 ...

$ medv : num 24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...

So, the Boston Housing dataset has 506 observations along with 14 variables.

*?BostonHousing* will show the data dictionary required to understand the variables.

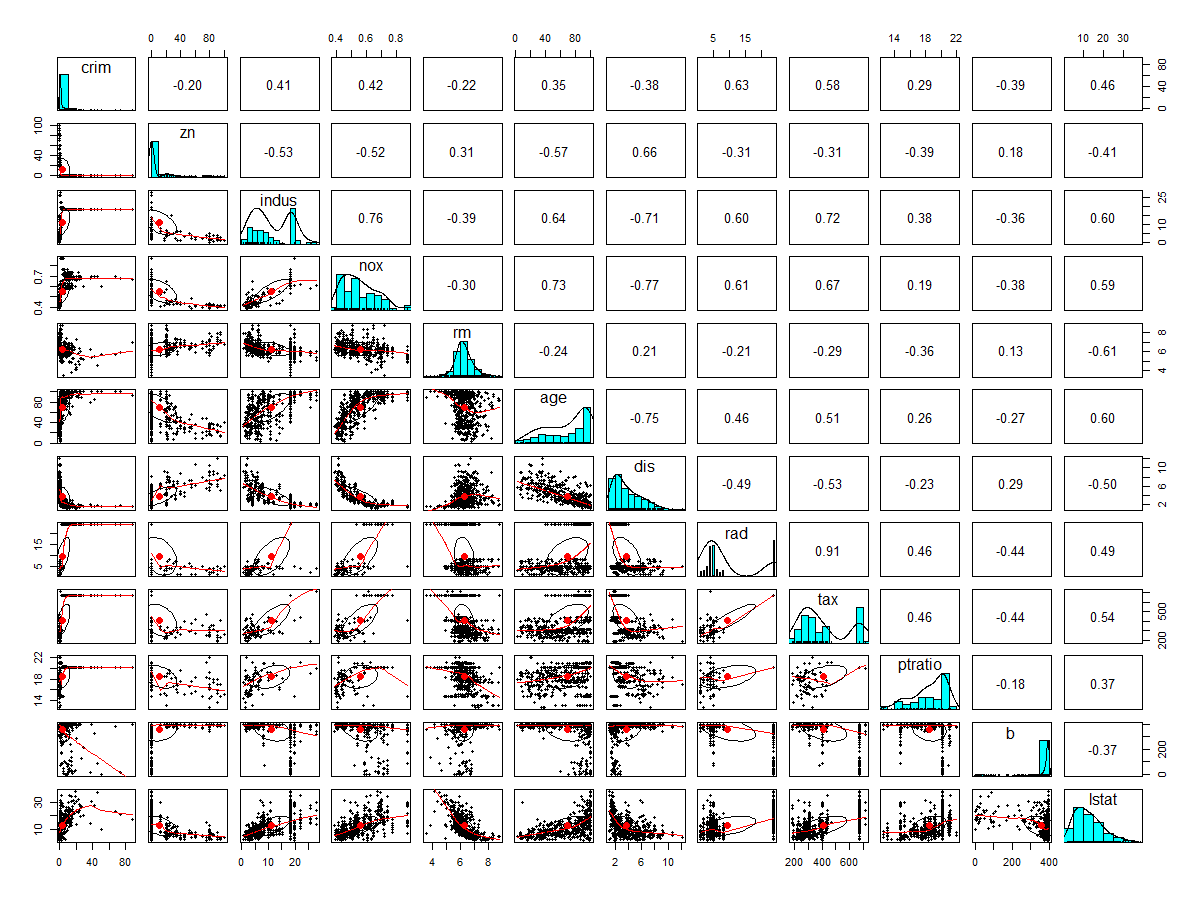
For example, crim – per capita crime rate by town. So here I have made the prediction model of medv – median value of owner-occupied homes in USD 1000s. Here all the variables are numeric variables except “chas” variable which has 2 factor levels i.e. 0 and 1.

The following code shows the correlation between independent variables but “chas” and “medv” variables are excluded.

*Code:*

*pairs. panels(mydata[c(-4,-14)], cex = 1)*

Output:



This is the scatterplot of every possible combination of independent numeric variables.

If these variables are highly correlated than it will lead to a multi collinearity problem.

So, these collinearities may lead to the overfitting.

**STEP 1: DATA DIVISON**

So, first step comes here is Data Division. Here the seed is set to get the repeatable results. 2 independent samples are taken with replacement sampling and the data set is divided into 70:30 ratio of training and testing respectively.

*Code:*

*set.seed(222)*

*independent <- sample(2, nrow(mydata),replace = TRUE, prob = c(0.7,0.3)) # sampling with replacement*

*train <- mydata[independent == 1,]*

*test <- mydata[independent == 2,]*

So now training data contains 353 observations and testing data has 153 observations.

**STEP 2: CROSS VALIDATION**

Train control function is used which is included in caret package to create the custom control parameters with method “repeated cross validation”.

*Code:*

custom <- trainControl(method = "repeatedcv",

number = 10,

repeats = 5, verboseIter = T)

Here we have used “number = 10” which breaks the data into 10 parts from which 9 parts are used to make the model and 1 part is used for error estimation. This process is repeated 10 times with each different part for error estimation. This is repeated 5 times and “verboseIter = T” shows what is going on while the model is running.

# MULTIPLE LINEAR REGRESSION

*Code:*

*set.seed(1234)*

*lm <- train(medv ~. ,*

*train,*

*method = 'lm', trControl = custom)*

For consistency “set.seed()” is used. Here, train function is used with linear model method and custom control.

*Code:*

*lm*

Linear Regression

353 samples

13 predictor

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 316, 318, 318, 319, 317, 318, ...

Resampling results:

RMSE Rsquared MAE

4.23222 0.778488 3.032342

*lm$results*

intercept RMSE Rsquared MAE RMSESD RsquaredSD MAESD

1 TRUE 4.23222 0.778488 3.032342 0.9833981 0.09350015 0.4154734

This shows that the model is with intercept, R squared shows the coefficient of determination which displays that there is more than 77% of variability in response to the “medv”

*summary(lm).*

Output:

Call:

lm(formula = .outcome ~ ., data = dat)

Residuals:

Min 1Q Median 3Q Max

-10.1018 -2.3528 -0.7279 1.7047 27.7868

Coefficients:

Estimate Std. Error t value Pr(>|t|)

(Intercept) 25.742808 5.653389 4.554 7.37e-06 \*\*\*

crim -0.165452 0.036018 -4.594 6.15e-06 \*\*\*

zn 0.047202 0.015401 3.065 0.002352 \*\*

indus 0.013377 0.067401 0.198 0.842796

chas1 1.364633 0.947288 1.441 0.150630

nox -13.065313 4.018576 -3.251 0.001264 \*\*

rm 5.072891 0.468889 10.819 < 2e-16 \*\*\*

age -0.028573 0.013946 -2.049 0.041247 \*

dis -1.421107 0.208908 -6.803 4.66e-11 \*\*\*

rad 0.260863 0.070092 3.722 0.000232 \*\*\*

tax -0.013556 0.004055 -3.343 0.000922 \*\*\*

ptratio -0.906744 0.139687 -6.491 3.03e-10 \*\*\*

b 0.008912 0.002986 2.985 0.003040 \*\*

lstat -0.335149 0.056920 -5.888 9.40e-09 \*\*\*

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.192 on 339 degrees of freedom

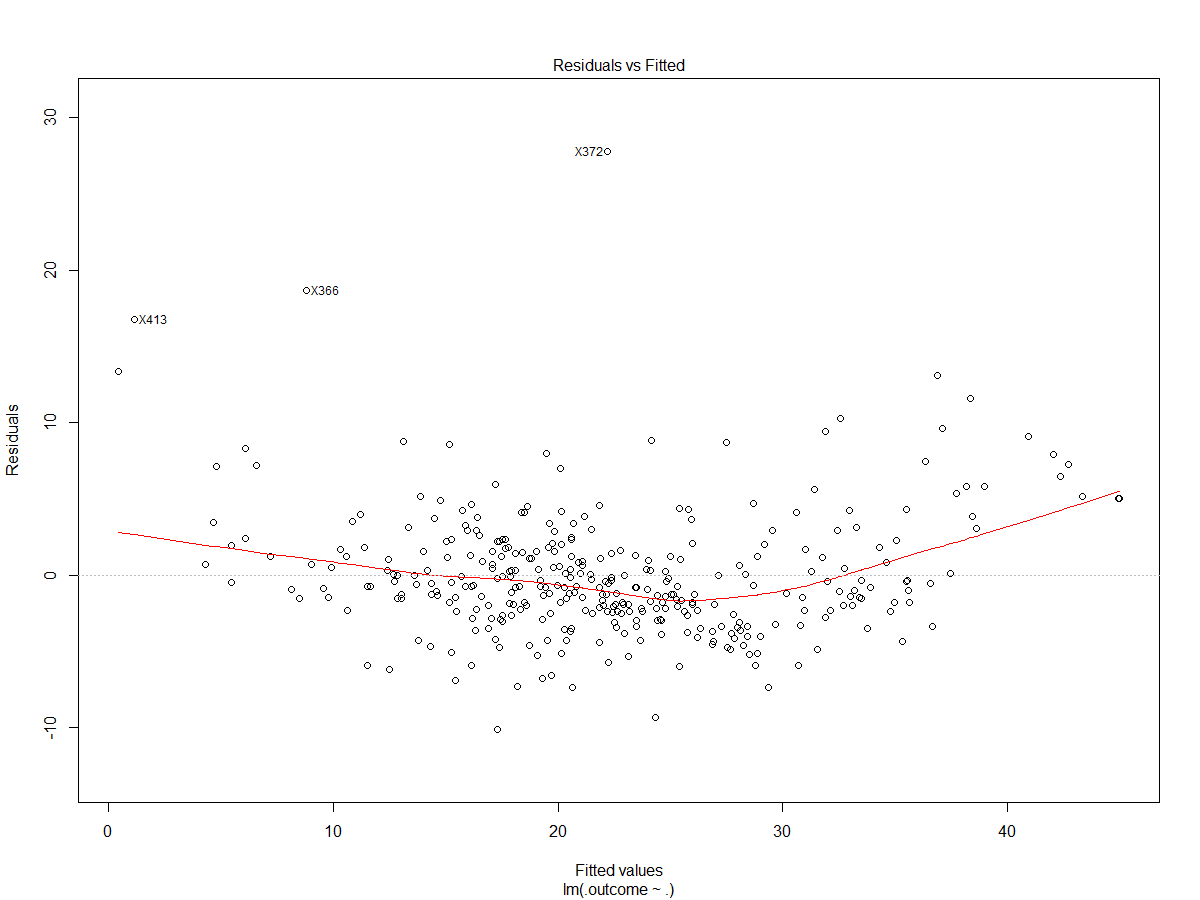
Multiple R-squared: 0.7874, Adjusted R-squared: 0.7793

F-statistic: 96.59 on 13 and 339 DF, p-value: < 2.2e-16

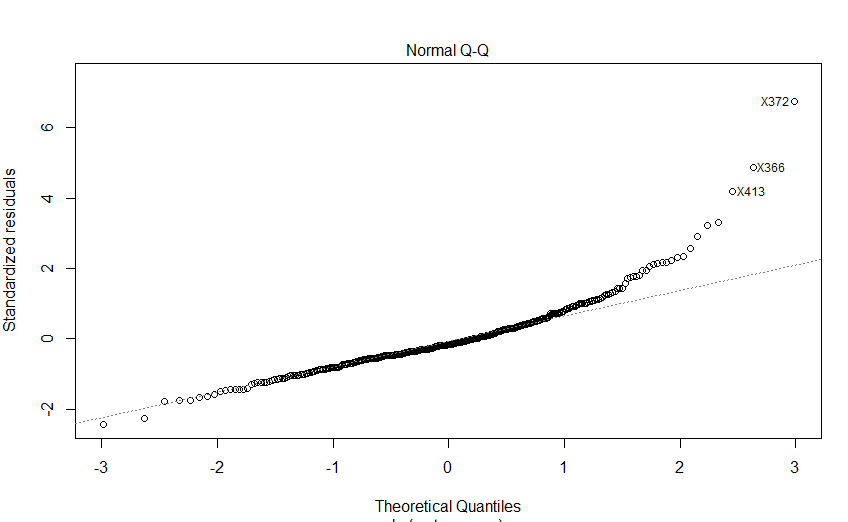
The variables which do not have a star are statistically insignificant.

Let’s plot the final model

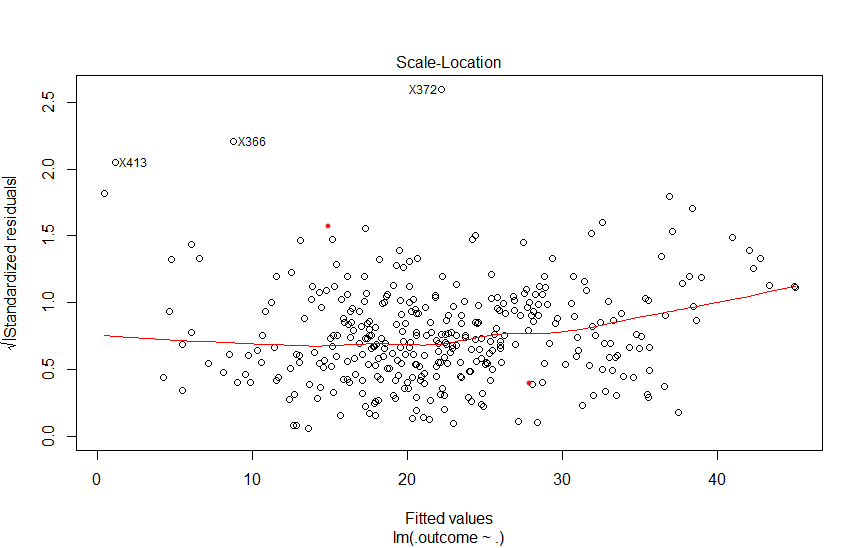
*plot(lm$finalmodel)*

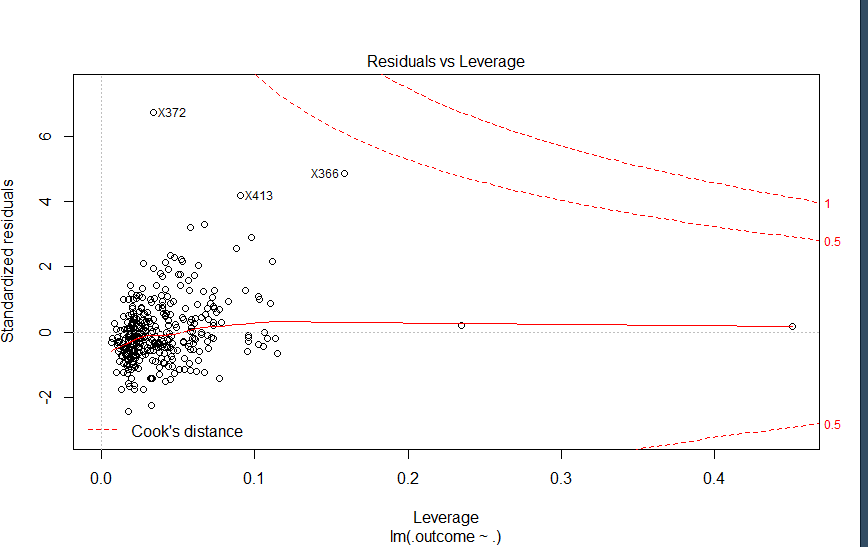


The first plot shows fitted values vs residuals.



This is a standardized residual plot. Usually all the points should fall on the line.





This plot is for leverage vs standardized residuals.

## RIDGE REGRESSION

Ridge regression tries to shrink the coefficients in the model but keeps all the variables of the model.

*Code:*

*set.seed(1234)*

*RIDGE <- train(medv~.,train,*

*method = 'glmnet',*

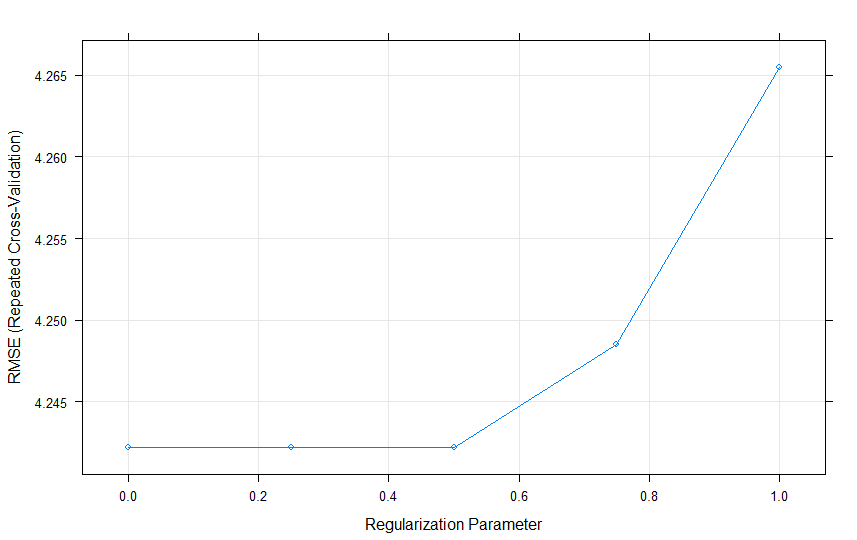
*tuneGrid = expand.grid(alpha=0,*

*lambda = seq(0.0001,1,length = 5)),*

*trControl = custom)*

Here, the model is named as RIDGE, ‘glmnet (general linear model)’ method is used. The alpha for ridge regression is 0 and we have 5 values between 0.0001 and 5. After running this model it shows that best value for lambda is 0.5. As the lambda increases we increase the penalty which will make coefficients to shrink, and when lambda decreases the penalty also decreases.

*plot(RIDGE)*

**

Here on the y-axis we can see Root Means Squared Error with respect to repeated cross validation and as the value of lambda increases error also increases. So the best value for lambda is 0.5

353 samples

13 predictor

No pre-processing

Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 316, 318, 318, 319, 317, 318, ...

Resampling results across tuning parameters:

lambda RMSE Rsquared MAE

0.000100 4.242204 0.7782278 3.008339

0.250075 4.242204 0.7782278 3.008339

0.500050 4.242204 0.7782278 3.008339

0.750025 4.248536 0.7779462 3.012397

1.000000 4.265479 0.7770264 3.023091

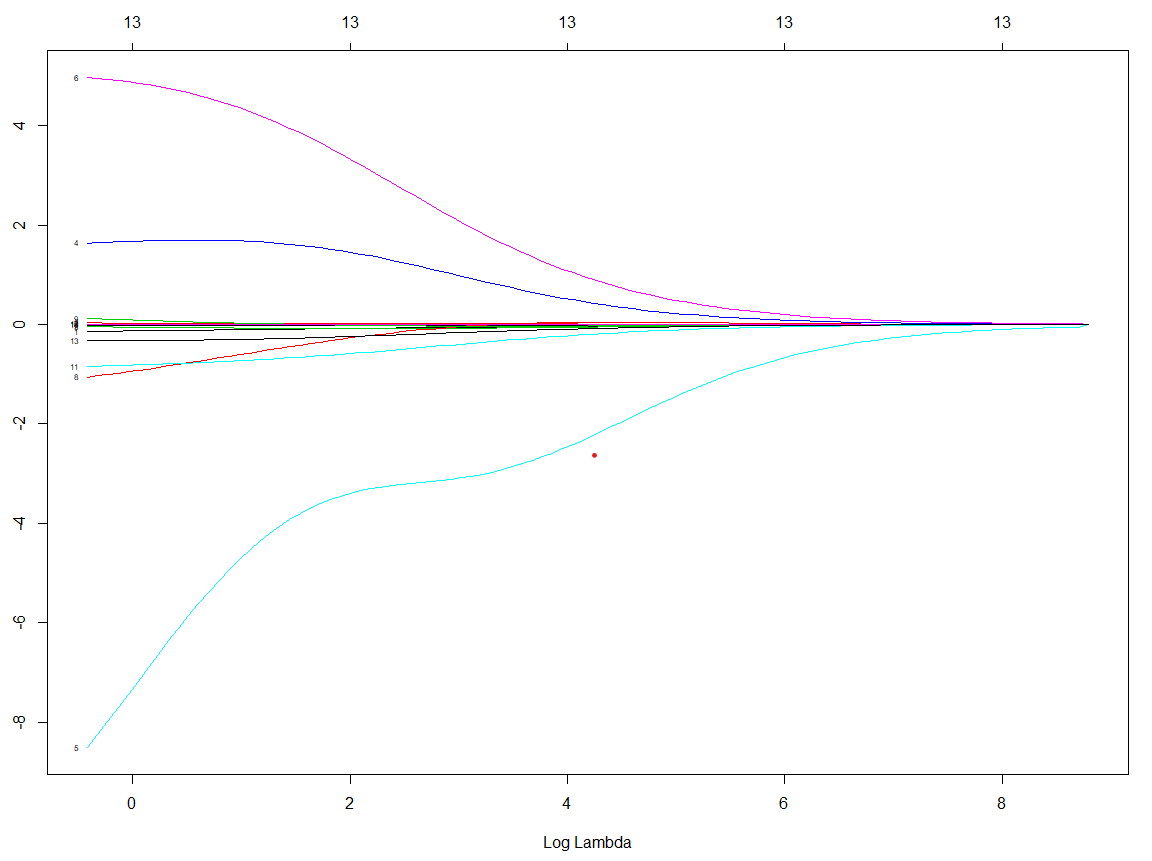
Tuning parameter 'alpha' was held constant at a value of 0

RMSE was used to select the optimal model using the smallest value.

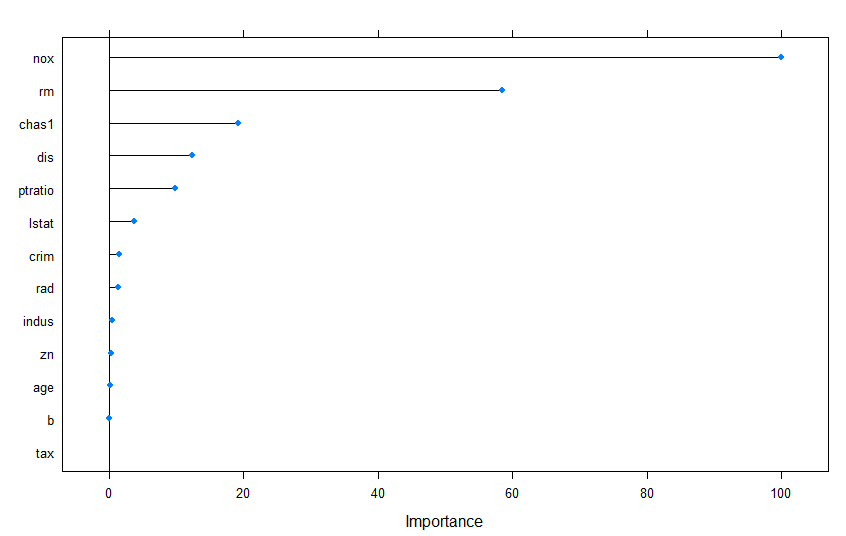
The final values used for the model were alpha = 0 and lambda = 0.50005.

*Code:*

*plot(RIDGE$finalModel,xvar = "lambda", label = TRUE)*



Here, we can see that x-axis = Log Lambda and y-axis = coefficients. At the top it shows we have all 13 variables. So, as the lambda increases all the coefficients are zero and as the lambda decreases the coefficients starts to grow.



Here we can see that “nox” and “rm” are the most important variables while “b” and “tax” are the least useful variables.

*plot(RIDGE$finalModel,xvar = "dev")*

*plot(varImp(RIDGE, scale=T))*

### LASSO

In lasso if there are highly corelated variables then it will select one variable and ignore others.

*Code:*

***set.seed(1234)***

***lasso <- train(medv~.,train,***

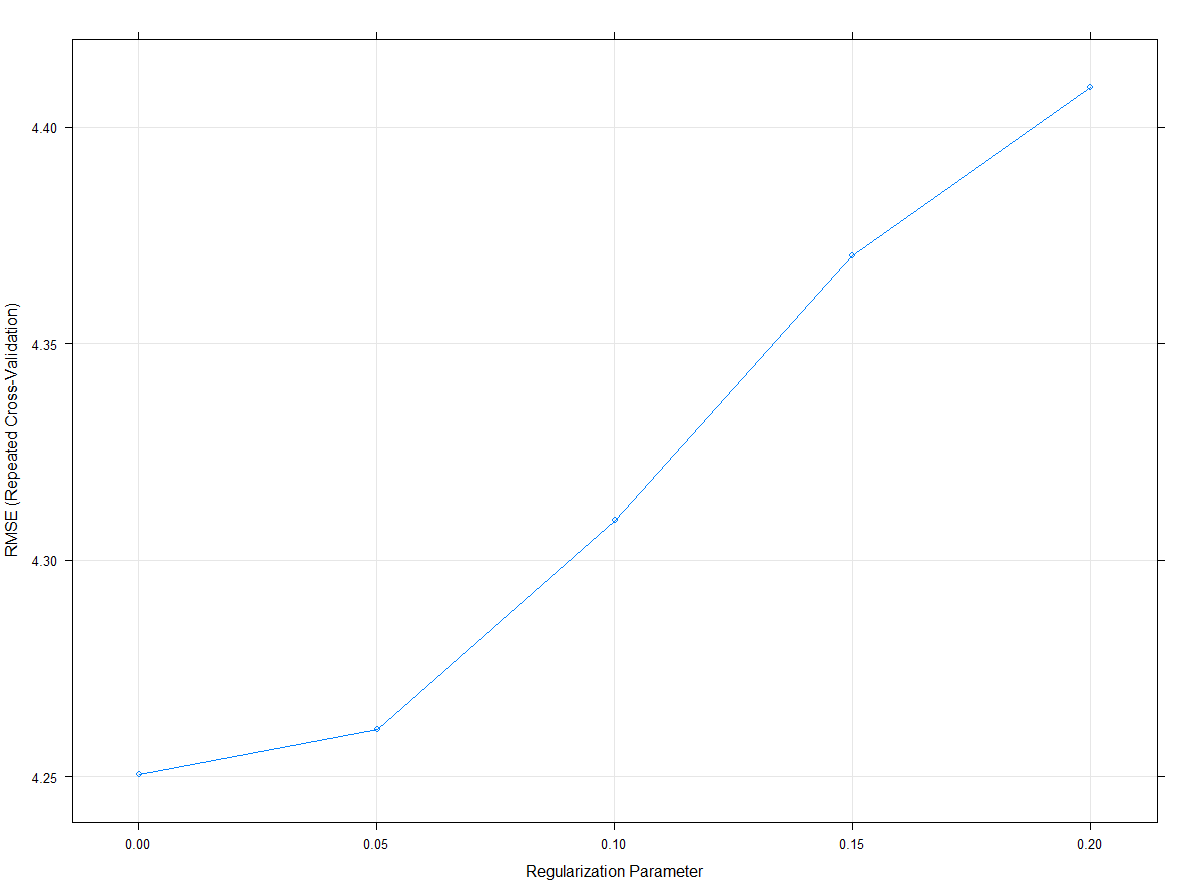
***method = 'glmnet',***

***tuneGrid = expand.grid(alpha=1,***

***lambda = seq(0.0001,1,length = 5)),***

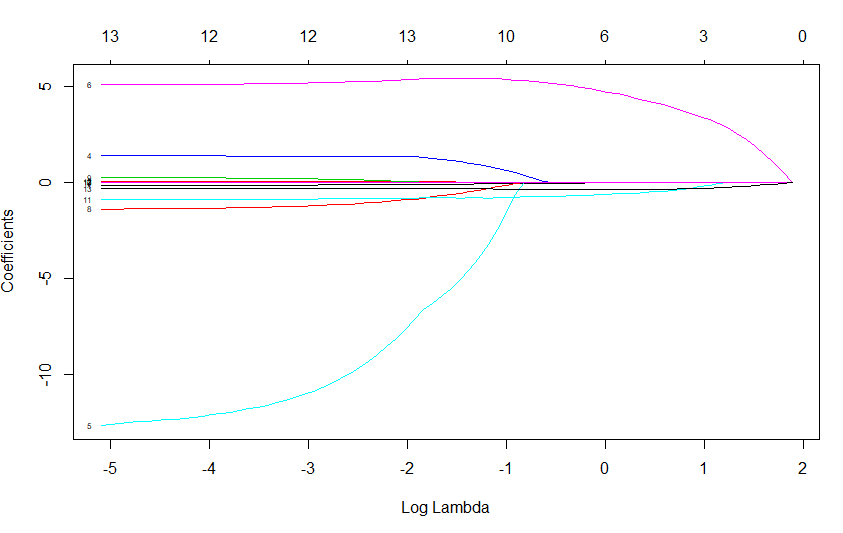
***trControl = custom)***

Here the lambda is very small i.e. 1e-04.



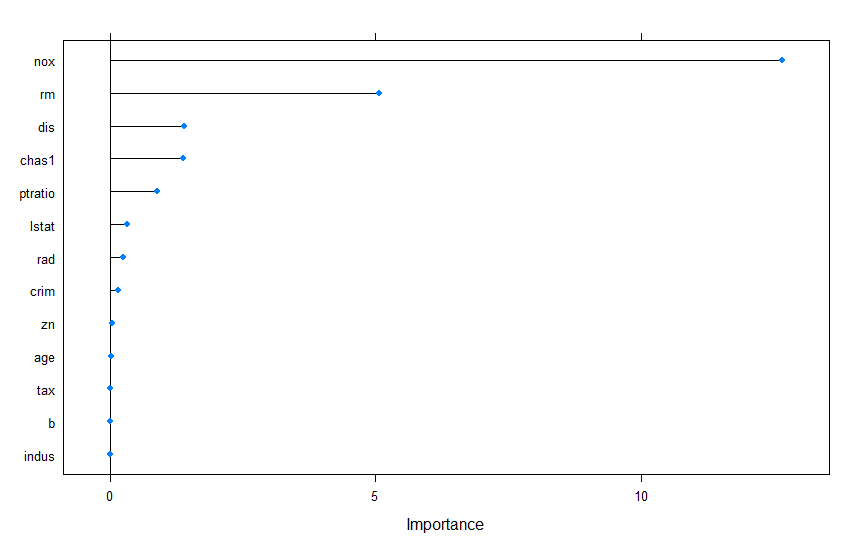
*Code:*

***plot(lasso$finalModel,xvar = "lambda",label = T)***

******

Here, we can see that 6th variable i.e. rm is performing very well while the 5th variable i.e. nox is shrinking.

***plot(lasso$finalModel,xvar = "dev")***

***plot(varImp(lasso,scale = 'F'))***

The above plot is variable importance plot which shows that nox and rm are at the top.

**ELASTICNET REGRESSION**

***Code:  
set.seed(1234)***

***elasticnet <- train(medv~.,train,***

***method = 'glmnet',***

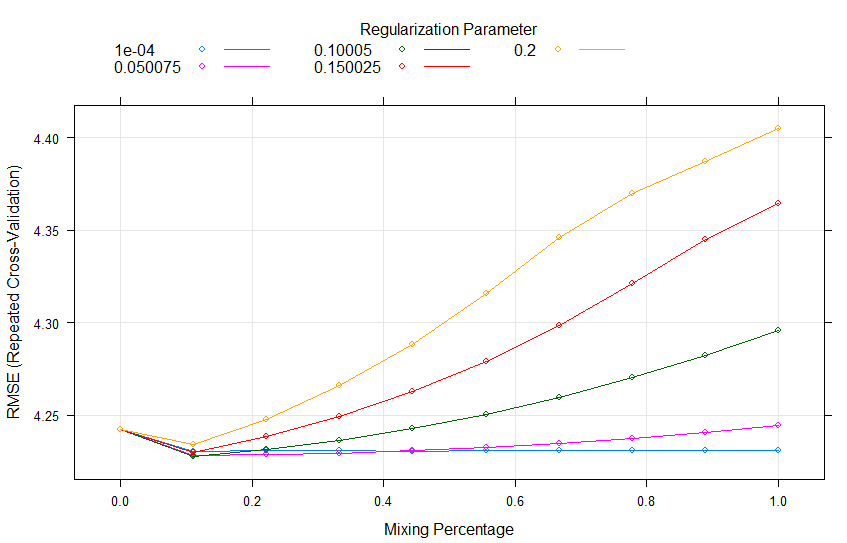
***tuneGrid = expand.grid(alpha=seq(0,1,length=10),***

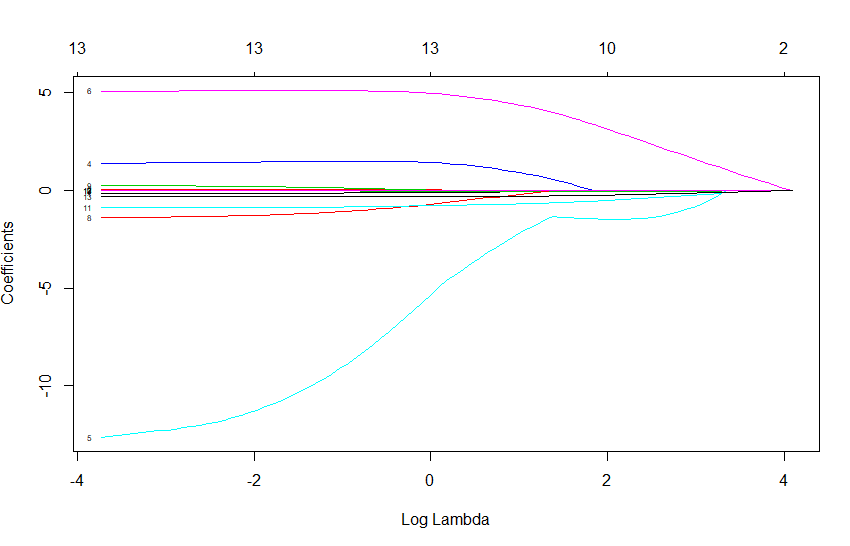
***lambda = seq(0.0001,0.2,length = 5)),***

***trControl = custom)***

***plot(elasticnet,xvar='lambda',label = T)***

***plot(elasticnet$finalModel,xvar = 'lambda',label = TRUE)***

******

******

#### COMPARISON OF MODELS.

*Code:*

*models <- list(linear\_model = lm, Ridge = RIDGE, Lasso = lasso*, *Elastic = elasticnet* *)*

*resamples <- resamples(models)*

*summary(resamples)*

The resamples method which is stored in the caret package helps us to compare the models.

MAE

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

linear\_model 2.080208 2.767061 3.002455 3.032342 3.355281 3.874270 0

Ridge 2.094151 2.736246 2.934350 3.008339 3.366834 3.971337 0

Lasso 2.273042 2.670495 2.972924 3.043786 3.300927 4.390683 0

RMSE

Min. 1st Qu. Median Mean 3rd Qu. Max. NA's

linear\_model 2.673817 3.495197 3.998562 4.232220 4.751509 7.027551 0

Ridge 2.478993 3.477912 4.169422 4.242204 4.759265 7.035089 0

Lasso 2.916518 3.538142 4.011283 4.250579 4.755815 7.207666 0

Rsquared

Min. 1st Qu. Median Mean 3rd Qu. Max.

linear\_model 0.4865769 0.7269864 0.7991104 0.7784880 0.8472274 0.9128278

Ridge 0.4796929 0.7339342 0.8018589 0.7782278 0.8459744 0.9141020

Lasso 0.3779540 0.7432619 0.7994717 0.7674937 0.8396132 0.9200985

NA's

The summary of resamples shows us the Quartiles, Mean and Median for all the models(Linear model, Ridge and Lasso) for MAE (Mean Absolute Error), RMSE (Root Mean Squared Error) and Rsquared.

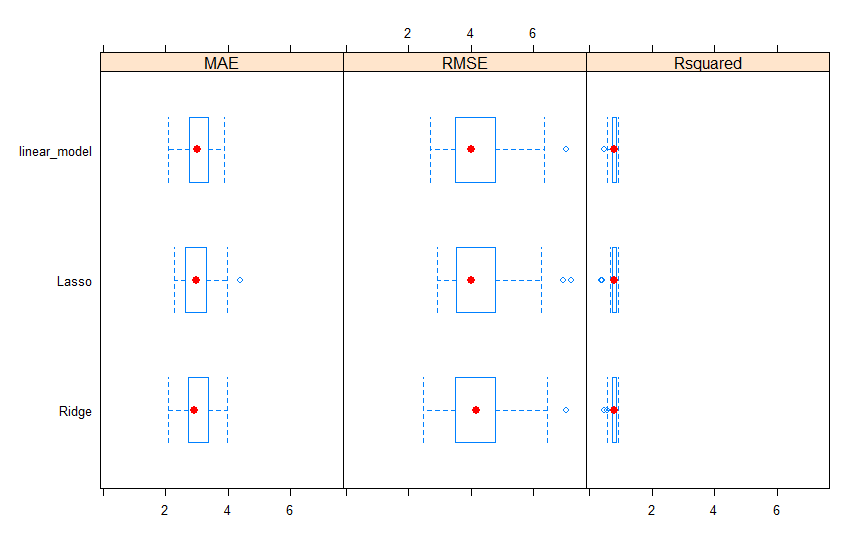
The lowest mean for Root Mean Squared Error is of linear model (4.232220) and the lowest mean for MAE is of Ridge model which is 3.008339

Boxplot of resamples:

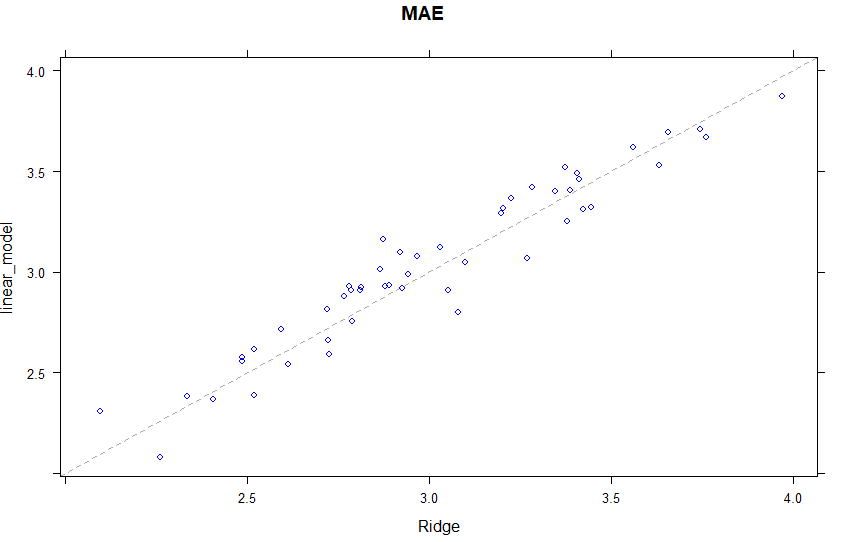
*Code:*

*?xyplot*

*bwplot(resamples,col = 'red')*

**

*xyplot(resamples,col = 'blue')*

**

The dots above the dotted line shows that the models perform better when we have ridge regression models and when the dots are below the dotted lines, models perform better when we have linear regression models.

To check the best values for the Lasso model

*Code;*

*lasso$bestTune*

Output:

alpha lambda

1 1 1e-04

So, the best values of alpha is 1 and for lambda is 1e-04

*bestmodel <- lasso$bestTune*

*coef(bestmodel, s = lasso$bestTune$lambda)*

##### **PREDICITION**:

***Code:***

***predection <- predict(lasso,train)***

***sqrt(mean(train$medv-predection)^2)***

This will give the root mean square error for training data.

Output:

4.511352e-14

***Code:***

***predection1 <- predict(lasso,test)***

***sqrt(mean(test$medv-predection1)^2)***

This will give the root mean square error for test data.

Output:

[1] 0.553538

**EXERCISES**

Exercise 1:

The lars (Least Angle Regression Annals of Statistics) package is loaded which has the diabetes dataset. This dataset has the patient level data on the progression of diabetes. Glmnet (General Linear Model) package is loaded which is used to implement Lasso.

*Code:*

***install.packages(lars)***

***library(lars)***

***library(glmnet)***

***data(diabetes)***

***attach(diabetes)***

***str(diabetes)***

***head(diabetes)***

***nrow(diabetes)***

***ncol(diabetes)***

Output:

> str(diabetes)

'data.frame': 442 obs. of 3 variables:

$ x : 'AsIs' num [1:442, 1:10] 0.03808 -0.00188 0.0853 -0.08906 0.00538 ...

..- 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' num [1:442, 1:64] 0.03808 -0.00188 0.0853 -0.08906 0.00538 ...

..- attr(\*, ".Names")= chr "age" "age" "age" "age" ...

..- attr(\*, "dimnames")=List of 2

.. ..$ : chr "1" "2" "3" "4" ...

.. ..$ : chr "age" "sex" "bmi" "map" ...

> nrow(diabetes)

[1] 442

> ncol(diabetes)

[1] 3

So these dataset has 3 columns and 442 rows.

Exercise2:

Three matrices in diabetes dataset i.e. x and x2 contains smaller set of independent variables, quadratic and interaction terms while y measures the progression of diabetes and also it is an dependent variable.

***Code:***

***summary(x)***

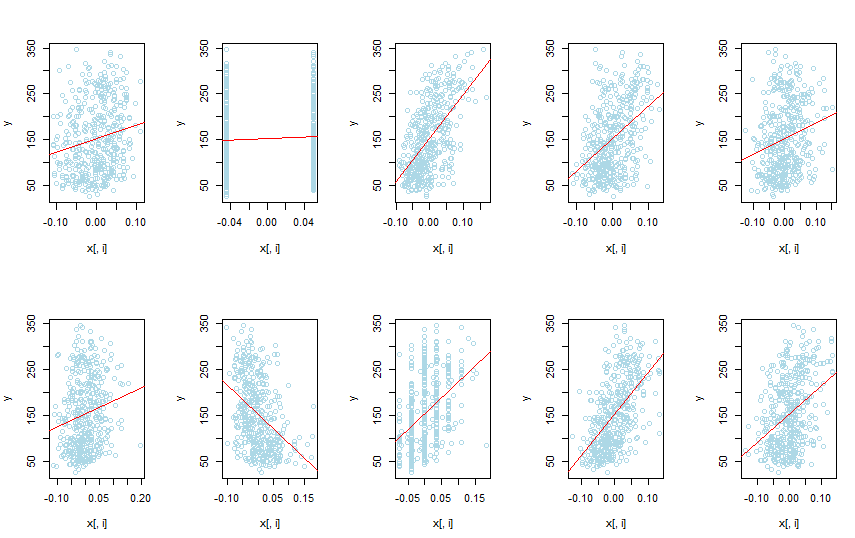
***par(mfrow = c(2,5))***

***for(i in 1:10){***

***plot(x[,i],y,col = 'lightblue')***

***abline(lm(y~x[,i]),col = 'red')***

***}***

******

Loop is used to automate the process. The above scatterplot shows the relationship of every predictors with dependent variable along with the best fit line for predictors on x and y on the vertical axis.

Exercise:

***Code:***

***model <- lm(y~x)***

***summary(model)***

*Output:*

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

Using the Ordinary Least Squares y is regressed on the predictors in x. The variables here in the output which does not have stars shows that they are statistically insignificant while the variables with three stars “\*\*\*” are highly statistically significant.

Exercise 4:

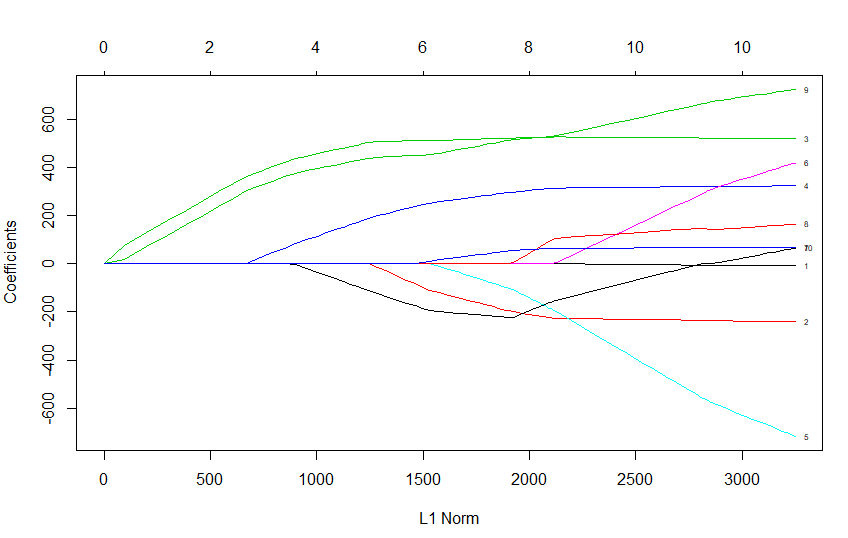
***Code:***

***lasso <- glmnet(x,y)***

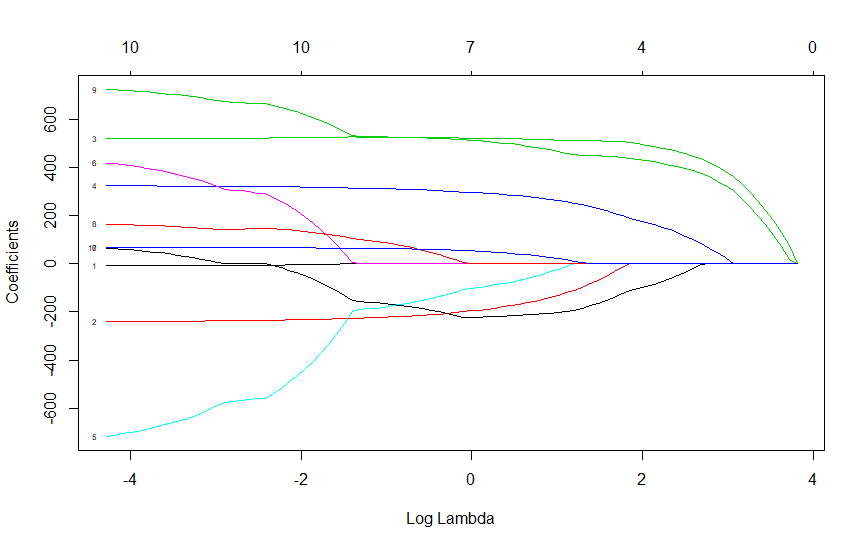
***plot(lasso, xvar = 'norm', label = T)***

***plot(lasso, xvar = ‘lambda’, label = T)***

Output:



The above plot shows the variable coefficients against L1 norm and describes when the coefficients shrinks to zero. 9th and 3rd variable shrinks to zero as we can see.



On the x-axis we have log lambda and on y-axis coefficients. On the top we can see we have all the 10 variables. As the lambda increases we can see the coefficients shrink almost to zero but as the lambda decreases the coefficients starts growing.

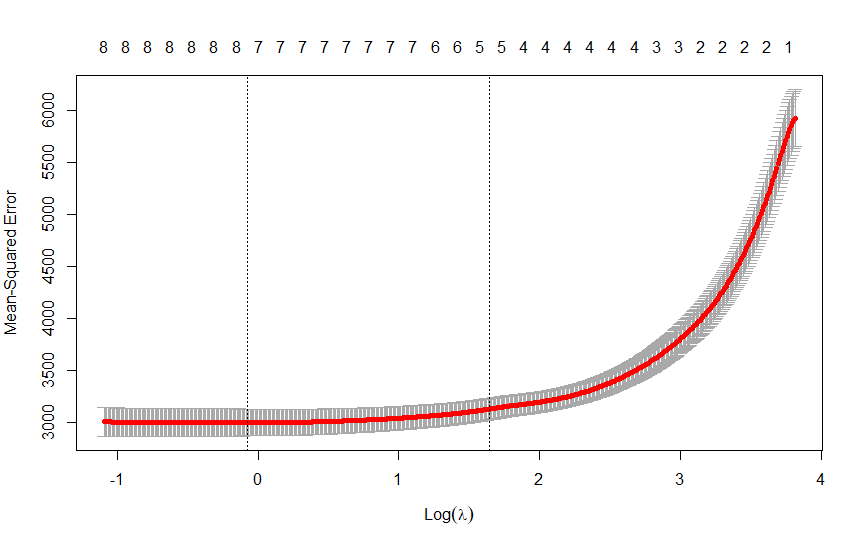
Exercise 5:  
The plot below shows the cross-validation curve and lambda value that minimizes the cross- validation error.

***Code:***

***fit <- cv.glmnet(x=x,y=y,alpha = 1,nlambda = 999)***

***plot(fit)***

***fit$lambda.min***



This plot shows the log lambda vs Mean Squared Error.

Output:

> fit$lambda.min

[1] 0.92764

So, 0.92764 is the minimum value of lambda which reduces the cross-validation error.

Exercise 6:

***Code:***

***fit1 <- glmnet(x=x, y=y, alpha = 1, lambda = fit$lambda.min)***

***fit1$beta***

Output:

10 x 1 sparse Matrix of class "dgCMatrix"

s0

age .

sex -198.785452

bmi 522.288820

map 297.739059

tc -105.907479

ldl .

hdl -222.903948

tch 2.178174

ltg 515.068573

glu 55.086593

Here we have taken the minimum value of lambda and alpha = 1 using the glmnet function. This will show the beta matrix which includes all 10 variables. The values of these predictors explain the variations in y.

Exercise 7:

***Code:***

***fit$lambda.1se***

***fit2 <- glmnet(x=x,y=y,alpha = 1,lambda = fit$lambda.1se)***

***fit2***

***fit2$beta***

Output:

[1] 5.162623

Call: glmnet(x = x, y = y, alpha = 1, lambda = fit$lambda.1se)

Df %Dev Lambda

1 5 0.488 5.163

10 x 1 sparse Matrix of class "dgCMatrix"

s0

age .

sex -39.14719

bmi 508.61511

map 213.68417

tc .

ldl .

hdl -143.16915

tch .

ltg 445.40263

glu .

Here the value of lambda is increased, and beta coefficients shows that many of them are shrunk to zero.

Exercise 8:

***Code:  
model1 <- lm(y~x2)***

***summary(model1)***

Output:  
Call:

lm(formula = y ~ x2)

Residuals:

Min 1Q Median 3Q Max

-158.216 -30.809 -3.857 31.348 153.946

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

x2tc:glu -176.295 595.459 -0.296 0.7673

x2ldl:hdl 2642.645 3165.926 0.835 0.4044

x2ldl:tch 1206.822 1470.512 0.821 0.4123

x2ldl:ltg 2773.697 10960.214 0.253 0.8004

x2ldl:glu 85.626 505.102 0.170 0.8655

x2hdl:tch 1188.406 1002.242 1.186 0.2365

x2hdl:ltg 1467.845 4609.793 0.318 0.7503

x2hdl:glu 217.541 296.749 0.733 0.4640

x2tch:ltg 389.805 624.671 0.624 0.5330

x2tch:glu 235.693 235.064 1.003 0.3167

x2ltg:glu 83.525 264.726 0.316 0.7525

---

Signif. codes: 0 ‘\*\*\*’ 0.001 ‘\*\*’ 0.01 ‘\*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 53.23 on 377 degrees of freedom

Multiple R-squared: 0.5924, Adjusted R-squared: 0.5233

F-statistic: 8.563 on 64 and 377 DF, p-value: < 2.2e-16

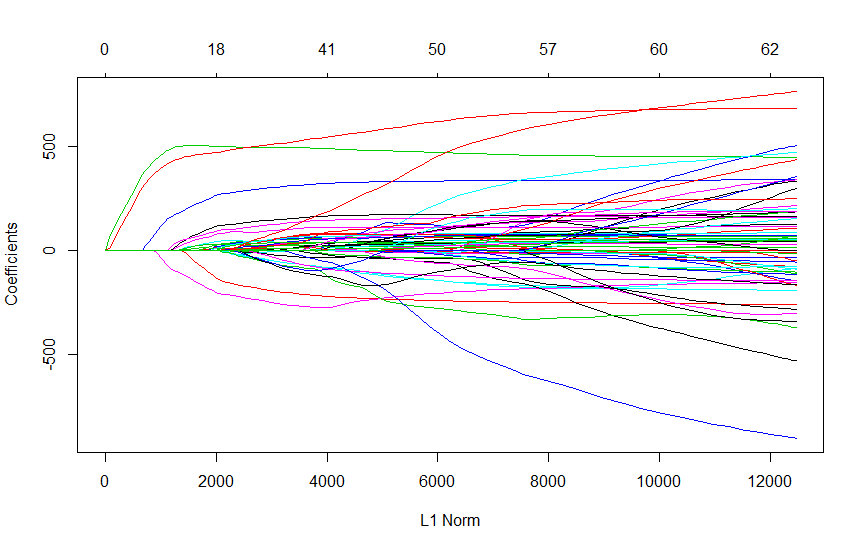
Here y is regressed on x2. Only few variables are statistically significant.

Exercise 9:

***Code:***

***lasso2 <- glmnet(x2,y)***

***plot(lasso2,xvar = "norm")***

Output:  


These lines show the different values taken by the coefficients and lambda is the weight given to the L1 norm. As the lambda approaches zero the loss function approaches the OLS loss function.

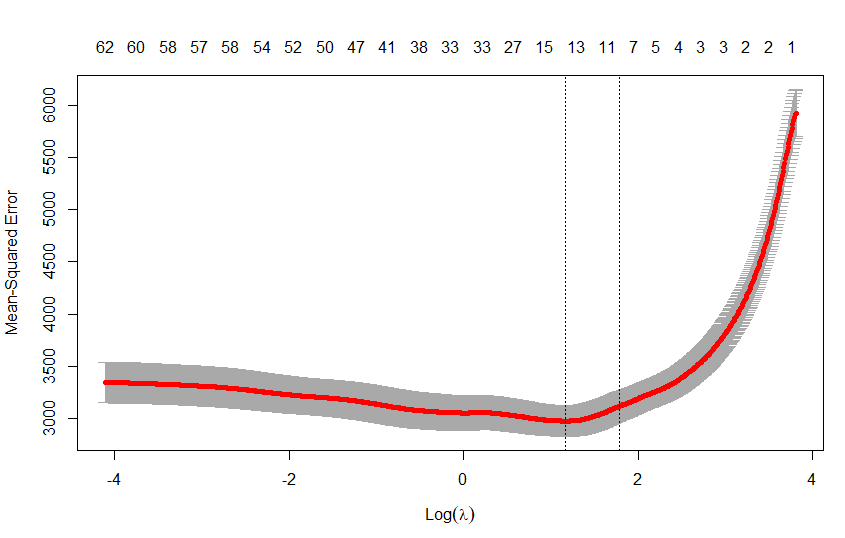
Exercise 10:

***Code:***

***fit3 <- cv.glmnet(x=x2,y=y,alpha=1,nlambda = 999)***

***plot(fit3)***

Output:



This plot shows log lambda vs Mean squared error. At the top we can see the variables and their value of log lambda and which coefficients have shrunk to zero

References

Supplemental Information 1: K–M plots of LASSO, Elastic Net and Ridge models in training cohort. (n.d.). doi: 10.7717/peerj.8827/supp-1