REGRESSION ANALYSIS ON BOSTON HOUSE PRICE PREDICTION

By,
PAVITHRA D, 21-PST-025
Loyola College

ABSTRACT

This study proposes a performance comparison between machine learning regression algorithms. The regression algorithms used in this study are Multiple linear, Least Absolute Selection Operator (Lasso), Ridge, Elastic Net and Polynomial Regression. Moreover, this study attempts to analyze the correlation between variables to determine the most important factors that affect house prices in Boston.

The accuracy of the prediction is evaluated by checking the R-square and root mean square error scores of the training model. The test is performed after applying the required pre-processing methods and splitting the data into two parts. However, one part will be used in the training and the other in the test phase.

This thesis attempts to show that Poly Ridge gives the best score among other algorithms when using the dataset in training. The correlation graphs show the variables' level of dependency. In addition, the empirical results show that crime, pupil-teacher ratio, nitric oxide concentration and distance between city and office influence the house prices negatively. Whereas river, rooms, and radial highway rate impact the house prices positively.

ACKNOWLEDGEMENT

We would like to express our special thanks of gratitude to our Professor and mentor, Mr. Selva Arul Pandian who helped us in doing a lot of research on regression analysis and the models. His valuable guidance helped us patch this project.

And this project wouldn't have been possible without the support of the team members who helped each other in finishing this project within the limited time.

Table of Contents

INTRODUCTION:

Machine learning is a subfield of Artificial Intelligence (AI) that works with algorithms and technologies to extract useful information from data. Machine learning methods are appropriate in big data ,since attempting to manually process vast volumes of data would be impossible without the support of machines.

The performance will be measured upon predicting house prices since the prediction in many regression algorithms relies not only on a specific feature but on an unknown number of attributes that result in the value to be predicted. House prices depend on an individual house specification. Houses have a variant number of features that may not have the same cost due to its location. For instance, a big house may have a higher price if it is located in a desirable rich area than being placed in a poor neighbourhood.

The data used in the experiment will be handled by using a combination of pre-processing methods to improve the prediction accuracy. In addition, some factors will be added to the local dataset in order to study the relationship between these factors and the MEDV price in Boston.

In this project, we will evaluate the performance and predictive power of a model that has been trained and tested on data collected from homes in suburbs of Boston, Massachusetts. A model trained on this data that is seen as a good fit could then be used to make certain predictions about a home's monetary value.

PROBLEM IDENTIFICATION:

The Boston Housing dataset was drawn from the Boston Standard Metropolitan Statistical Area (SMSA) in 1970. Each of the 506 rows in the dataset describes a Boston suburb or town, and it has 14 columns with information such as average number of rooms per dwelling, pupil-teacher ratio, and per capita crime rate. The last row describes the median price of owner-occupied homes (this leaves out homes that are rented out), and it's the row that we are trying to predict when we use it for regression tasks.

The attributes of the dataset is given below:

1. CRIM: per capita crime rate by town.
2. ZN: proportion of residential land zoned for lots over 25,000 sq.ft.
3. INDUS: proportion of non-retail business acres per town.
4. CHAS: Charles River dummy variable (= 1 if tract bounds river; 0 otherwise).
5. NOX: nitric oxides concentration (parts per 10 million).
6. RM: average number of rooms per dwelling.
7. AGE: proportion of owner-occupied units built prior to 1940.
8. DIS: weighted distances to five Boston employment centers.
9. RAD: index of accessibility to radial highways.
10. TAX: full-value property-tax rate per $10,000.
11. PTRATIO: pupil-teacher ratio by town.
12. B: 1000(Bk−0.63)2 where Bk is the proportion of blacks by town.
13. LSTAT: % lower status of the population.
14. MEDV: Median value of owner-occupied homes in $1000s

OBJECTIVE:

The main objective of the study is to determine the MEDV (Median value of owner_occupied homes) Price value and the factors which are influencing it,i.e, the 13 explanatory variables. This is done by fitting the data to the following Regression Models and find out the best fitting regression model amongst that.

The Regression Models which are used here to predict the MEDV are as follows:

- Multiple Linear Regression
- Polynomial Regression
- Ridge Regression
- Lasso Regression
- Elastic Net Regression

MODEL DESCRIPTION:

1. MULTIPLE LINEAR REGRESSION :

Regression models are used to describe relationships between variables by fitting a line to the observed data.

Multiple linear regression is used to estimate the relationship between two or more independent variables and one dependent variable.

Multiple linear regression formula

The formula for a multiple linear regression is:

$$y = \beta_0 + \beta_1 X_1 + ... + \beta_n X_n + \varepsilon$$

- y = the predicted value of the dependent variable
- B_0 = the y-intercept (value of y when all other parameters are set to 0)
- B_1X_1= the regression coefficient (B_1) of the first independent variable (X_1) (a.k.a. the effect that increasing the value of the independent variable has on the predicted y value)
- … = do the same for however many independent variables you are testing
- B_nX_n = the regression coefficient of the last independent variable
- e = model error (a.k.a. how much variation there is in our estimate of y)

Assumptions of Multiple Linear Regression Model

Homogeneity of variance (homoscedasticity): the size of the error in our prediction doesn't change significantly across the values of the independent variable.

Independence of observations: the observations in the dataset were collected using statistically valid methods, and there are no hidden relationships among variables.

In multiple linear regression, it is possible that some of the independent variables are actually correlated with one another, so it is important to check these before developing the regression model.

Normality: The data should follow a normal distribution.

Linearity: the line of best fit through the data points is a straight line, rather than a curve or some sort of grouping factor.

Feature Selection in Multiple Linear Regressions

Multiple linear Regression is to examine how multiple independent variables are related to a dependent variable. We are concerned about Feature Selection. It is the important step which impacts the model and its performance. If we put all the variable into the model even the best model will perform worst as Garbage in Garbage Out. So we need to select the most relevant features and drop the irrelevant features

Stepwise linear regression is a method of regressing multiple variables while simultaneously removing those that aren't important.

Stepwise regression essentially does multiple regression a number of times, each time removing the weakest correlated variable. At the end you are left with the variables that explain the distribution best. The only requirements are that the data is normally distributed (or rather, that the residuals are), and that there is no correlation between the independent variables (known as collinearity).

2. POLYNOMIAL REGRESSION:

In multiple linear regression algorithms only work when the relationship between the data is linear. But suppose if we have non-linear data, then Linear regression will not be capable of drawing a best-fit line and it fails in such conditions. What if we can see that the straight line is unable to capture the patterns in the data? This is an example of under-fitting.
To overcome under-fitting, we need to increase the complexity of the model.

So, we introduce polynomial regression, which helps identify the curvilinear relationship between independent and dependent variables.
It is nothing, but a form of linear regression where only due to the Non-linear relationship between dependent and independent variables we add some polynomial terms to linear regression to convert it into Polynomial regression.
To generate a higher order equation we can add powers of the original features

as new features. The linear model,

$$Y = \theta_0 + \theta_1 x$$

Can be transformed to

$$Y = \theta_0 + \theta_1 x + \theta_2 x^2$$

This is still considered to be a linear model as the coefficients/weights associated with the features are still linear. x^2 is only a feature. However the curve that we are fitting is quadratic in nature.

3. RIDGE REGRESSION (L2 Regularization):

Ridge regression is a regularized linear model. Regularization techniques need to be used to prevent overfitting. Before we train a ridge model on the training, we need to find the optimal parameters for the model. To do this, grid search along with k-fold cross validation was used to find the optimal parameters that led to the best score.
The optimal parameters are then used for model evaluation using both the training and test sets.
This technique performs L2 regularization. The main algorithm behind this is to modify the RSS by adding the penalty which is equivalent to the square of the magnitude of coefficients. However, it is considered to be a technique used when the info suffers from multicollinearity (independent variables are highly correlated). In multicollinearity, the smallest amount squares estimates (OLS) are unbiased, their variances are large which deviates the observed value faraway from truth value. By adding a degree of bias to the regression estimates, ridge regression reduces the quality errors. It tends to solve the multicollinearity problem through the shrinkage parameter λ. Now, let us have a look at the equation below.
In this equation, we have two components. The foremost one denotes the least square term and the latter one is the lambda of the summation of $\beta 2$ (beta-square) where β is the coefficient. This is added to least square term so as to shrink the parameter to possess a really low variance.

Limitations of Ridge Regression:

It decreases the complexity of a model but does not reduce the number of variables since it never leads to a coefficient tending to zero rather only minimizes it. Hence, this model is not a good fit for feature reduction. We can conclude that it shrinks the parameters only. Therefore, it reduces the model complexity by shrinking the coefficient.

4.LASSO REGRESSION (L1 Regularization):

This regularization technique performs L1 regularization. Unlike Ridge Regression, it modifies the RSS by adding the penalty (shrinkage quantity) equivalent to the sum of the absolute value of coefficients.
Looking at the equation below, we can observe that similar to Ridge Regression, Lasso (Least Absolute Shrinkage and Selection Operator) also penalizes the absolute size of the regression coefficients. In addition to this, it is quite capable of reducing the variability and improving the accuracy of linear regression models.
Lasso regression model uses shrinkage technique. In this technique, the data values are shrunk towards a central point similar to the concept of mean. The lasso regression algorithm suggests a simple, sparse models (i.e. models with fewer parameters), which is well-suited for models or data showing high levels of multicollinearity or when we would like to automate certain parts of model

selection, like variable selection or parameter elimination using feature engineering.

5. ELASTIC NET:

Elastic net is a popular type of regularized linear regression that combines two popular penalties, specifically the L1 and L2 penalty functions.
Elastic Net first emerged as a result of critique on lasso, whose variable selection can be too dependent on data and thus unstable. The solution is to combine the penalties of ridge regression and lasso to get the best of both worlds. Elastic Net aims at minimizing the following loss function:
where α is the mixing parameter between ridge ($\alpha = 0$) and lasso ($\alpha = 1$).

EVALUATION METRICS:

Several evaluation metrics measure the performance of machine learning algorithms such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), R-Squared, and Mean Absolute Error (MAE). However, in this study, the performance of the algorithms is measured by using RMSE and R-Squared.

Root Mean Square Error (RMSE) is used as an evaluation metric in machine learning to measure the performance of the model. However, RMSE is similar to the Mean Square Error (MAE). Where all errors in MAE have the same weight, but RMSE penalises the variance, which means it gives more weight to the errors that have large absolute values than that have small absolute values. Therefore, when RMSE and MAE are calculated, RMSE is always bigger than MAE. RMSE is more sensitive to the errors than MAE; therefore, using RMSE for measuring the performance is better than MAE [38]. RMSE can be calculated as the square root of the sum of squared errors over the sample size n.

RMSE can be presented as:

$$RMSE = \sqrt{MSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y})^2}$$

It can be observed from the equation when the sum of squared errors is closer to zero, RMSE is closer to zero. Therefore, when RMSE is zero, it means there are no errors between the actual value y and the predicted value $y\hat{}$.R-Squared or as known as coefficient determination is a statistical measurement that is used in machine learning to measure how close the data are to the fitted regression line. R-Squared takes the value between 0 and 1. Where 1 indicates the prefect score and 0 is imperfect. In addition, R-Squared is calculated by measuring the deviations of the observations from their predicted values over the measurement of the deviations of the observations from their mean.

R-Squared can be presented as:

$$R^2 = 1 - \frac{SS_{RES}}{SS_{TOT}} = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \overline{y})^2}$$

SOFTWARE SPECIFICATION

The R-software is used for analysis of the data. The following packages have been installed to perform analysis of the data.

```
install.packages("mlbench")
install.packages("psych")
install.packages("glmnet")
install.packages("tidyverse")
install.packages("lmtest")
install.packages("caret")
install.packages("Hmisc")
install.packages("corrplot")
install.packages("dplyr")
install.packages("PerformanceAnalytics")
library(mlbench)
library(psych)
library(glmnet)
library(tidyverse)
library(lmtest)
library(caret)
liblibrary(corrplot)
lirary(Hmisc)
library(dplyr)
library(PerformanceAnalytics)
```

mlbench stands for Machine Learning Benchmark Problems which is a collection of artificial and real-world machine learning benchmark problems.

glmnet is used for fitting Lasso and elastic-Net Regularized Generalized Linear Models.

tidyverse is a set of packages used for preparing, wrangling and visualizing data.

lmtest is a collection of tests, datasets and examples for diagnostic checking in Linear Regression Models.

caret stands for Classification And Regression Training. It is a set of functions that attempt to streamline the process for creating predictive models. The package contains tools for data splitting and pre-processing.

corrplot provides a visual exploratory tool on correlation matrix that supports automatic variable reordering to help detect hidden patterns among variables.

Hmisc is Harrell Miscellaneous that contains functions useful for data analysis, high-level graphics, utility operations and importing & annotating datasets.

dplyr is a Grammar of Data Manipulation.

PerformanceAnalytics is a collection of econometric functions for performance and risk analysis.

DATA DESCRIPTION

Importing the Dataset:

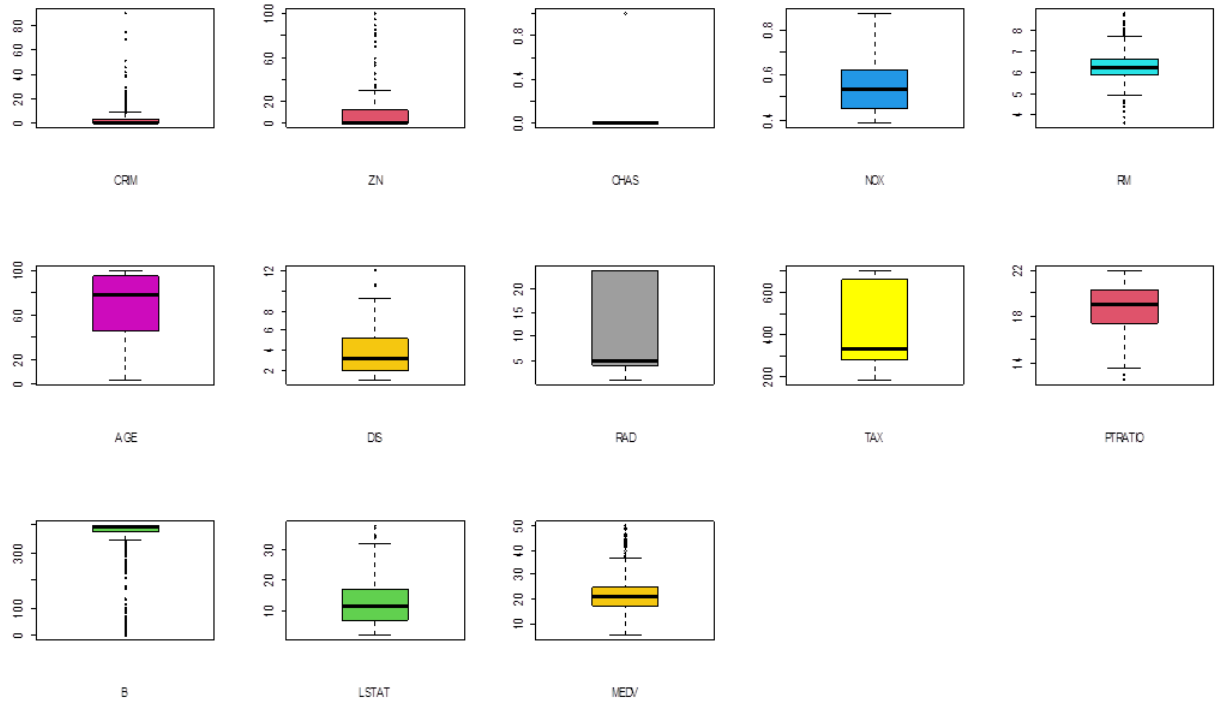
The dataset has been stored in the variable "*data*" and it has 506 observations & 14 variables.

```
> #IMPORTING THE DATASET
> data=read.csv("E:\\Pavi\\LOYOLA PG\\Applied Regression Analysis\\Project\\Boston House\\boston.csv")
> data=select(data,-c(1))
> head(data)
     CRIM ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO      B LSTAT
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90  4.98
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90  9.14
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83  4.03
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63  2.94
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12  5.21
  MEDV
1 24.0
2 21.6
3 34.7
4 33.4
5 36.2
6 28.7
> dim(data)
[1] 506  14
```

The dataset has been stored in the variable "*data*" and it has 506 observations & 14 variables.

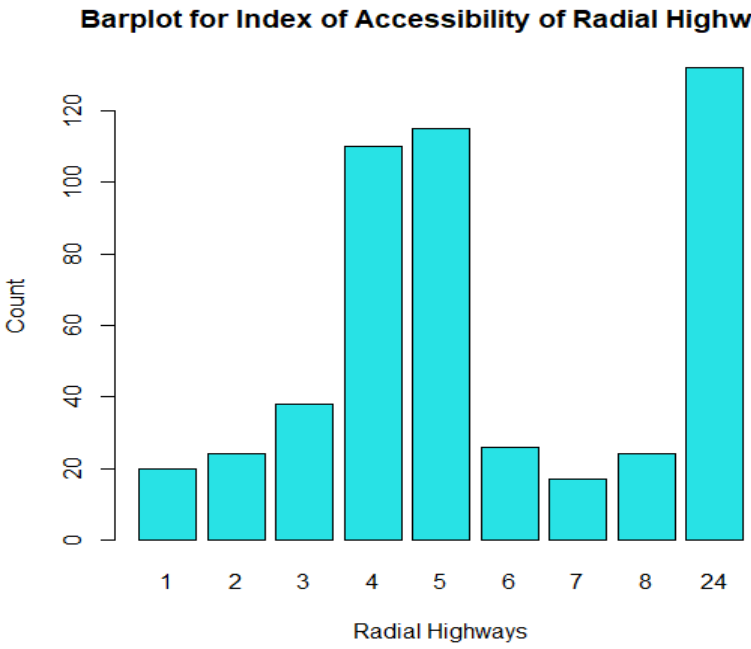
BOX-PLOT

```
> par(mfrow=c(3,5))
> boxplot(data$CRIM,col=50,xlab="CRIM")
> boxplot(data$ZN,col=2,xlab="ZN")
> boxplot(data$CHAS,col="BLUE",xlab="CHAS")
> boxplot(data$NOX,col=4,xlab="NOX")
> boxplot(data$RM,col=5,xlab="RM")
> boxplot(data$AGE,col=6,xlab="AGE")
> boxplot(data$DIS,col=7,xlab="DIS")
> boxplot(data$RAD,col=8,xlab="RAD")
> boxplot(data$TAX,col="YELLOW",xlab="TAX")
> boxplot(data$PTRATIO,col=10,xlab="PTRATIO")
> boxplot(data$B,col=11,xlab="B")
> boxplot(data$LSTAT,col=43,xlab="LSTAT")
> boxplot(data$MEDV,col=15,xlab="MEDV")
```



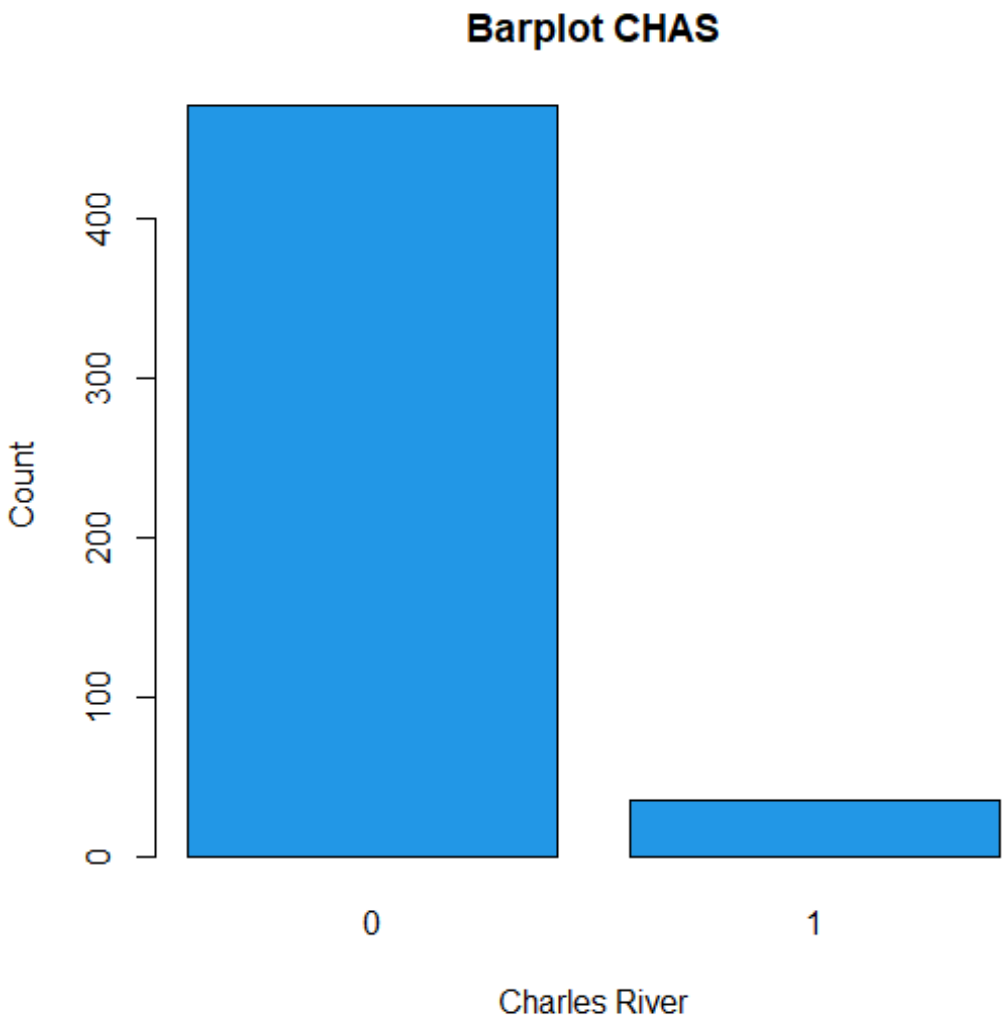
BARPLOT:

```
> #RAD
> barplot(table(data$RAD),col=5,xlab="Radial Highways",ylab="Count",main="Barplot for Index of Accessibility of Radial Highways")
```



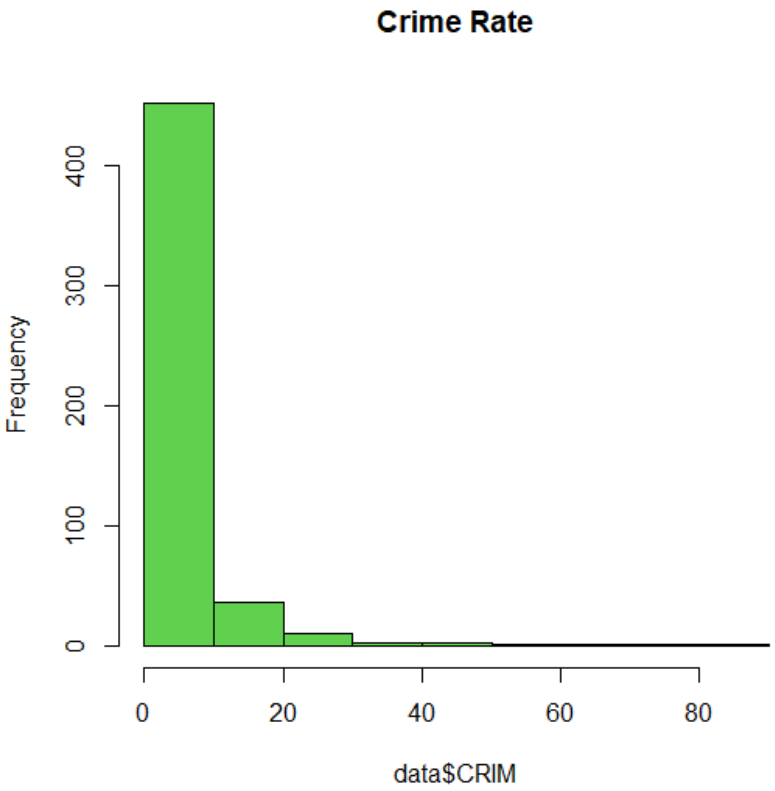
4,5,24 Index of Accessibility of Radial highways is high

```
> #CHAS
> barplot(table(data$CHAS),col=4,xlab="Charles River",ylab="Count",main="Barplot CHAS")
```



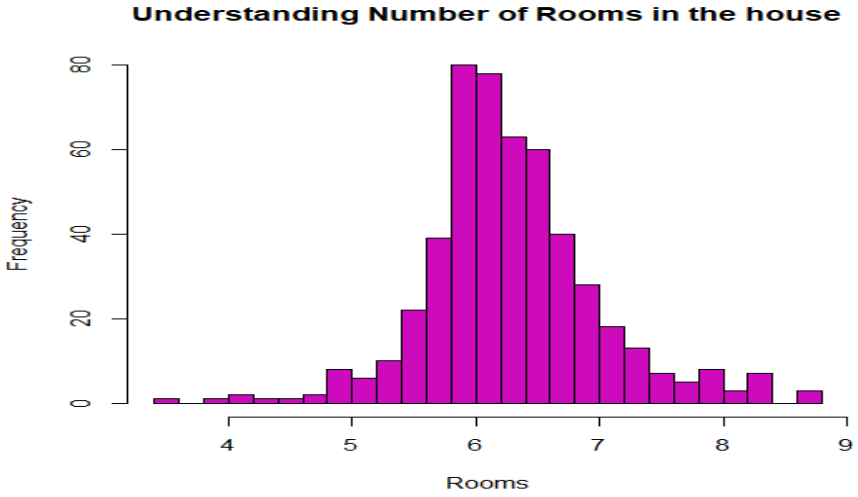
It can be seen that most of the houses don't have Charles River in the nearby locality.

```
> #CRIM
> hist(data$CRIM,breaks=10,col=3,main="Crime Rate")
```



Around 80% of the houses have only minimal crime rate ranging from 0 to 10% Remaining houses have crime rate between 10% and 30%. Very less number of houses have crime rate from 30% to 50%.

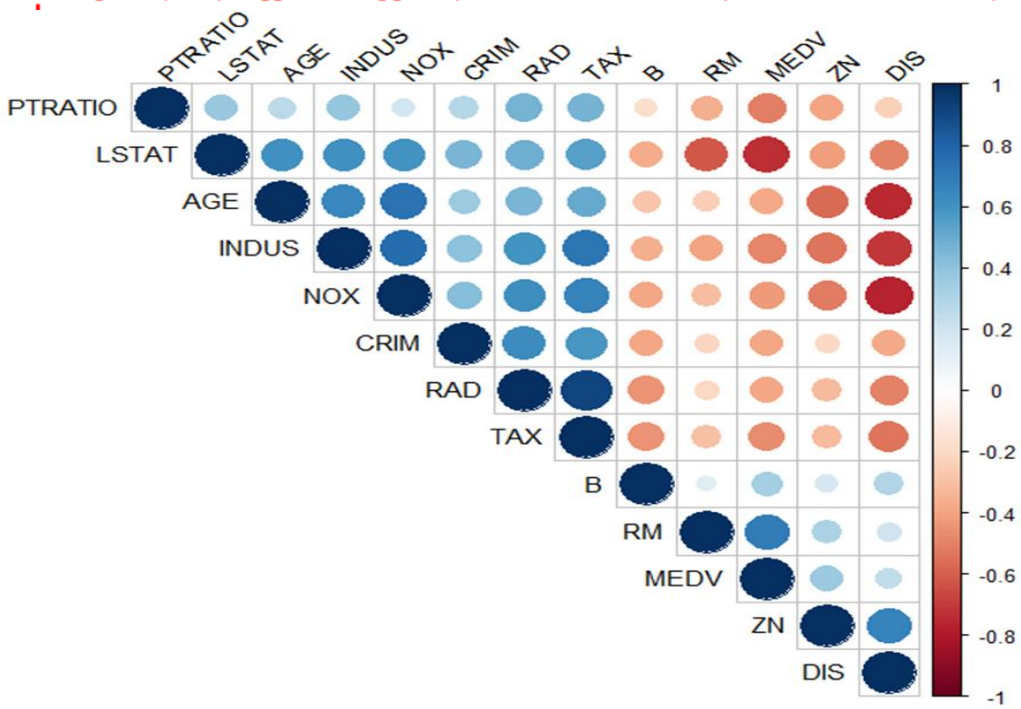
```
> #RM
> hist(data$RM,breaks=30,col=6,xlab="Rooms",main="Understanding Number of Rooms in the house")
```



There are six average numbers of rooms in the dwelling.

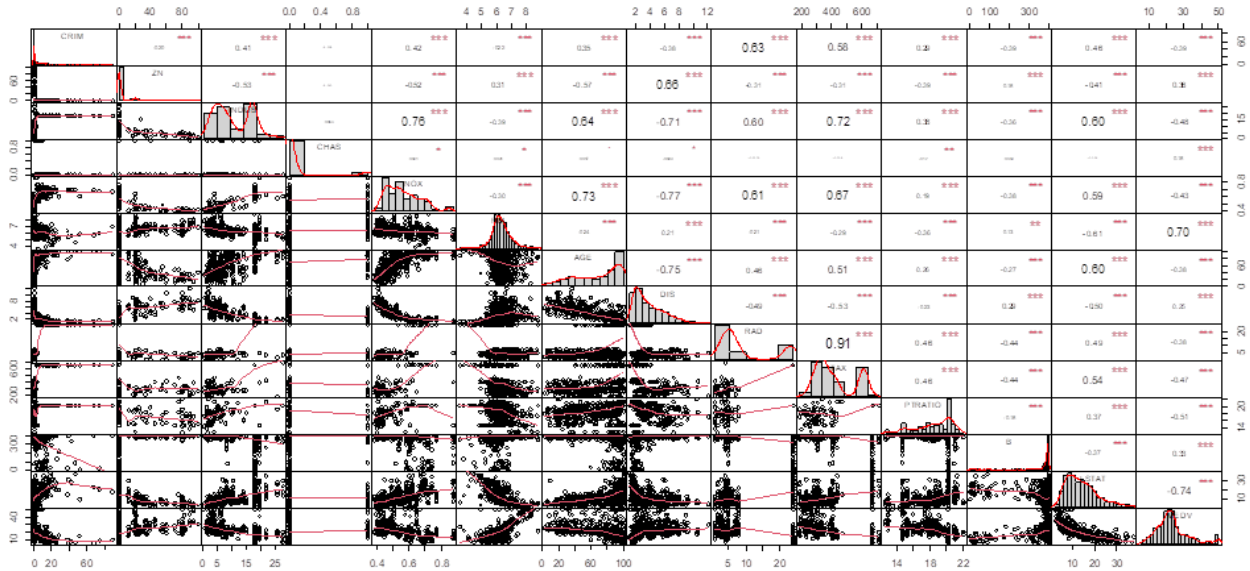
CORRELATION MATRIX PLOT:

```
> res=cor(data[,-c(4)],use = "complete.obs")     #Removing the Categorical Variable
> corrplot(res, type = "upper", order = "hclust", tl.col = "black", tl.srt = 45)
```



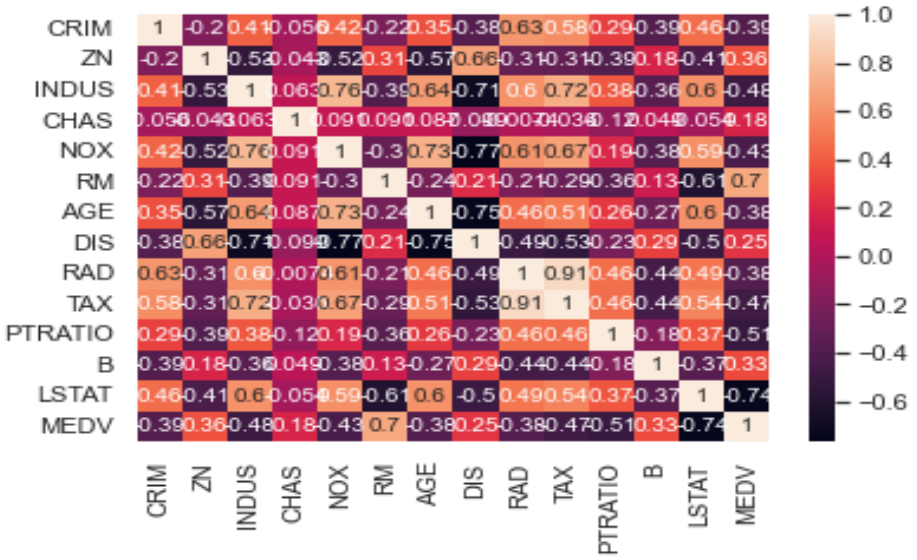
From the given correlation matrix plot, it can be seen that the correlation between TAX and INDUS, AGE and NOX is very high.

```
> chart.Correlation(data, histogram=TRUE, pch=19)
```



HEAT MAP:

```
In [10]: corrmatrix=df.corr()
         sns.heatmap(corrmatrix, annot= True)
         plt.show()
```



From correlation matrix, we see TAX and RAD are highly correlated features. The columns LSTAT, INDUS, RM, TAX, NOX, PTRAIO has a correlation score above 0.4 with MEDV which is a good indication of using them as predictors.

Splitting The Data Into 70% Train And 30% Test

```
> ### SPLITTING THE DATA INTO TRAIN AND TEST ###
>
> install.packages("caTools")
Warning: package 'caTools' is in use and will not be installed
> library(caTools)
> set.seed(25)
>
> sample_size=floor(0.7*nrow(data))
> train_ind=sample(seq_len(nrow(data)),size=sample_size)
> train=data[train_ind,]  #data[r,c]
> test=data[-train_ind,]
> dim(train)
[1] 354  14
> dim(test)
[1] 152  14
>
> #----------------------------------------
>
> x_train=train[,1:13]
> x_test=test[,1:13]
> y_train=train[,14]
> y_test=test[,14]
```

MODEL BUILDING

1. Multiple Linear Regression Model

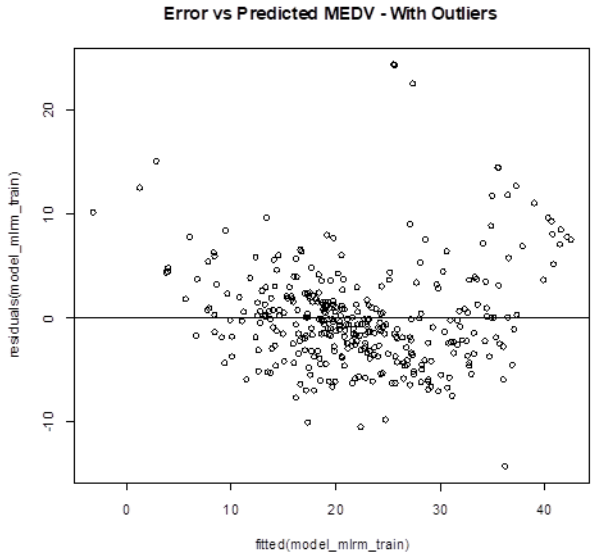
Fitting the Model for Multiple Linear Regression Model

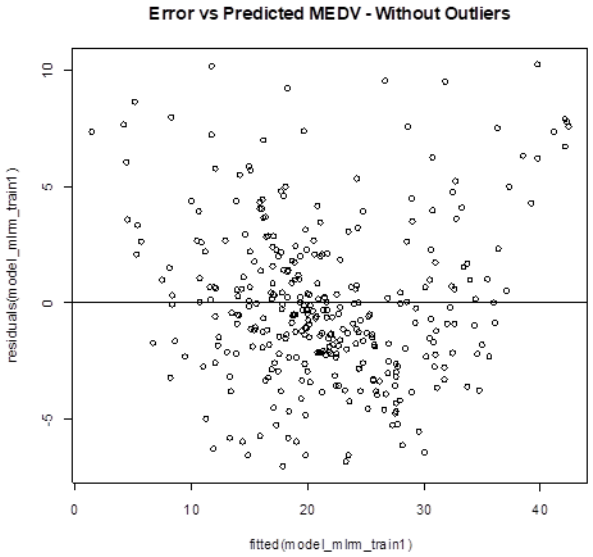
```
> #MLRM - TRAIN
> model_mlrm_train=lm(MEDV~.,train)
> summary(model_mlrm_train)

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

Residuals:
     Min       1Q   Median       3Q      Max
-14.2880  -2.9172  -0.6136   1.9000  24.4484

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  49.116291   6.724637   7.304 2.00e-12 ***
CRIM         -0.084604   0.045374  -1.865 0.063104 .
ZN            0.041550   0.017464   2.379 0.017906 *
INDUS        -0.014464   0.076302  -0.190 0.849769
CHAS          3.365177   1.114000   3.021 0.002712 **
NOX         -22.198683   4.726720  -4.696 3.85e-06 ***
RM            3.018403   0.541241   5.577 5.00e-08 ***
AGE           0.006187   0.017606   0.351 0.725497
DIS          -1.792542   0.263796  -6.795 4.86e-11 ***
RAD           0.288868   0.084057   3.437 0.000662 ***
TAX          -0.010468   0.004576  -2.288 0.022767 *
PTRATIO      -1.139338   0.160998  -7.077 8.48e-12 ***
B             0.007790   0.003283   2.373 0.018201 *
LSTAT        -0.587483   0.065727  -8.938  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.963 on 340 degrees of freedom
Multiple R-squared:  0.7388,    Adjusted R-squared:  0.7288
F-statistic: 73.97 on 13 and 340 DF,  p-value: < 2.2e-16
```

Fitting the Model for Multiple Linear Regression Model after removing outliers is given below.

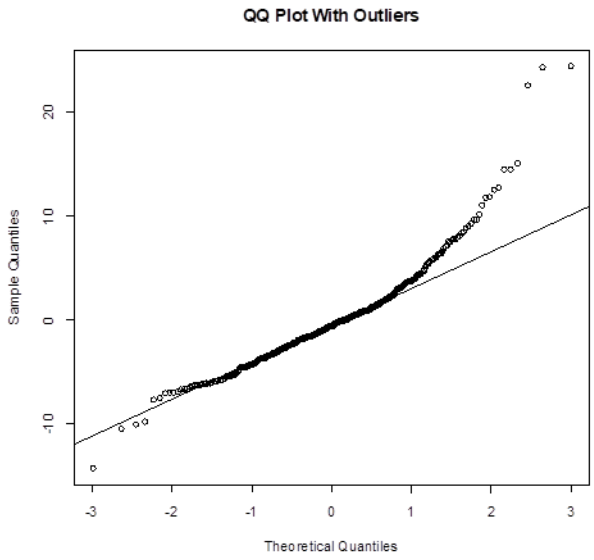
```
> #Removal of Outliers
> data3=subset(train,abs(rstandard(model_mlrm_train))<1.96)
> model_mlrm_train1=lm(MEDV~.,data3)
> summary(model_mlrm_train1)

Call:
lm(formula = MEDV ~ ., data = data3)

Residuals:
    Min      1Q  Median      3Q     Max
-7.0019 -2.1952 -0.4626  1.8663 10.3450

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 19.343442   5.374793   3.599  0.00037 ***
CRIM        -0.082004   0.032778  -2.502  0.01285 *
ZN           0.027600   0.012547   2.200  0.02853 *
INDUS       -0.012704   0.053889  -0.236  0.81378
CHAS         2.453678   0.826930   2.967  0.00323 **
NOX         -8.500007   3.436858  -2.473  0.01391 *
RM           5.237377   0.461605  11.346  < 2e-16 ***
AGE         -0.026688   0.012933  -2.064  0.03986 *
DIS         -1.210417   0.188637  -6.417 4.98e-10 ***
RAD          0.139369   0.059683   2.335  0.02015 *
TAX         -0.010207   0.003203  -3.186  0.00158 **
PTRATIO     -0.868217   0.114898  -7.556 4.34e-13 ***
B            0.011548   0.002401   4.810 2.33e-06 ***
LSTAT       -0.360247   0.052077  -6.918 2.48e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.462 on 322 degrees of freedom
Multiple R-squared:  0.8374,    Adjusted R-squared:  0.8309
F-statistic: 127.6 on 13 and 322 DF,  p-value: < 2.2e-16
```

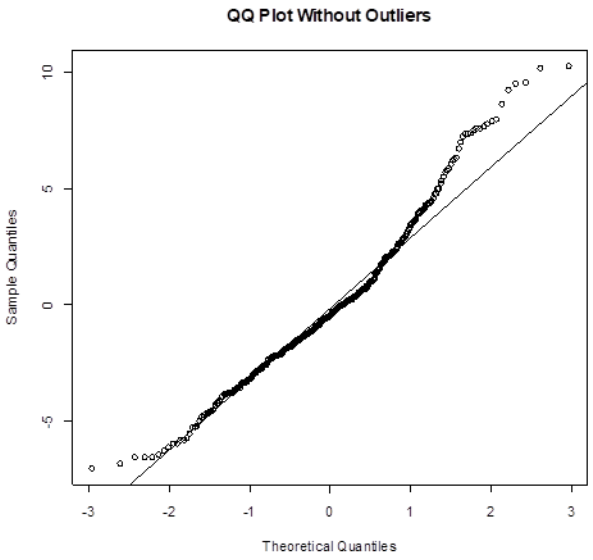
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:



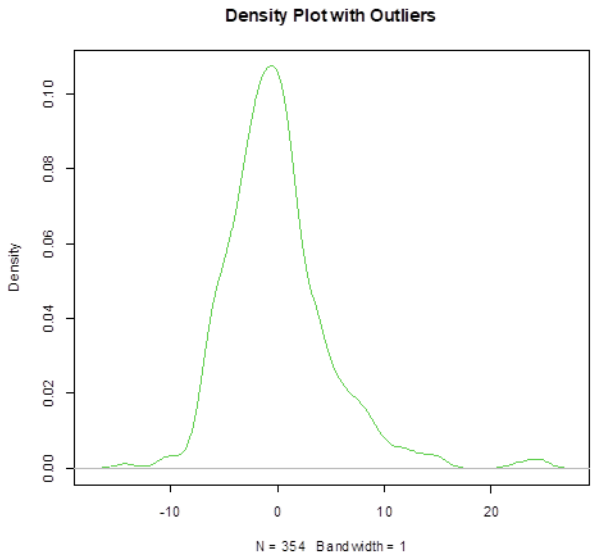


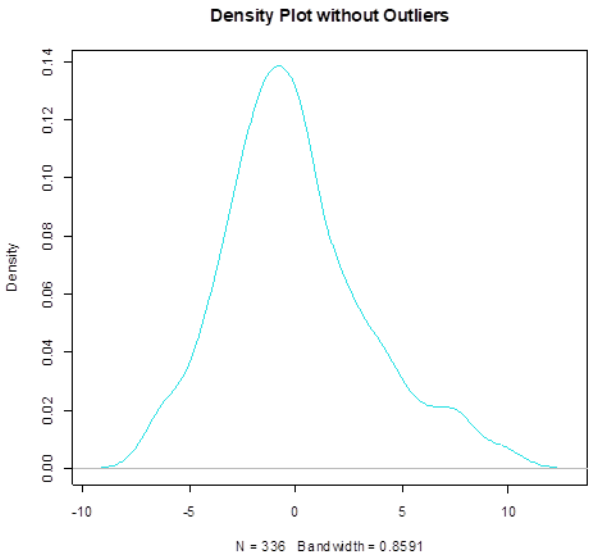
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.





The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.





The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

Hence, it can be concluded that the Multiple Linear Regression Model without outliers is the best fit without any violation of Assumptions.

2. Multiple Linear Regression Model – Stepwise Method

```
> #MLRM - Step Wise MLRM
> model_mlrm_step=step(lm(MEDV~.,train))
Start:  AIC=1147.94
MEDV ~ CRIM + ZN + INDUS + CHAS + NOX + RM + AGE + DIS + RAD +
    TAX + PTRATIO + B + LSTAT

          Df Sum of Sq     RSS    AIC
- INDUS    1      0.89  8375.5 1146.0
- AGE      1      3.04  8377.6 1146.1
<none>                  8374.6 1147.9
- CRIM     1     85.63  8460.2 1149.5
- TAX      1    128.91  8503.5 1151.3
- B        1    138.70  8513.3 1151.8
- ZN       1    139.42  8514.0 1151.8
- CHAS     1    224.77  8599.4 1155.3
- RAD      1    290.90  8665.5 1158.0
- NOX      1    543.27  8917.9 1168.2
- RM       1    766.05  9140.7 1176.9
- DIS      1   1137.33  9511.9 1191.0
- PTRATIO  1   1233.54  9608.1 1194.6
- LSTAT    1   1967.85 10342.4 1220.7

Step:  AIC=1145.97
MEDV ~ CRIM + ZN + CHAS + NOX + RM + AGE + DIS + RAD + TAX +
    PTRATIO + B + LSTAT
```

```
Step:  AIC=1145.97
MEDV ~ CRIM + ZN + CHAS + NOX + RM + AGE + DIS + RAD + TAX +
    PTRATIO + B + LSTAT

          Df Sum of Sq     RSS    AIC
- AGE      1      3.02  8378.5 1144.1
<none>                  8375.5 1146.0
- CRIM     1     84.91  8460.4 1147.5
- B        1    140.88  8516.4 1149.9
- ZN       1    141.39  8516.9 1149.9
- TAX      1    166.55  8542.0 1150.9
- CHAS     1    223.97  8599.5 1153.3
- RAD      1    324.59  8700.1 1157.4
- NOX      1    590.67  8966.2 1168.1
- RM       1    778.52  9154.0 1175.4
- DIS      1   1203.39  9578.9 1191.5
- PTRATIO  1   1265.48  9641.0 1193.8
- LSTAT    1   1995.73 10371.2 1219.6

Step:  AIC=1144.1
MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO +
    B + LSTAT

          Df Sum of Sq     RSS    AIC
<none>                  8378.5 1144.1
- CRIM     1     84.72  8463.2 1145.7
- ZN       1    138.70  8517.2 1147.9
- B        1    143.85  8522.4 1148.1
- TAX      1    164.81  8543.3 1149.0
- CHAS     1    228.17  8606.7 1151.6
- RAD      1    321.60  8700.1 1155.4
- NOX      1    611.07  8989.6 1167.0
- RM       1    838.50  9217.0 1175.9
- PTRATIO  1   1266.67  9645.2 1191.9
- DIS      1   1358.45  9737.0 1195.3
- LSTAT    1   2207.12 10585.6 1224.9
```

```
> summary(model_mlrm_step)

Call:
lm(formula = MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS + RAD +
    TAX + PTRATIO + B + LSTAT, data = train)

Residuals:
     Min       1Q   Median       3Q      Max
-14.4038  -2.9849  -0.5843   1.9675  24.5892

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  48.960444   6.682903   7.326 1.72e-12 ***
CRIM         -0.084008   0.045175  -1.860 0.063801 .
ZN            0.040581   0.017055   2.379 0.017889 *
CHAS          3.372354   1.105019   3.052 0.002452 **
NOX         -21.999087   4.404839  -4.994 9.43e-07 ***
RM            3.069021   0.524587   5.850 1.15e-08 ***
DIS          -1.805816   0.242505  -7.447 7.87e-13 ***
RAD           0.290276   0.080117   3.623 0.000335 ***
TAX          -0.010759   0.004148  -2.594 0.009903 **
PTRATIO      -1.137531   0.158198  -7.191 4.09e-12 ***
B             0.007901   0.003260   2.423 0.015902 *
LSTAT        -0.580792   0.061190  -9.492  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.95 on 342 degrees of freedom
Multiple R-squared:  0.7387,    Adjusted R-squared:  0.7303
F-statistic: 87.88 on 11 and 342 DF,  p-value: < 2.2e-16
```

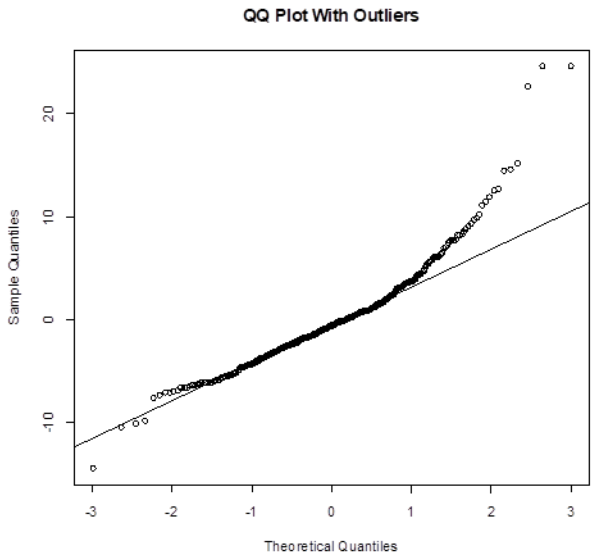
Fitting the Model for Multiple Linear Regression Model - Stepwise after removing outliers

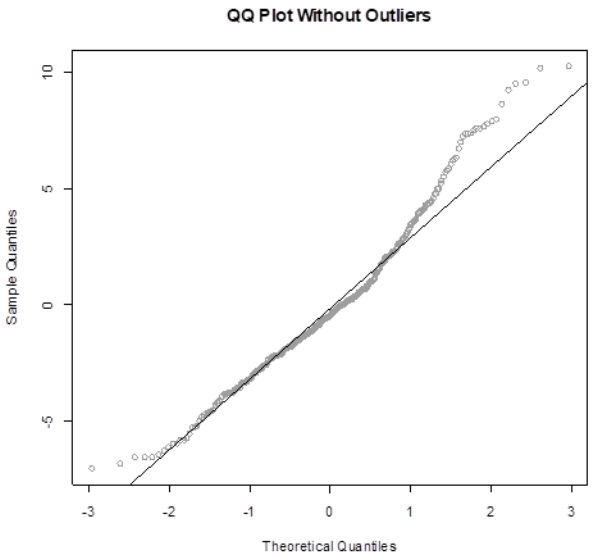
```
> #Removal of Outliers
> data4=subset(train,abs(rstandard(model_mlrm_step))<1.96)
> model_mlrm_step1=step(lm(MEDV~.,data4))
Start:  AIC=848.24
MEDV ~ CRIM + ZN + INDUS + CHAS + NOX + RM + AGE + DIS + RAD +
    TAX + PTRATIO + B + LSTAT

          Df Sum of Sq    RSS    AIC
- INDUS    1      0.67 3860.2 846.30
<none>                 3859.5 848.24
- AGE      1     51.04 3910.6 850.65
- ZN       1     58.00 3917.5 851.25
- RAD      1     65.36 3924.9 851.88
- NOX      1     73.31 3932.8 852.56
- CRIM     1     75.02 3934.5 852.71
- CHAS     1    105.53 3965.0 855.30
- TAX      1    121.70 3981.2 856.67
- B        1    277.26 4136.8 869.55
- DIS      1    493.51 4353.0 886.67
- LSTAT    1    573.58 4433.1 892.79
- PTRATIO  1    684.39 4543.9 901.09
- RM       1   1542.99 5402.5 959.24

Step:  AIC=846.3
MEDV ~ CRIM + ZN + CHAS + NOX + RM + AGE + DIS + RAD + TAX +
    PTRATIO + B + LSTAT
Step:  AIC=846.3
MEDV ~ CRIM + ZN + CHAS + NOX + RM + AGE + DIS + RAD + TAX +
    PTRATIO + B + LSTAT

          Df Sum of Sq    RSS    AIC
<none>                 3860.2 846.30
- AGE      1     50.98 3911.2 848.70
- ZN       1     58.83 3919.0 849.38
- RAD      1     74.20 3934.4 850.69
- CRIM     1     74.49 3934.7 850.72
- NOX      1     81.03 3941.2 851.28
- CHAS     1    104.86 3965.0 853.30
- TAX      1    153.85 4014.0 857.43
- B        1    280.99 4141.2 867.90
- DIS      1    520.48 4380.7 886.80
- LSTAT    1    585.80 4446.0 891.77
- PTRATIO  1    703.25 4563.4 900.53
- RM       1   1562.51 5422.7 958.50
```

```
> summary(model_mlrm_step1)

Call:
lm(formula = MEDV ~ CRIM + ZN + CHAS + NOX + RM + AGE + DIS +
    RAD + TAX + PTRATIO + B + LSTAT, data = data4)

Residuals:
    Min      1Q  Median      3Q     Max
-7.0106 -2.1951 -0.4699  1.8991 10.2516

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 19.355759   5.366676   3.607 0.000359 ***
CRIM        -0.081602   0.032686  -2.497 0.013039 *
ZN           0.027758   0.012511   2.219 0.027198 *
CHAS         2.438349   0.823164   2.962 0.003281 **
NOX         -8.689148   3.337006  -2.604 0.009643 **
RM           5.247507   0.458928  11.434  < 2e-16 ***
AGE         -0.026670   0.012913  -2.065 0.039692 *
DIS         -1.198635   0.181630  -6.599 1.69e-10 ***
RAD          0.143121   0.057438   2.492 0.013212 *
TAX         -0.010510   0.002929  -3.588 0.000385 ***
PTRATIO     -0.871905   0.113662  -7.671 2.03e-13 ***
B            0.011592   0.002391   4.849 1.93e-06 ***
LSTAT       -0.361656   0.051657  -7.001 1.48e-11 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.457 on 323 degrees of freedom
Multiple R-squared:  0.8374,    Adjusted R-squared:  0.8314
F-statistic: 138.6 on 12 and 323 DF,  p-value: < 2.2e-16
```

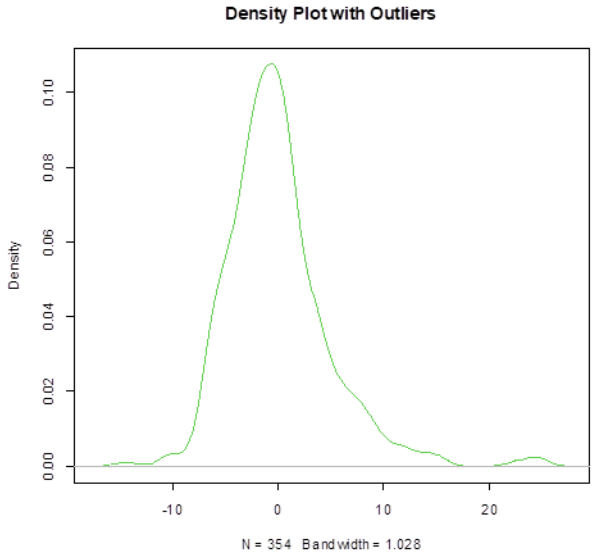
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

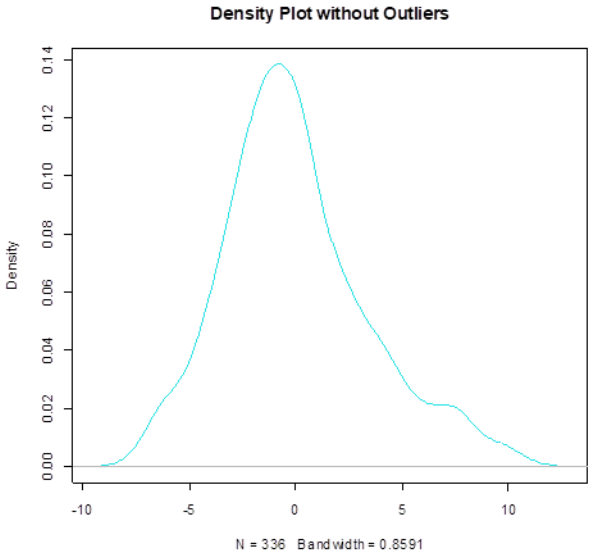
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.





The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.



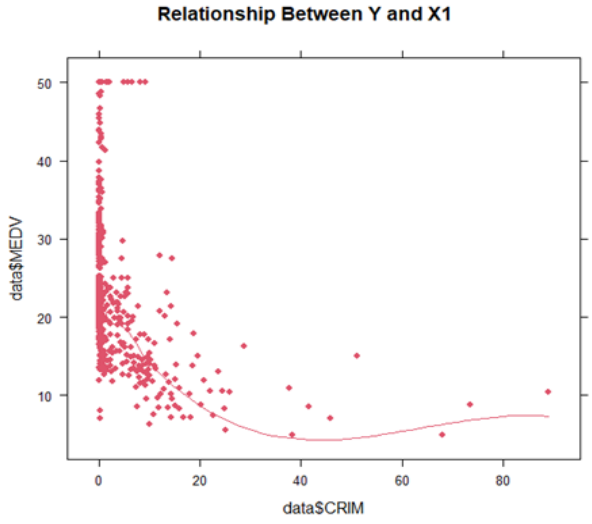


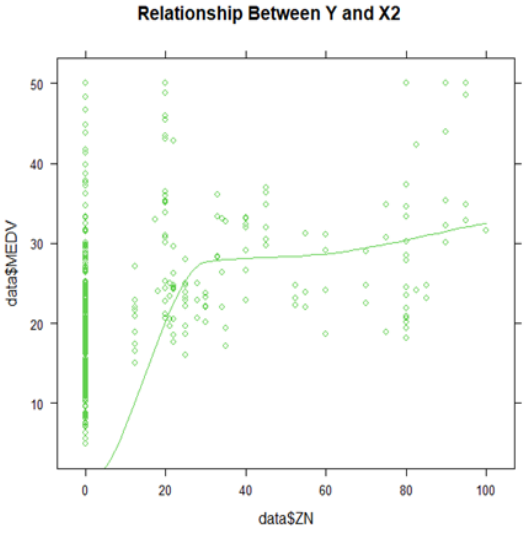
The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

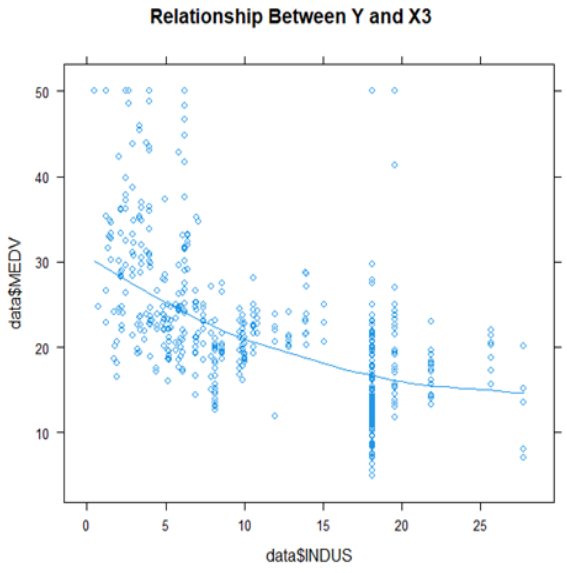
Hence, it can be concluded that the Multiple Linear Regression Model – Step wise without outliers is the best fit when compared to MLRM – Step Wise (with outliers).

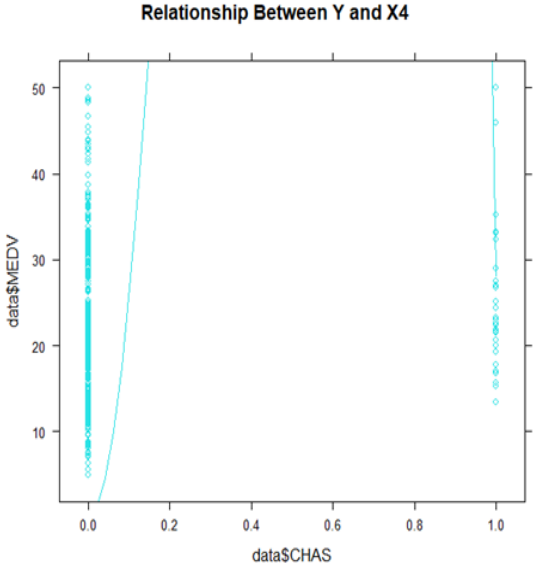
CHECKING THE RELATIONSHIP BETWEEN STUDY VARIABLE AND EXPLANATORY VARIABLES:

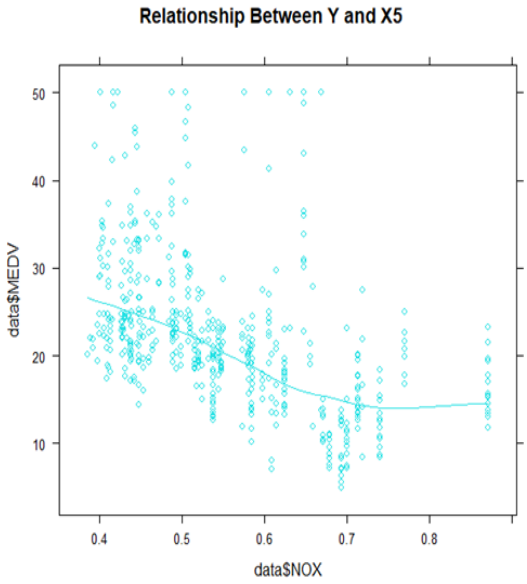
```
> #LINEARITY BETWEEN STUDY VARIABLE AND EXPLANATORY VARIABLES
> xyplot(data$MEDV~data$CRIM,type=c("p","smooth"),pch=16,col=2,main="Relationship Between Y and X1")
> xyplot(data$MEDV~data$ZN,type=c("p","smooth"),col=3,main="Relationship Between Y and X2")
There were 35 warnings (use warnings() to see them)
> xyplot(data$MEDV~data$INDUS,type=c("p","smooth"),col=4,main="Relationship Between Y and X3")
> xyplot(data$MEDV~data$CHAS,type=c("p","smooth"),col=5,main="Relationship Between Y and X4")
There were 40 warnings (use warnings() to see them)
> xyplot(data$MEDV~data$NOX,type=c("p","smooth"),col=5,main="Relationship Between Y and X5")
> xyplot(data$MEDV~data$RM,type=c("p","smooth"),col=6,main="Relationship Between Y and X6")
> xyplot(data$MEDV~data$AGE,type=c("p","smooth"),col=7,main="Relationship Between Y and X7")
> xyplot(data$MEDV~data$DIS,type=c("p","smooth"),col=2,main="Relationship Between Y and X8")
> xyplot(data$MEDV~data$RAD,type=c("p","smooth"),col=24,main="Relationship Between Y and X9")
> xyplot(data$MEDV~data$TAX,type=c("p","smooth"),col=22,main="Relationship Between Y and X10")
> xyplot(data$MEDV~data$PTRATIO,type=c("p","smooth"),col=12,main="Relationship Between Y and X11")
> xyplot(data$MEDV~data$B,type=c("p","smooth"),col=35,main="Relationship Between Y and X12")
> xyplot(data$MEDV~data$LSTAT,type=c("p","smooth"),col=28,main="Relationship Between Y and X13")
```

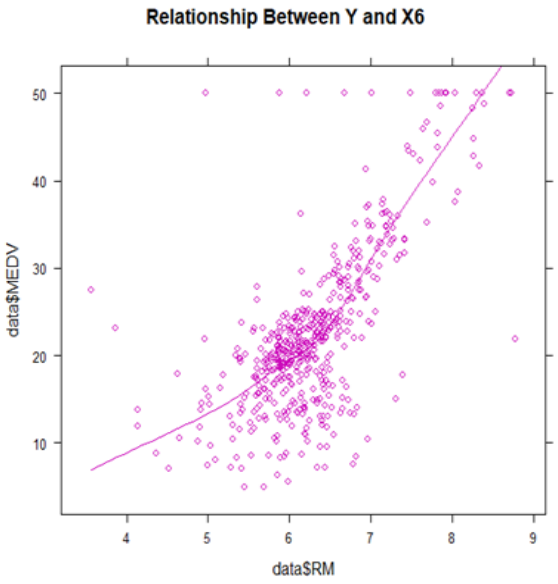


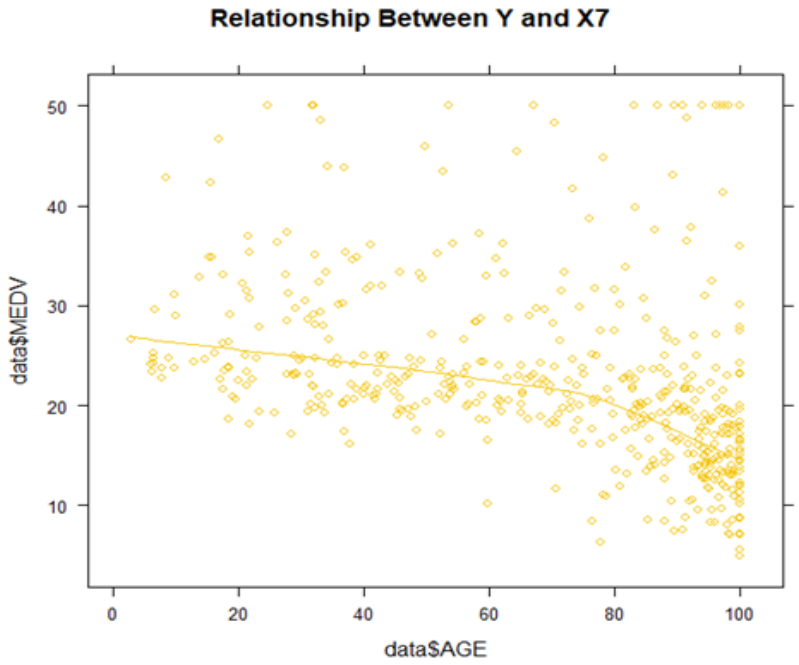


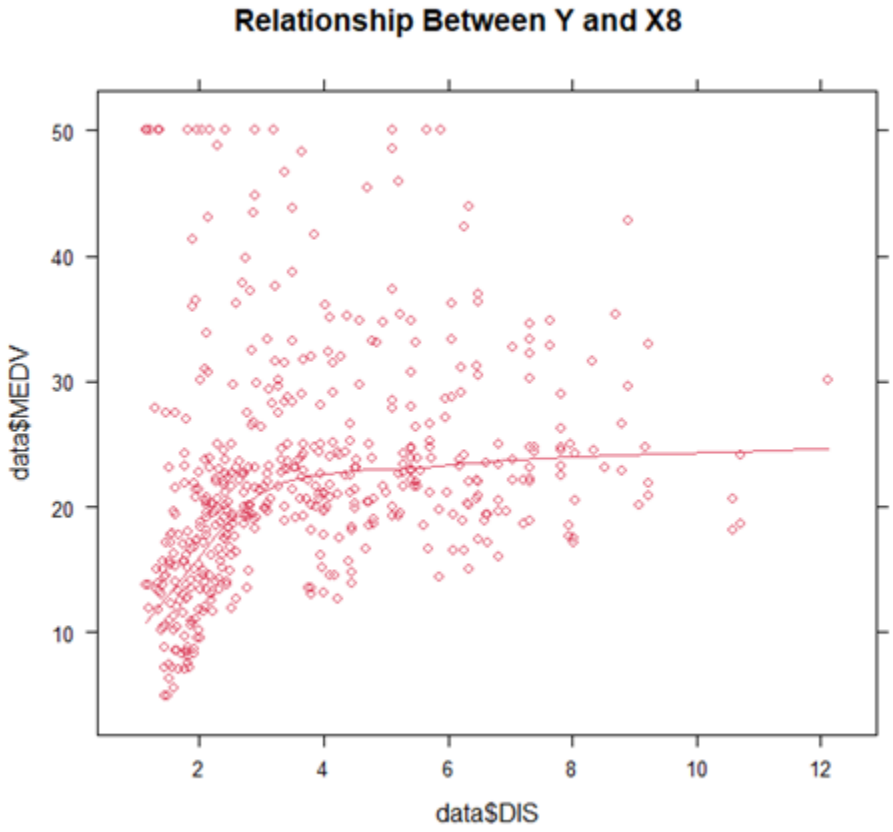


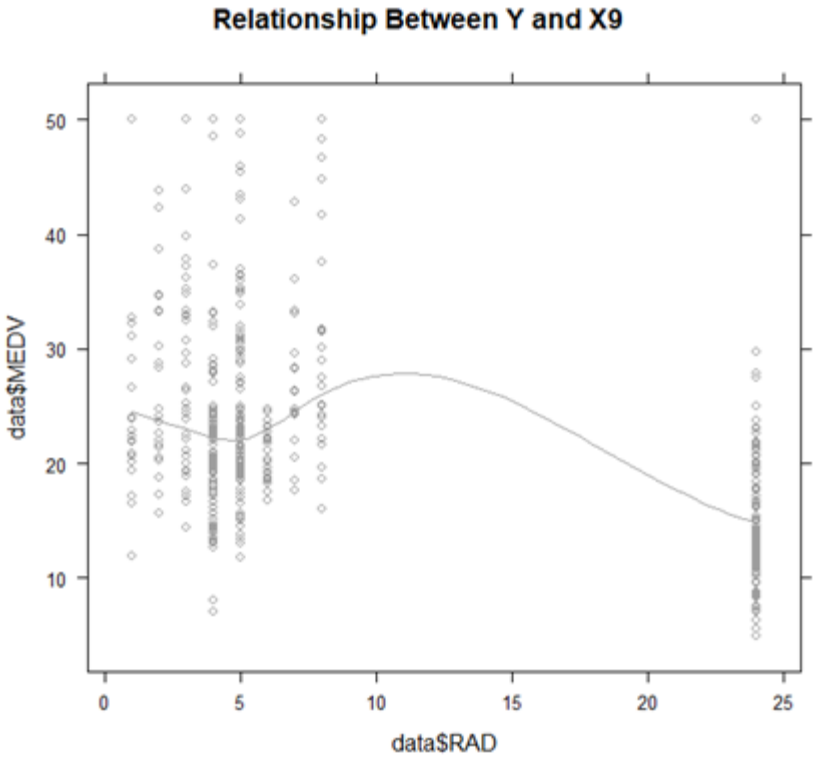


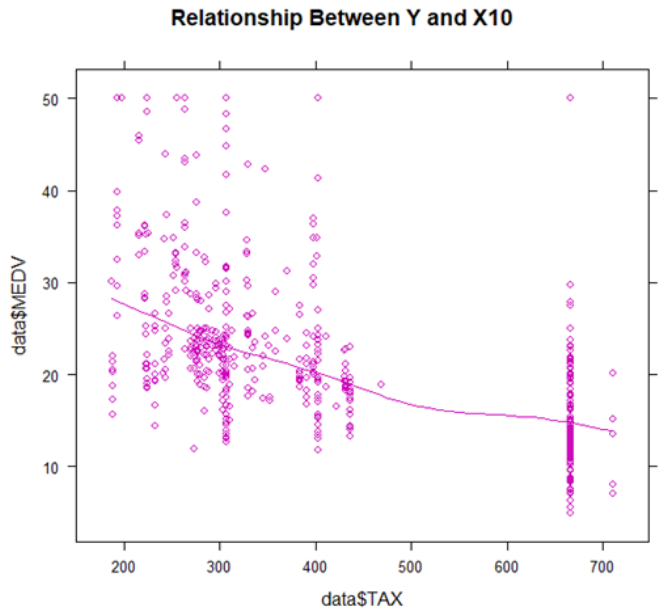

Relationship Between Y and X5
data$MEDV
data$NOX
50
40
30
20
10
0.4
0.5
0.6
0.7
0.8

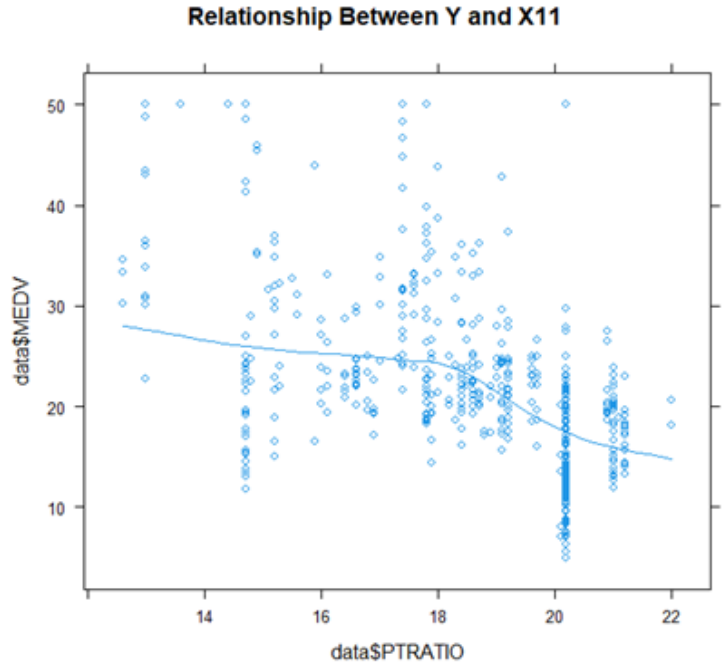

Relationship Between Y and X6
data$MEDV
data$RM
50
40
30
20
10
4
5
6
7
8
9

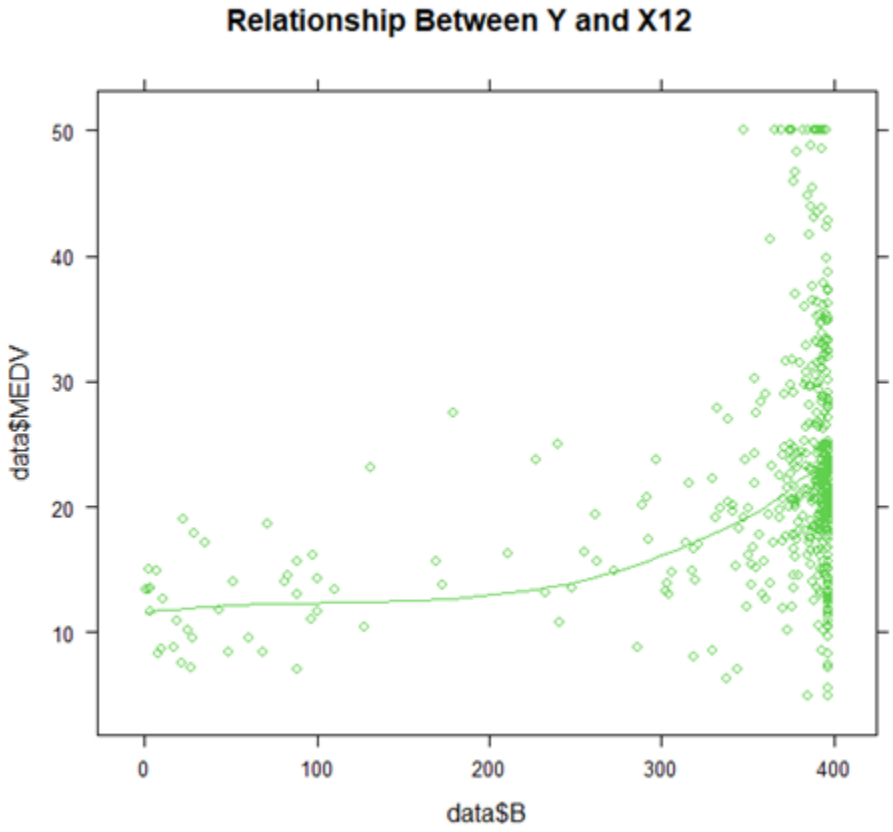

Relationship Between Y and X7
data$MEDV
data$AGE
50
40
30
20
10
0
20
40
60
80
100

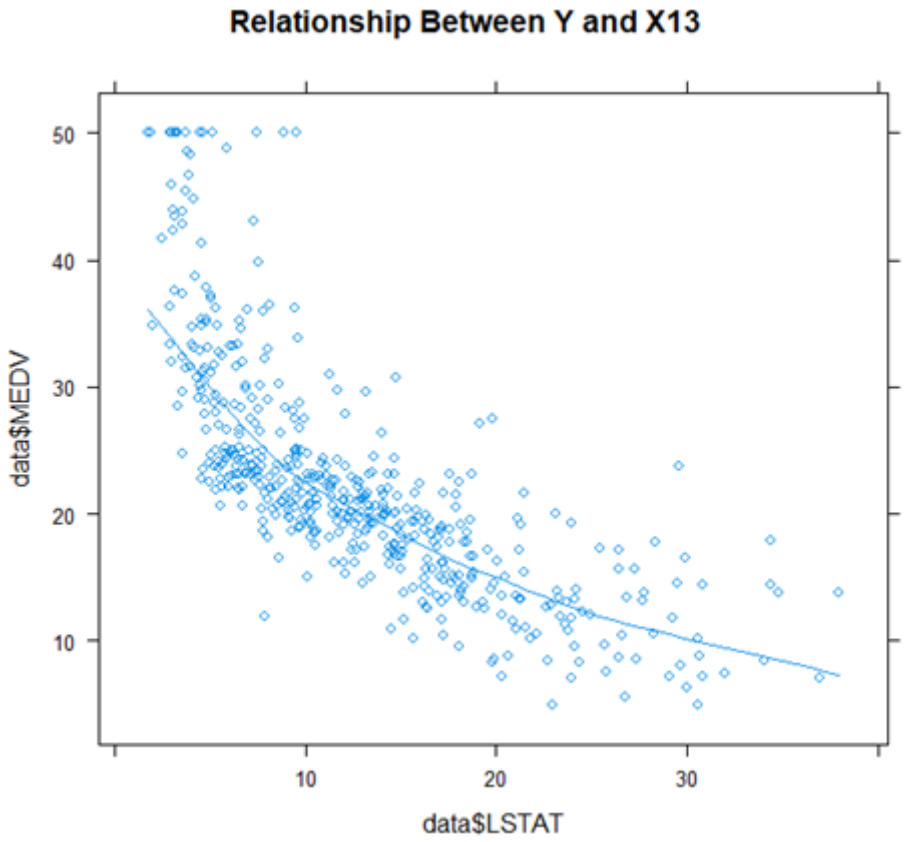

Relationship Between Y and X8
data$MEDV
data$DIS
50
40
30
20
10
2
4
6
8
10
12


Relationship Between Y and X9
data$MEDV
data$RAD
50
40
30
20
10
0
5
10
15
20
25


Relationship Between Y and X10
data$MEDV
data$TAX
50
40
30
20
10
200
300
400
500
600
700


Relationship Between Y and X11
data$MEDV
data$PTRATIO
50
40
30
20
10
14
16
18
20
22





It can be seen that the relationship between all the explanatory variables X1, X2,….X13 and Y (MEDV) is curvilinear. Hence, we fit the Polynomial Model for the given data.

3. Polynomial Regression Model

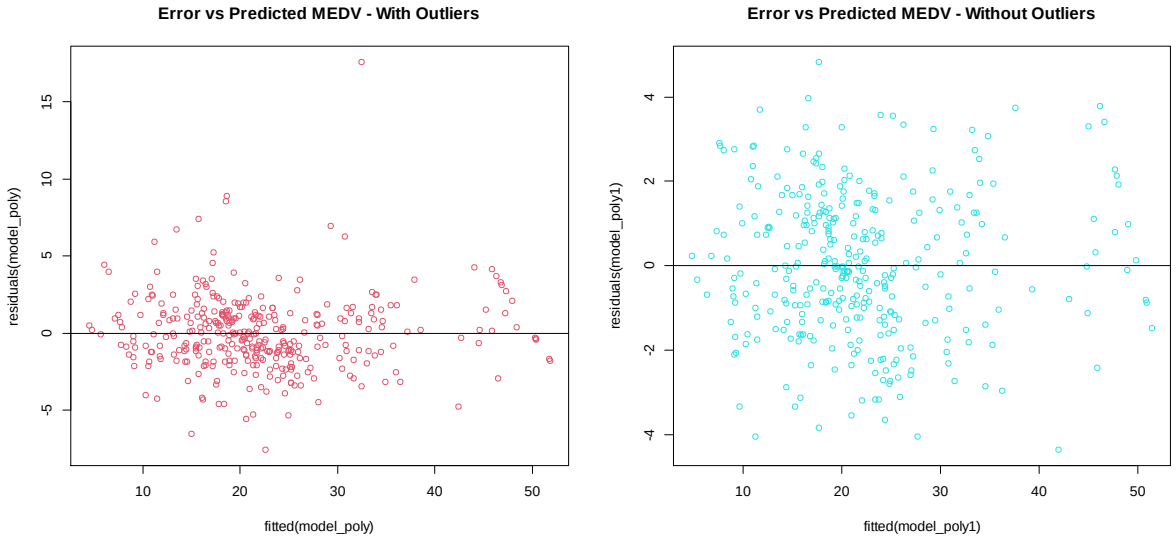
```
> #POLYNOMIAL WITH 2nd DEGREE
> model_poly=lm(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),data=train)
> summary(model_poly)

Call:
lm(formula = MEDV ~ poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE,
    DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE),
    data = train)

Residuals:
    Min      1Q  Median      3Q     Max
-7.5503 -1.3915 -0.0869  1.2061 17.5849

Coefficients: (1 not defined because of singularities)
                                                                                                                       Estimate
(Intercept)                                                                                                           1.965e-01
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)1.0.0.0.0.0.0.0.0.0.0.0.0 -4.490e+00
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)2.0.0.0.0.0.0.0.0.0.0.0.0  3.275e-03
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.1.0.0.0.0.0.0.0.0.0.0.0  2.933e-01
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)1.1.0.0.0.0.0.0.0.0.0.0.0  2.844e-01
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.1.0.0.1
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.0.1.0.1
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.0.0.1.1 .
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.0.0.0.2 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.899 on 250 degrees of freedom
Multiple R-squared:  0.9345,	Adjusted R-squared:  0.9075
F-statistic: 34.62 on 103 and 250 DF,  p-value: < 2.2e-16
```

Fitting the Model for Polynomial Model after removing outliers

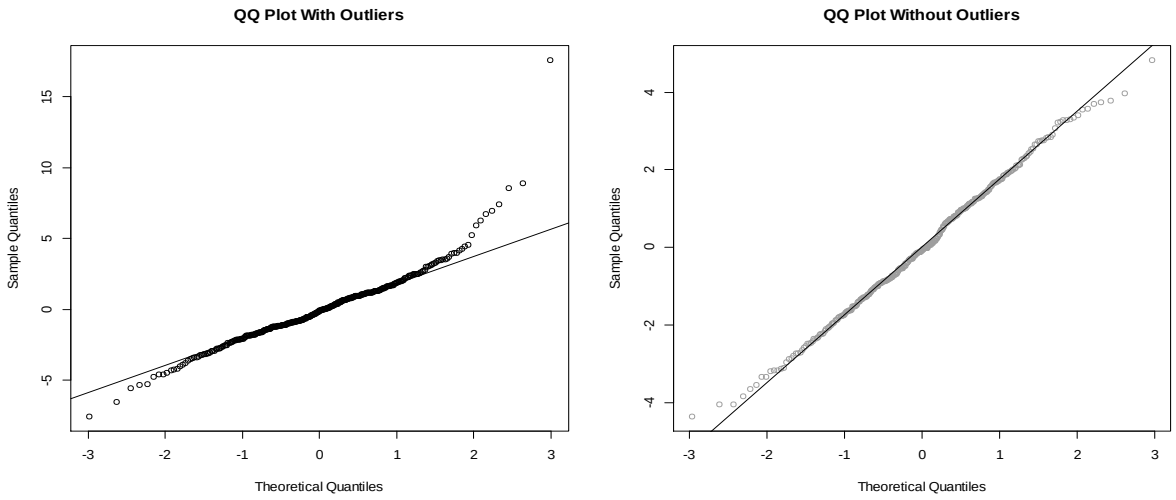
```
> #Removal of Outliers
> data5=subset(train,abs(rstandard(model_poly))<1.96)
> model_poly1=lm(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),data=data5)
> summary(model_poly1)

Call:
lm(formula = MEDV ~ poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE,
    DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE),
    data = data5)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3644 -1.1631 -0.0747  1.1947  4.8303

Coefficients: (1 not defined because of singularities)
                                                                                                                       Estimate
(Intercept)                                                                                                           1.423e+02
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)1.0.0.0.0.0.0.0.0.0.0.0.0 -7.056e+00
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)2.0.0.0.0.0.0.0.0.0.0.0.0  1.879e-03
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.1.0.0.0.0.0.0.0.0.0.0.0  1.134e-02
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)1.1.0.0.0.0.0.0.0.0.0.0.0  2.770e-01
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.2.0.0.0.0.0.0.0.0.0.0.0  2.378e-04
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.1.0.0.0.1 .
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.1.0.0.1
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.0.1.0.1 **
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.0.0.1.1 **
poly(CRIM, ZN, INDUS, CHAS, NOX, RM, AGE, DIS, RAD, TAX, PTRATIO, B, LSTAT, degree = 2, raw = TRUE)0.0.0.0.0.0.0.0.0.0.0.0.2 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.051 on 233 degrees of freedom
Multiple R-squared:  0.9668,	Adjusted R-squared:  0.9521
F-statistic: 65.82 on 103 and 233 DF,  p-value: < 2.2e-16
```

The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

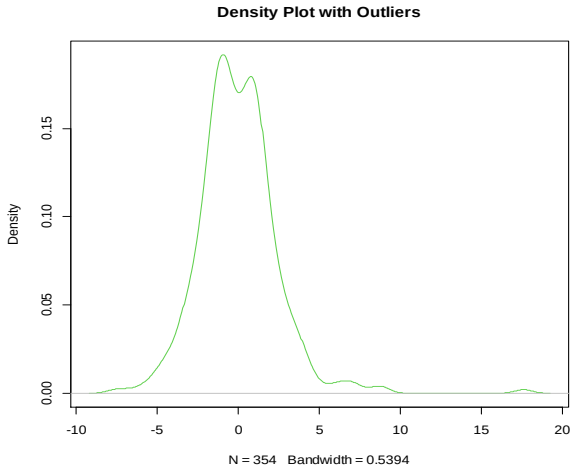
```
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted - OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(polyridge),residuals(polyridge),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(polyridgel),residuals(polyridgel),col=5,main="Error vs Predicted MEDV - Without Outliers")
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(polyridge),col=9,main="QQ Plot With Outliers")
> qqline(residuals(polyridge))
>
> qqnorm(residuals(polyridgel),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(polyridgel))
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(polyridge)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(polyridgel)),main="Density Plot without Outliers",col=5)
>
```

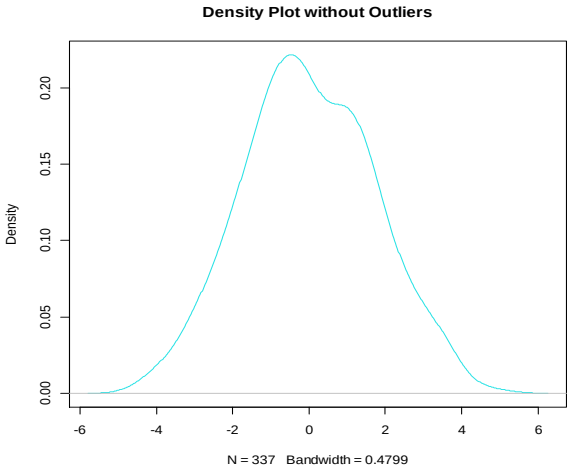


It can be seen that the Residual Plot without outliers follows a horizontal band fashion.

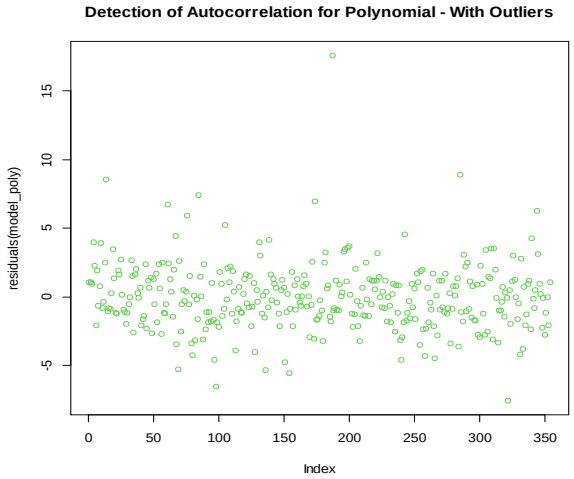


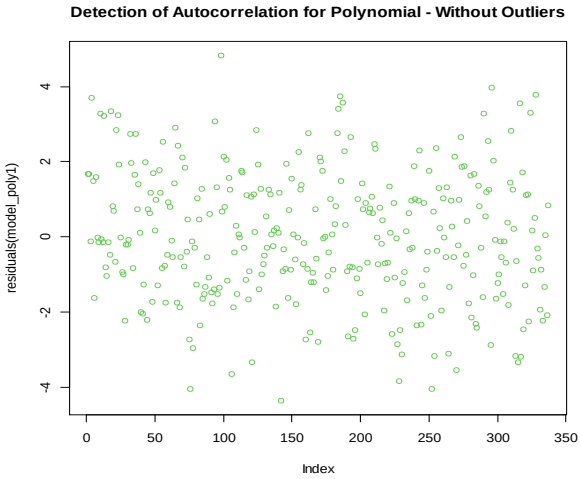
The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.





The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.





Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

Hence, it can be concluded that the Polynomial Ridge Regression Model without outliers is the best fit when compared to Poly Ridge Regression Model (with outliers).

4. Polynomial Ridge Model

```
> #RIDGE USING POLYNOMIAL 2nd DEGREE
>
> #custom control parameters
>
> custom<- trainControl(method = "repeatedcv",number=10,repeats=5,verboseIter= T)
> #ridge regression
> set.seed(123)
>
> #MODEL BUILDING
> polyridge<-train(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),
+                  data=train,method='glmnet',
+                  tunegrid=expand.grid(alpha=0,lambda=seq(0.0001,1,length=5)),
+                   trControl=custom)
+ Fold01.Rep1: alpha=0.10, lambda=1.438
- Fold01.Rep1: alpha=0.10, lambda=1.438
+ Fold01.Rep1: alpha=0.55, lambda=1.438
- Fold01.Rep1: alpha=0.55, lambda=1.438
+ Fold01.Rep1: alpha=1.00, lambda=1.438
- Fold01.Rep1: alpha=1.00, lambda=1.438
+ Fold02.Rep1: alpha=0.10, lambda=1.438
- Fold09.Rep5: alpha=1.00, lambda=1.438
+ Fold10.Rep5: alpha=0.10, lambda=1.438
- Fold10.Rep5: alpha=0.10, lambda=1.438
+ Fold10.Rep5: alpha=0.55, lambda=1.438
- Fold10.Rep5: alpha=0.55, lambda=1.438
+ Fold10.Rep5: alpha=1.00, lambda=1.438
- Fold10.Rep5: alpha=1.00, lambda=1.438
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.0144 on full training set
```

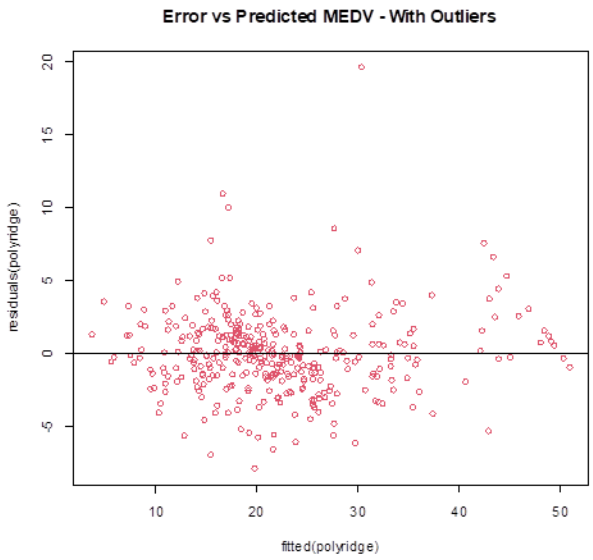
Fitting the Model for Poly Ridge after removing outliers.

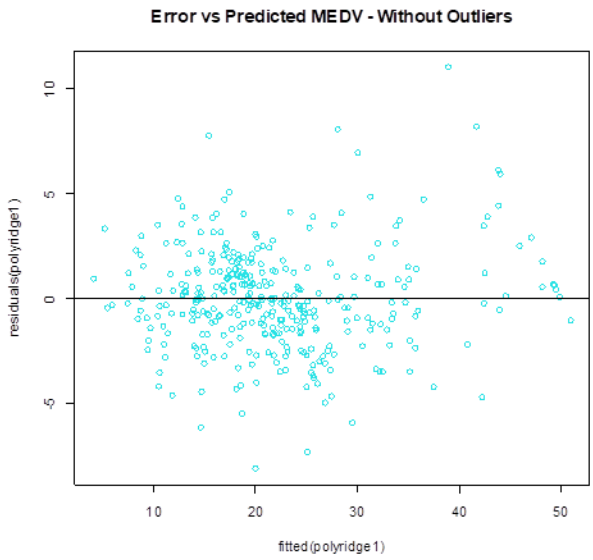
```
> #Removal of Outliers
> error=residuals(polyridge)
> n=length(error)
> MSRes=sum(error^2)/(n-14)
> data6=subset(train,(error/sqrt(MSRes))<3)
> polyridge1<-train(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),
+                   data=data6,method='glmnet',
+                   tunegrid=expand.grid(alpha=0,lambda=seq(0.0001,1,length=5)),
+                   trControl=custom)
+ Fold01.Rep1: alpha=0.10, lambda=1.445
- Fold01.Rep1: alpha=0.10, lambda=1.445
+ Fold01.Rep1: alpha=0.55, lambda=1.445
- Fold01.Rep1: alpha=0.55, lambda=1.445
+ Fold01.Rep1: alpha=1.00, lambda=1.445
- Fold01.Rep1: alpha=1.00, lambda=1.445
+ Fold02.Rep1: alpha=0.10, lambda=1.445
+ Fold10.Rep5: alpha=0.55, lambda=1.445
- Fold10.Rep5: alpha=0.55, lambda=1.445
+ Fold10.Rep5: alpha=1.00, lambda=1.445
- Fold10.Rep5: alpha=1.00, lambda=1.445
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.0144 on full training set
```

```
> #MODEL BUILDING USING BEST LAMBDA
> polyridge2=glmnet(x_train,y_train,alpha=0.1,lambda=0.0144)
> coef(polyridge2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  48.688856865
CRIM         -0.083539523
ZN            0.041003733
INDUS        -0.018113913
CHAS          3.371533643
NOX         -21.958401782
RM            3.037703508
AGE           0.005628192
DIS          -1.783495823
RAD           0.280230583
TAX          -0.010075074
PTRATIO      -1.133022699
B             0.007766838
LSTAT        -0.584926444
>
> mean(polyridge1$resample$RMSE)
[1] 3.354775
```

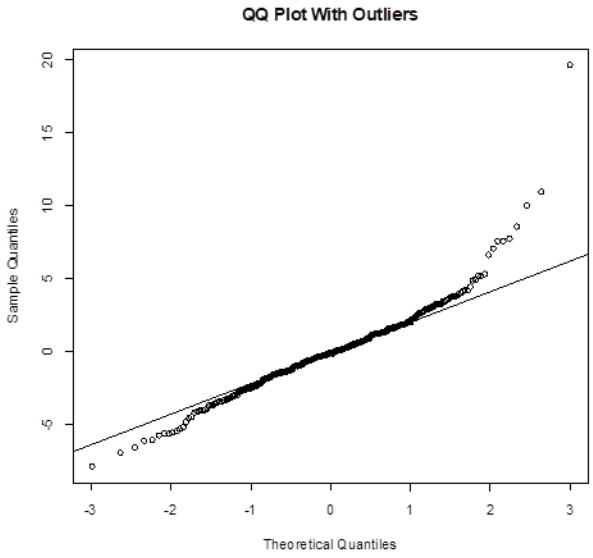
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

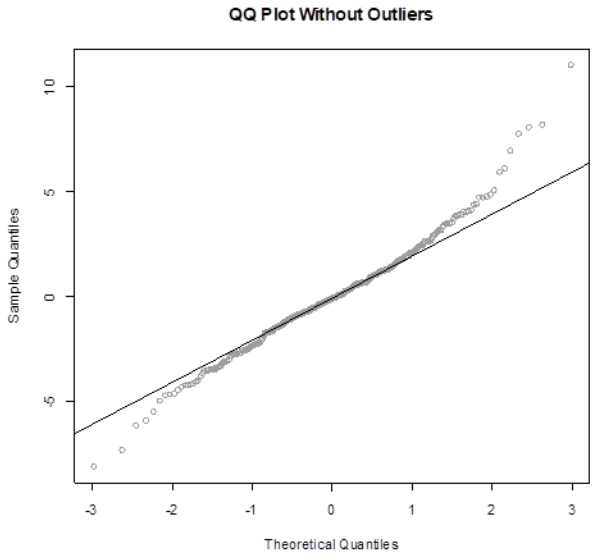
```
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted - OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(polyridge),residuals(polyridge),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(polyridge1),residuals(polyridge1),col=5,main="Error vs Predicted MEDV - Without Outliers")
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(polyridge),col=9,main="QQ Plot With Outliers")
> qqline(residuals(polyridge))
>
> qqnorm(residuals(polyridge1),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(polyridge1))
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(polyridge)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(polyridge1)),main="Density Plot without Outliers",col=5)
>
```



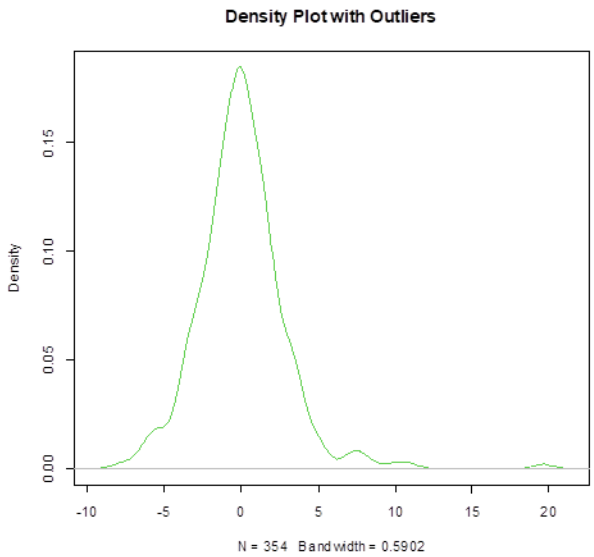


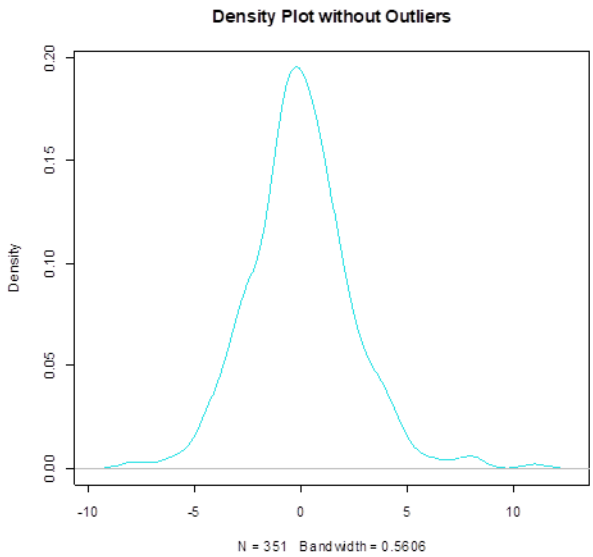
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.



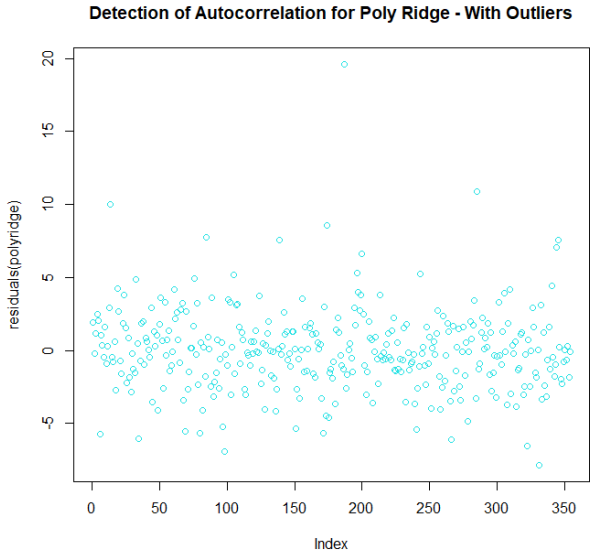


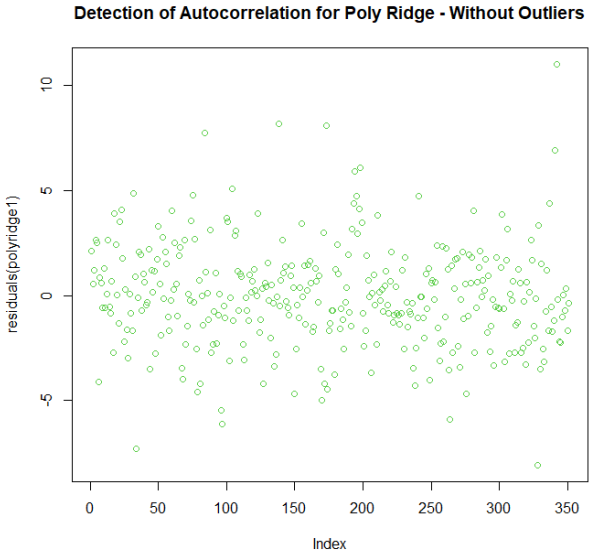
The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.





The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

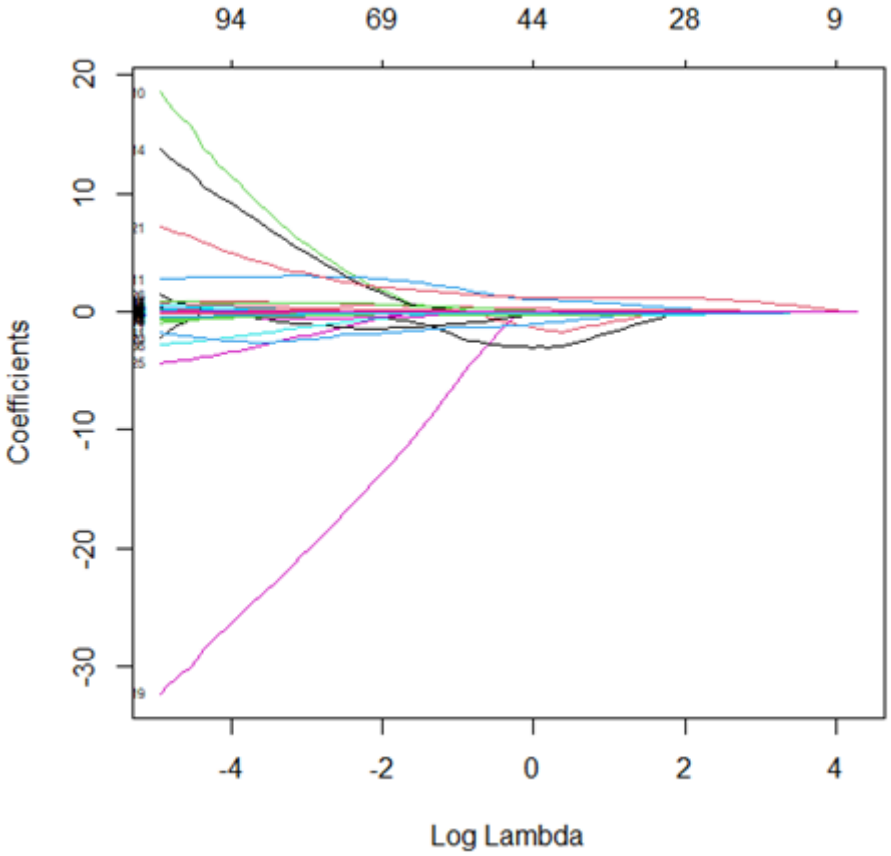


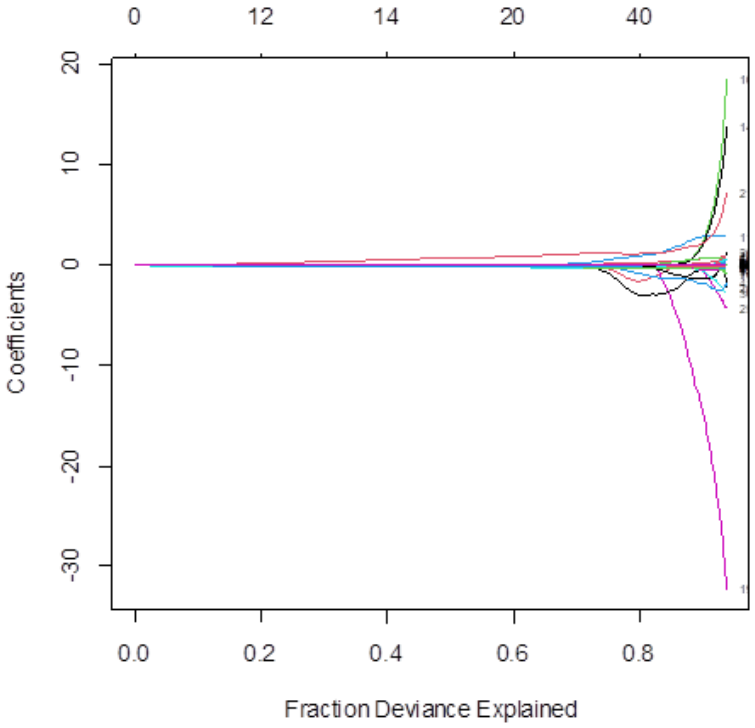


Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

Hence, it can be concluded that Poly Ridge without outliers is the best fit when compared to Poly Ridge (with outliers) without any violation of Assumptions.

PLOTTING RESULTS





```
> #plotting results
> plot(polyridge1)
> polyridge1
glmnet

351 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 316, 317, 315, 315, 315, 316, ...
Resampling results across tuning parameters:

  alpha  lambda      RMSE      Rsquared   MAE
  0.10   0.01444934  3.316795  0.8702564  2.340773
  0.10   0.14449340  3.613773  0.8510215  2.563079
  0.10   1.44493398  4.513437  0.7815064  3.179987
  0.55   0.01444934  3.383659  0.8639750  2.366128
  0.55   0.14449340  3.986180  0.8224094  2.775021
  0.55   1.44493398  4.816265  0.7642083  3.344434
  1.00   0.01444934  3.439523  0.8597559  2.397258
  1.00   0.14449340  4.059182  0.8172489  2.842358
  1.00   1.44493398  5.100003  0.7510797  3.594987

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1 and lambda = 0.01444934.
> plot(polyridge1$finalModel,xvar="lambda",label=T)
> plot(polyridge1$finalModel,xvar="dev",label=T)
```

Hence, it can be concluded that the Poly Ridge without outliers is the best fit when compared to Poly Ridge (with outliers).

5. Polynomial Lasso Model

```
> #POLY LASSO REGRESSION
>
> set.seed(123)
> polylasso<-train(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),
+                  data=train,method='glmnet',
+                  tuneGrid= expand.grid(alpha=1,lambda=seq(0.001,0.2,length=5)),
+                  trControl=custom)
+ Fold01.Rep1: alpha=1, lambda=0.2
- Fold01.Rep1: alpha=1, lambda=0.2
+ Fold02.Rep1: alpha=1, lambda=0.2
- Fold02.Rep1: alpha=1, lambda=0.2
+ Fold03.Rep1: alpha=1, lambda=0.2
- Fold03.Rep1: alpha=1, lambda=0.2
+ Fold04.Rep1: alpha=1, lambda=0.2
- Fold04.Rep1: alpha=1, lambda=0.2
+ Fold05.Rep1: alpha=1, lambda=0.2
- Fold05.Rep1: alpha=1, lambda=0.2
+ Fold06.Rep1: alpha=1, lambda=0.2
- Fold06.Rep1: alpha=1, lambda=0.2
- Fold07.Rep5: alpha=1, lambda=0.2
+ Fold08.Rep5: alpha=1, lambda=0.2
- Fold08.Rep5: alpha=1, lambda=0.2
+ Fold09.Rep5: alpha=1, lambda=0.2
- Fold09.Rep5: alpha=1, lambda=0.2
+ Fold10.Rep5: alpha=1, lambda=0.2
- Fold10.Rep5: alpha=1, lambda=0.2
Aggregating results
Selecting tuning parameters
Fitting alpha = 1, lambda = 0.001 on full training set
```

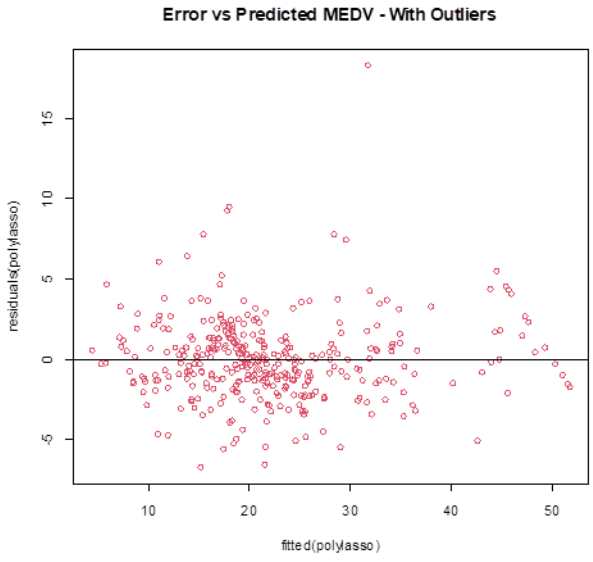
Fitting the Model for Poly Lasso after removing outliers.

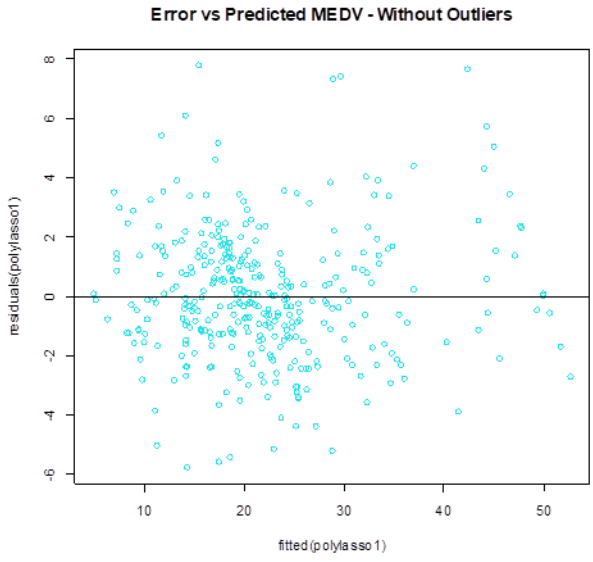
```
> #Removal of Outliers
> error=residuals(polylasso)
> n=length(error)
> MSRes=sum(error^2)/(n-14)
> data7=subset(train,(error/sqrt(MSRes))<3)
> polylasso1<-train(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),
+                   data=data7,method='glmnet',
+                   tuneGrid= expand.grid(alpha=1,lambda=seq(0.001,0.2,length=5)),
+                   trControl=custom)
+ Fold01.Rep1: alpha=1, lambda=0.2
- Fold01.Rep1: alpha=1, lambda=0.2
+ Fold02.Rep1: alpha=1, lambda=0.2
- Fold02.Rep1: alpha=1, lambda=0.2
+ Fold03.Rep1: alpha=1, lambda=0.2
- Fold03.Rep1: alpha=1, lambda=0.2
- Fold06.Rep5: alpha=1, lambda=0.2
+ Fold07.Rep5: alpha=1, lambda=0.2
- Fold07.Rep5: alpha=1, lambda=0.2
+ Fold08.Rep5: alpha=1, lambda=0.2
- Fold08.Rep5: alpha=1, lambda=0.2
+ Fold09.Rep5: alpha=1, lambda=0.2
- Fold09.Rep5: alpha=1, lambda=0.2
+ Fold10.Rep5: alpha=1, lambda=0.2
- Fold10.Rep5: alpha=1, lambda=0.2
Aggregating results
Selecting tuning parameters
Fitting alpha = 1, lambda = 0.001 on full training set
```

```
> polylasso2=glmnet(x_train,y_train,alpha=1,lambda=0.001)
> coef(polylasso2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  49.086106154
CRIM         -0.084369353
ZN            0.041450030
INDUS        -0.016266448
CHAS          3.368395143
NOX         -22.195942637
RM            3.020182495
AGE           0.005986928
DIS          -1.794998161
RAD           0.286127890
TAX          -0.010317401
PTRATIO      -1.137910770
B             0.007766150
LSTAT        -0.586911046
```

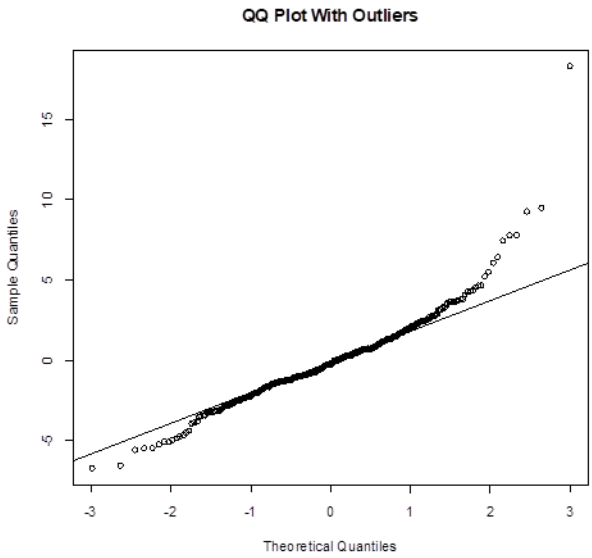
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

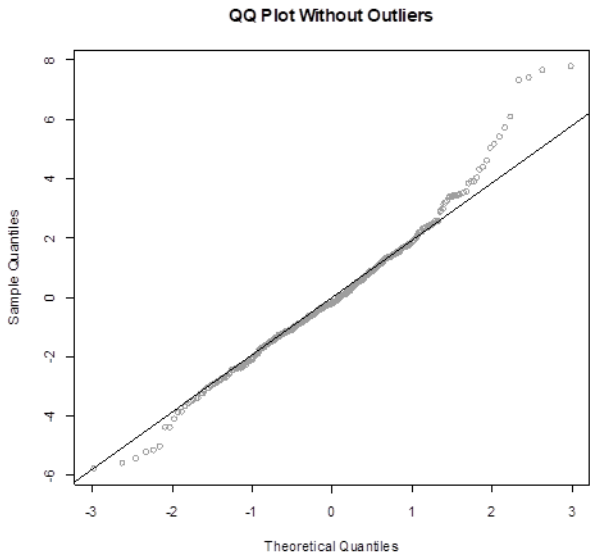
```
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted - OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(polylasso),residuals(polylasso),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(polylasso1),residuals(polylasso1),col=5,main="Error vs Predicted MEDV - Without Outliers")
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(polylasso),col=9,main="QQ Plot With Outliers")
> qqline(residuals(polylasso))
>
> qqnorm(residuals(polylasso1),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(polylasso1))
>
>
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(polylasso)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(polylasso1)),main="Density Plot without Outliers",col=5)
```



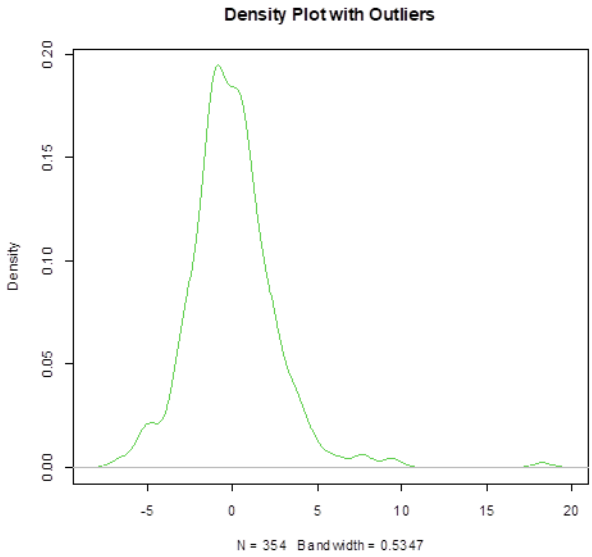


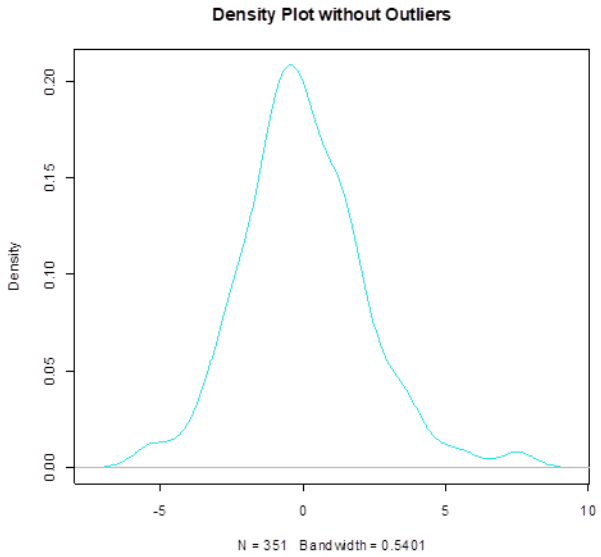
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.



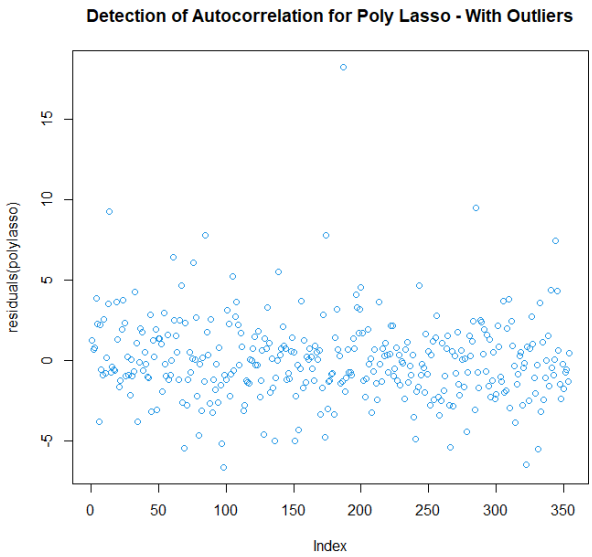


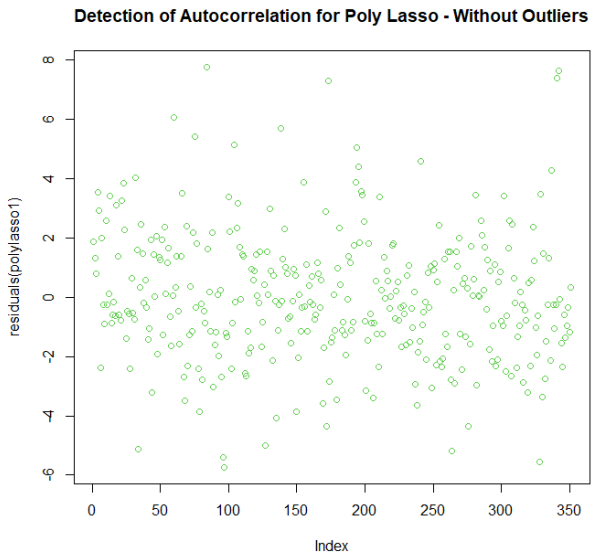
The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.





The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

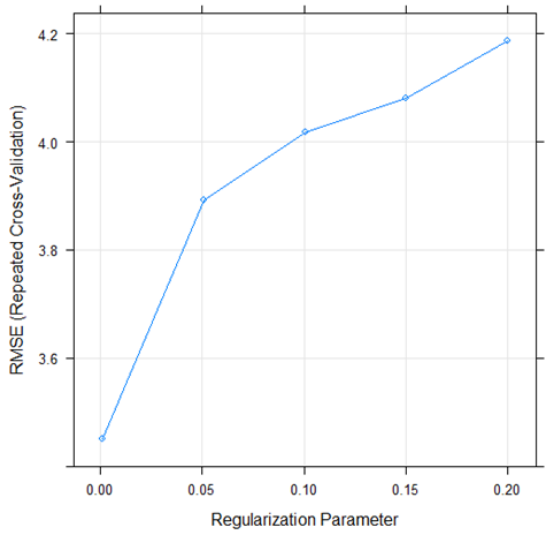


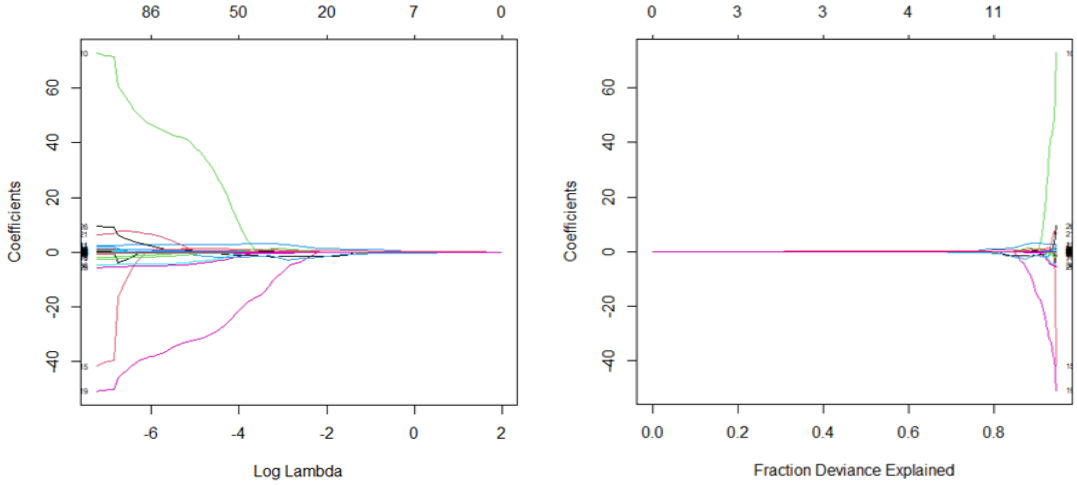


Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

Hence, it can be concluded that Poly Lasso without outliers is the best fit when compared to Poly Lasso (with outliers).

PLOTTING RESULTS:






```
> #Plot results
> plot(polylassol)
> polylassol
glmnet

351 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 317, 315, 315, 317, 316, 316, ...
Resampling results across tuning parameters:

  lambda   RMSE      Rsquared   MAE
  0.00100  3.451000  0.8624330  2.422392
  0.05075  3.892768  0.8238524  2.698531
  0.10050  4.017787  0.8152774  2.808577
  0.15025  4.079886  0.8117955  2.876286
  0.20000  4.186127  0.8039675  2.947946

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.001.
> plot(polylassol$finalModel,xvar='lambda',label=T)
> plot(polylassol$finalModel,xvar='dev',label=T)
```

6. Polynomial Elastic Net Model

```
> set.seed(123)
> polyelast_net<- train(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),
+                       data=train,method='glmnet',
+                       tuneGrid=expand.grid(alpha=seq(0,1,length=10),
+                       lambda=seq(0.0001,1,length=5)),trControl=custom)
+ Fold01.Rep1: alpha=0.0000, lambda=1
- Fold01.Rep1: alpha=0.0000, lambda=1
+ Fold01.Rep1: alpha=0.1111, lambda=1
- Fold01.Rep1: alpha=0.1111, lambda=1
+ Fold01.Rep1: alpha=0.2222, lambda=1
- Fold01.Rep1: alpha=0.2222, lambda=1
+ Fold01.Rep1: alpha=0.3333, lambda=1
+ Fold10.Rep5: alpha=0.7778, lambda=1
- Fold10.Rep5: alpha=0.7778, lambda=1
+ Fold10.Rep5: alpha=0.8889, lambda=1
- Fold10.Rep5: alpha=0.8889, lambda=1
+ Fold10.Rep5: alpha=1.0000, lambda=1
- Fold10.Rep5: alpha=1.0000, lambda=1
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.667, lambda = 1e-04 on full training set
```

Fitting the Model for Poly Elastic Net after removing outliers.

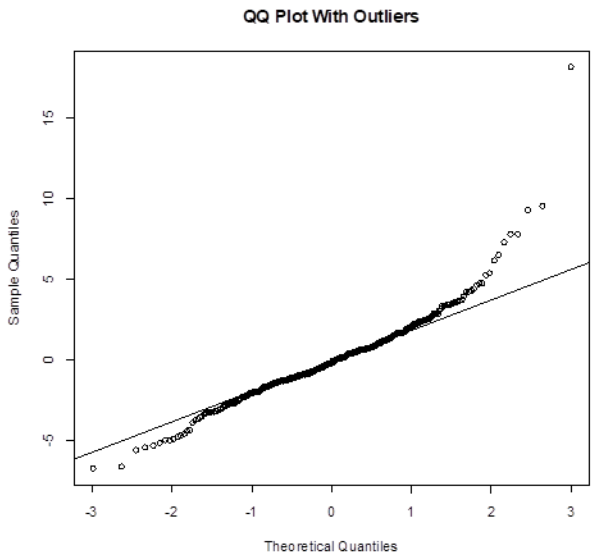
```
> #Removal of Outliers
> error=residuals(polyelast_net)
> n=length(error)
> MSRes=sum(error^2)/(n-14)
> data8=subset(train,(error/sqrt(MSRes))<3)
> polyelast_net1<- train(MEDV~poly(CRIM,ZN,INDUS,CHAS,NOX,RM,AGE,DIS,RAD,TAX,PTRATIO,B,LSTAT,degree=2,raw=TRUE),
+                       data=data8,method='glmnet',
+                       tuneGrid=expand.grid(alpha=seq(0,1,length=10),
+                       lambda=seq(0.0001,1,length=5)),trControl=custom)
+ Fold01.Rep1: alpha=0.0000, lambda=1
- Fold01.Rep1: alpha=0.0000, lambda=1
+ Fold01.Rep1: alpha=0.1111, lambda=1
- Fold01.Rep1: alpha=0.1111, lambda=1
+ Fold01.Rep1: alpha=0.2222, lambda=1
- Fold01.Rep1: alpha=0.2222, lambda=1
+ Fold01.Rep1: alpha=0.3333, lambda=1
- Fold01.Rep1: alpha=0.3333, lambda=1
+ Fold01.Rep1: alpha=0.4444, lambda=1
- Fold01.Rep1: alpha=0.4444, lambda=1
- Fold10.Rep5: alpha=0.6667, lambda=1
+ Fold10.Rep5: alpha=0.7778, lambda=1
- Fold10.Rep5: alpha=0.7778, lambda=1
+ Fold10.Rep5: alpha=0.8889, lambda=1
- Fold10.Rep5: alpha=0.8889, lambda=1
+ Fold10.Rep5: alpha=1.0000, lambda=1
- Fold10.Rep5: alpha=1.0000, lambda=1
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.111, lambda = 1e-04 on full training set
```

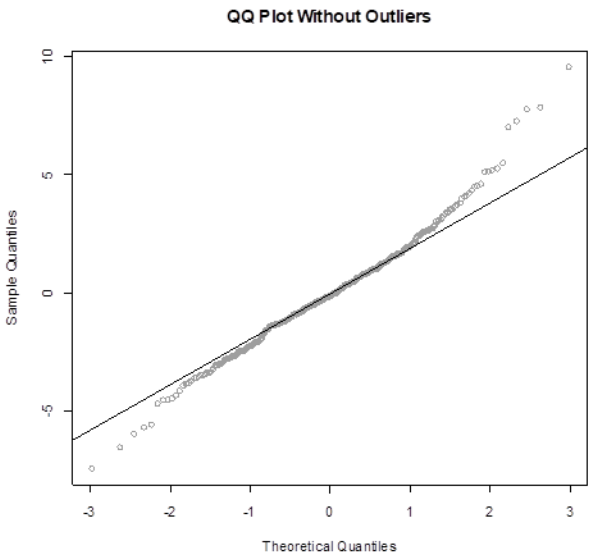
```
> #MODEL BUILDING USING BEST LAMBDA
> polyelast_net2=glmnet(x_train,y_train,alpha=0.111,lambda=0.0001)
> coef(polyelast_net2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  49.166334798
CRIM         -0.084776764
ZN            0.041621877
INDUS        -0.016147884
CHAS          3.369686034
NOX         -22.250597480
RM            3.016356524
AGE           0.006172314
DIS          -1.797441869
RAD           0.287866003
TAX          -0.010386175
PTRATIO      -1.138975096
B             0.007773633
LSTAT        -0.587121117
```

The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

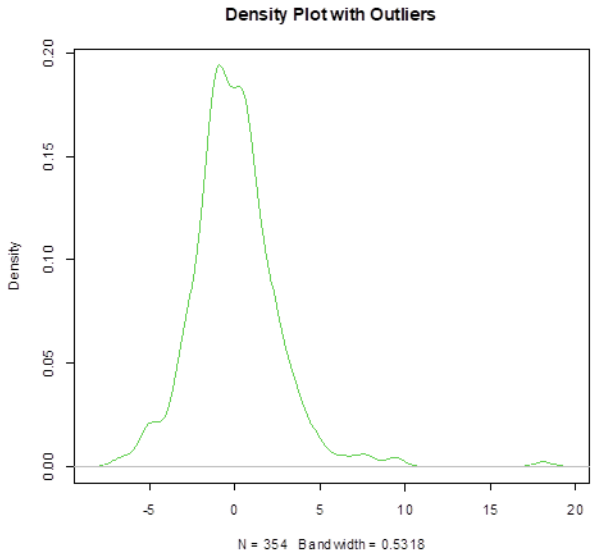
```
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted - OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(polyelast_net),residuals(polyelast_net),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(polyelast_net1),residuals(polyelast_net1),col=5,main="Error vs Predicted MEDV - Without Outliers")
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(polyelast_net),col=9,main="QQ Plot With Outliers")
> qqline(residuals(polyelast_net))
>
> qqnorm(residuals(polyelast_net1),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(polyelast_net1))
>
>
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(polyelast_net)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(polyelast_net1)),main="Density Plot without Outliers",col=5)
```

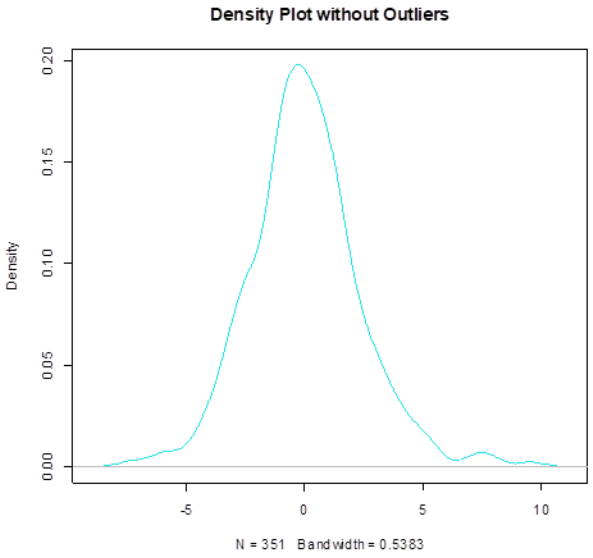
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.



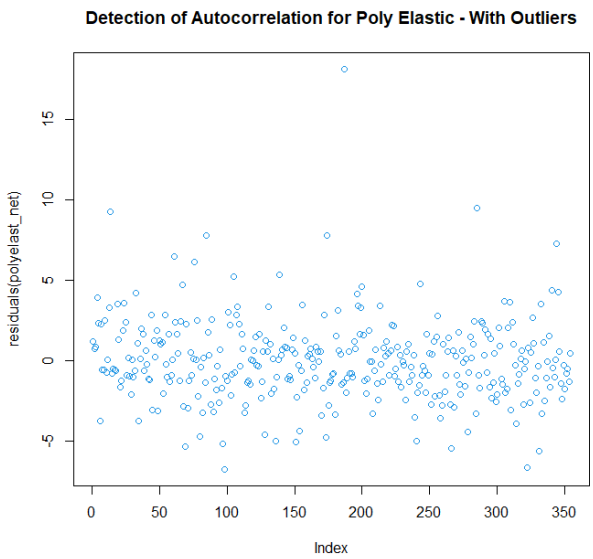


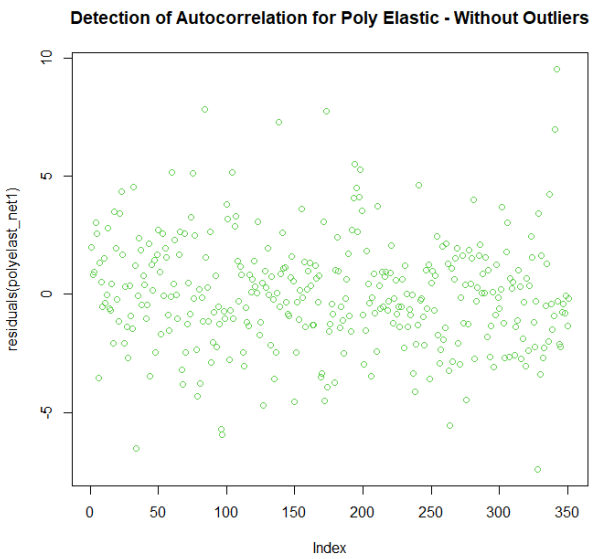
The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.





The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

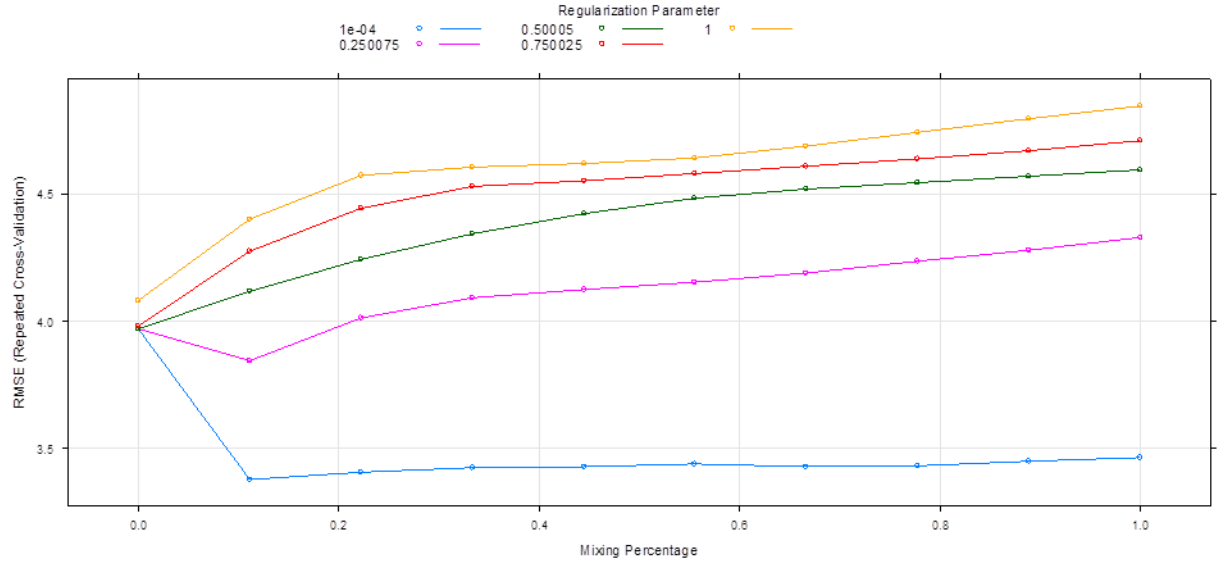


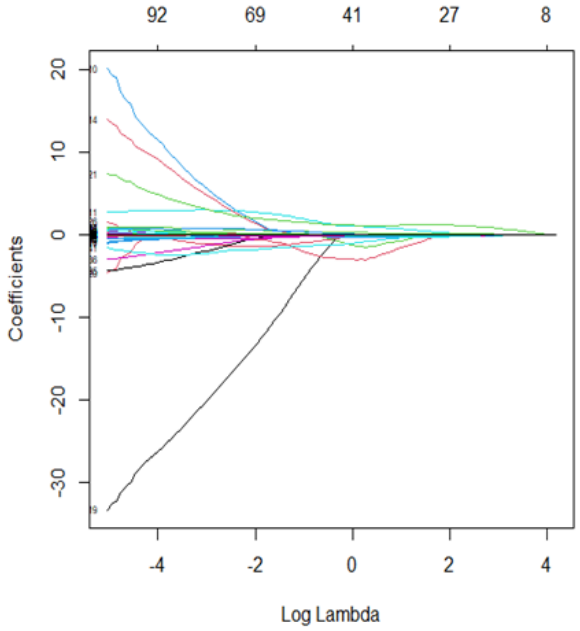


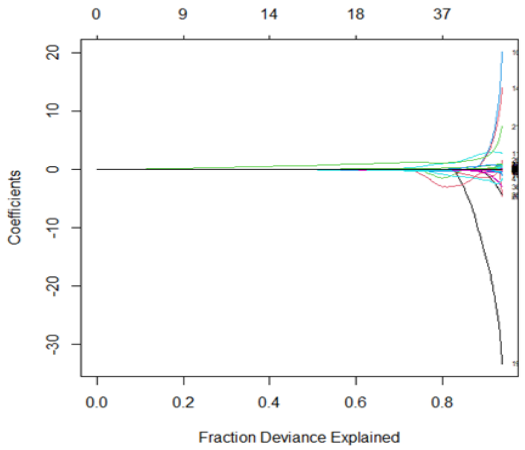
Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

Hence, it can be concluded that Poly Elastic Net without outliers is the best fit when compared to Poly Elastic Net (with outliers).

PLOTTING RESULTS







```
> #Plot results
>
> plot(polyelast_net1)
> polyelast_net1
glmnet

351 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 316, 315, 315, 316, 317, 316, ...
Resampling results across tuning parameters:

  alpha      lambda    RMSE      Rsquared   MAE
  0.0000000  0.000100  3.969387  0.8290711  2.816117
  0.0000000  0.250075  3.969387  0.8290711  2.816117
  0.0000000  0.500050  3.969387  0.8290711  2.816117
  0.0000000  0.750025  3.982009  0.8280999  2.825036
  0.0000000  1.000000  4.081532  0.8198836  2.897386
  0.1111111  0.000100  3.379455  0.8716965  2.371678
  0.1111111  0.250075  3.844060  0.8378822  2.688298
  0.1111111  0.500050  4.118524  0.8156635  2.878950
  0.1111111  0.750025  4.274002  0.8022733  3.006837
  0.1111111  1.000000  4.398461  0.7912807  3.104188
  0.6666667  0.750025  4.606985  0.7757093  3.224876
  0.6666667  1.000000  4.685650  0.7726841  3.274192
  0.7777778  0.000100  3.434268  0.8681707  2.409292
  0.7777778  0.250075  4.233337  0.8056153  2.977014
  0.7777778  0.500050  4.544125  0.7793520  3.189570
  0.7777778  0.750025  4.636185  0.7744500  3.247069
  0.7777778  1.000000  4.739143  0.7698926  3.321815
  0.8888889  0.000100  3.452536  0.8672278  2.423765
  0.8888889  0.250075  4.278778  0.8016663  3.002567
  0.8888889  0.500050  4.568708  0.7776354  3.206764
  0.8888889  0.750025  4.669416  0.7728913  3.272519
  0.8888889  1.000000  4.791571  0.7672470  3.364748
  1.0000000  0.000100  3.467283  0.8663045  2.433088
  1.0000000  0.250075  4.326196  0.7975264  3.030589
  1.0000000  0.500050  4.592195  0.7761571  3.223227
  1.0000000  0.750025  4.706467  0.7709589  3.301779
  1.0000000  1.000000  4.844741  0.7641987  3.410425

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1111111 and lambda = 1e-04.
> plot(polyelast_net1$finalModel, xvar='lambda',label=T)
> plot(polyelast_net1$finalModel, xvar='dev',label=T)
```

7. Ridge Regression Model

```
#RIDGE REGRESSION
#install.packages("caret")
custom<- trainControl(method = "repeatedcv",number=10,repeats=5,verboseIter= T)

set.seed(123)
ridge<-train(MEDV~.,data= train,method='glmnet',
             tunegrid=expand.grid(alpha=0,lambda=seq(0.0001,1,length=5)),
             trControl=custom)
Fold01.Rep1: alpha=0.10, lambda=1.418
Fold01.Rep1: alpha=0.10, lambda=1.418
Fold01.Rep1: alpha=0.55, lambda=1.418
Fold01.Rep1: alpha=0.55, lambda=1.418
Fold01.Rep1: alpha=1.00, lambda=1.418
+ Fold10.Rep5: alpha=1.00, lambda=1.418
- Fold10.Rep5: alpha=1.00, lambda=1.418
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.142 on full training set
> ridge
glmnet

354 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 318, 318, 318, 320, 318, 318, ...
Resampling results across tuning parameters:

  alpha  lambda      RMSE      Rsquared   MAE
  0.10   0.01417921  5.069491  0.7216543  3.648801
  0.10   0.14179214  5.059265  0.7226626  3.609159
  0.10   1.41792136  5.195943  0.7131704  3.595436
  0.55   0.01417921  5.069165  0.7217487  3.647507
  0.55   0.14179214  5.077429  0.7215284  3.591333
  0.55   1.41792136  5.573407  0.6813897  3.908612
  1.00   0.01417921  5.068074  0.7219051  3.642784
  1.00   0.14179214  5.127482  0.7172876  3.601763
  1.00   1.41792136  5.746328  0.6763555  4.057637

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1 and lambda = 0.1417921.
```

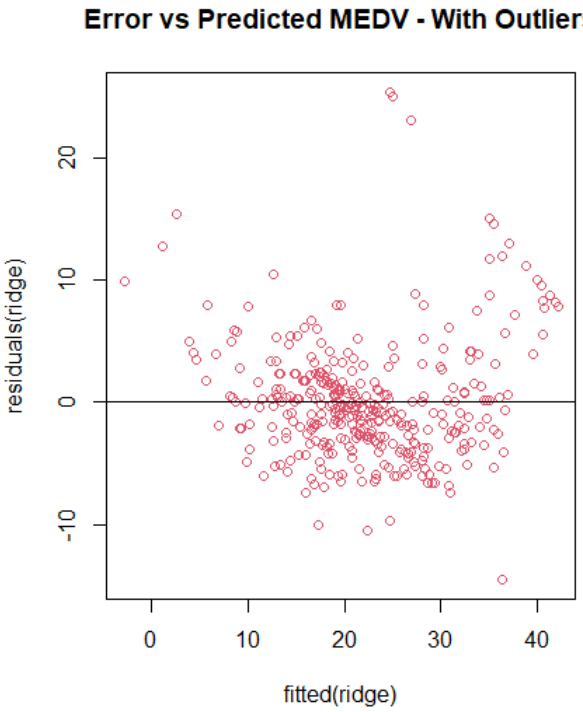
Ridge Regression model after removing the outliers is given below:

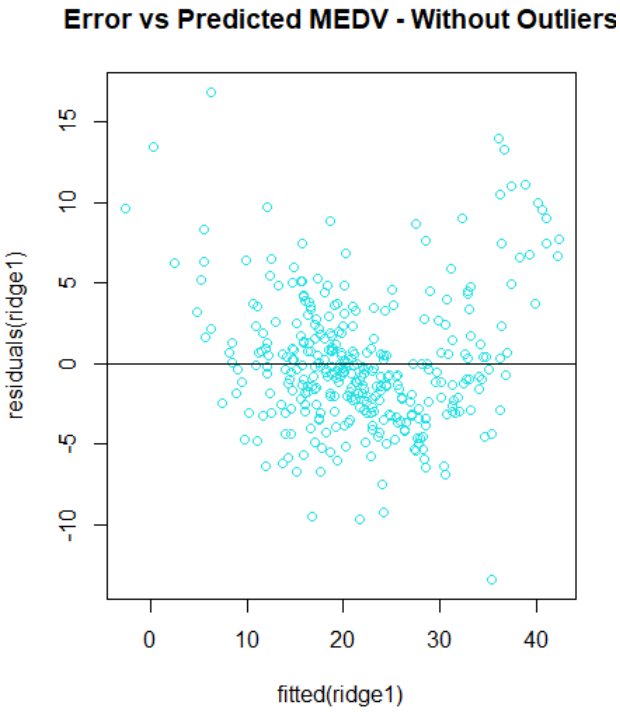
```
#MODEL AFTER REMOVING OUTLIERS:

error=residuals(ridge)
n=length(error)
MSRes= sum(error^2)/(n-14)
data9=subset(train,(error/sqrt(MSRes))<3)
ridge1<-train(MEDV~.,
              data= data9,
              method='glmnet',
              tunegrid=expand.grid(alpha=0,
                                   lambda=seq(0.0001,1,length=5)),
              trControl=custom)
Fold01.Rep1: alpha=0.10, lambda=1.387
Fold01.Rep1: alpha=0.10, lambda=1.387
Fold01.Rep1: alpha=0.55, lambda=1.387
Fold01.Rep1: alpha=0.55, lambda=1.387
+ Fold10.Rep5: alpha=1.00, lambda=1.387
- Fold10.Rep5: alpha=1.00, lambda=1.387
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.139 on full training set
> ridge1
glmnet

349 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 314, 313, 313, 314, 314, 313, ...
Resampling results across tuning parameters:

  alpha  lambda      RMSE      Rsquared   MAE
  0.10   0.01386975  4.270022  0.7877244  3.181592
  0.10   0.13869752  4.266912  0.7881790  3.165388
  0.10   1.38697516  4.376719  0.7819115  3.206643
  0.55   0.01386975  4.271511  0.7876125  3.182739
  0.55   0.13869752  4.304172  0.7850102  3.171442
  0.55   1.38697516  4.635002  0.7648876  3.428950
  1.00   0.01386975  4.272767  0.7875035  3.181010
  1.00   0.13869752  4.339839  0.7817185  3.186146
  1.00   1.38697516  4.871164  0.7560373  3.572588

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1 and lambda = 0.1386975.
```

```
> #Model building using best lambda
> ridge2=glmnet(x_train,y_train,alpha=0.1,lambda=0.1386975)
> coef(ridge2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  44.982898153
CRIM         -0.074154356
ZN            0.036300779
INDUS        -0.032019086
CHAS          3.375348527
NOX         -19.649012724
RM            3.198393685
AGE           0.001473376
DIS          -1.669995645
RAD           0.223722407
TAX          -0.007846644
PTRATIO      -1.085803259
B             0.007695768
LSTAT        -0.567369890
```

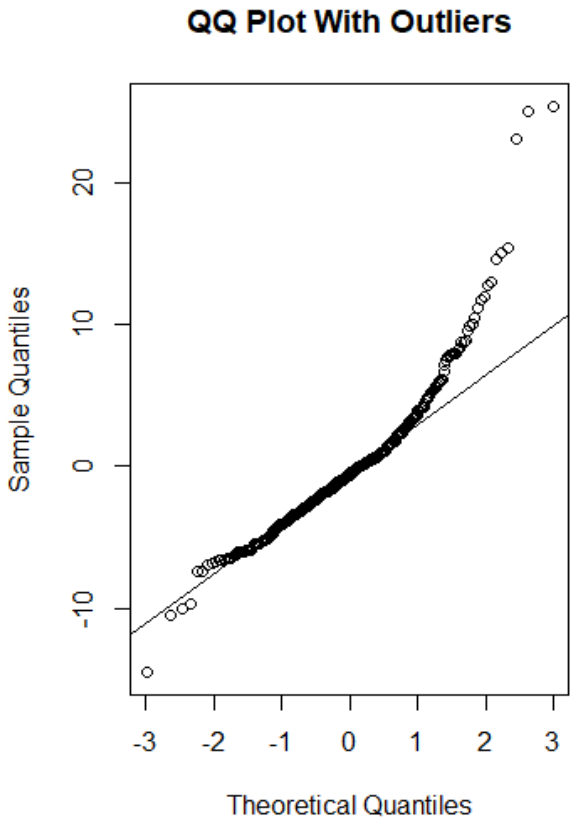
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

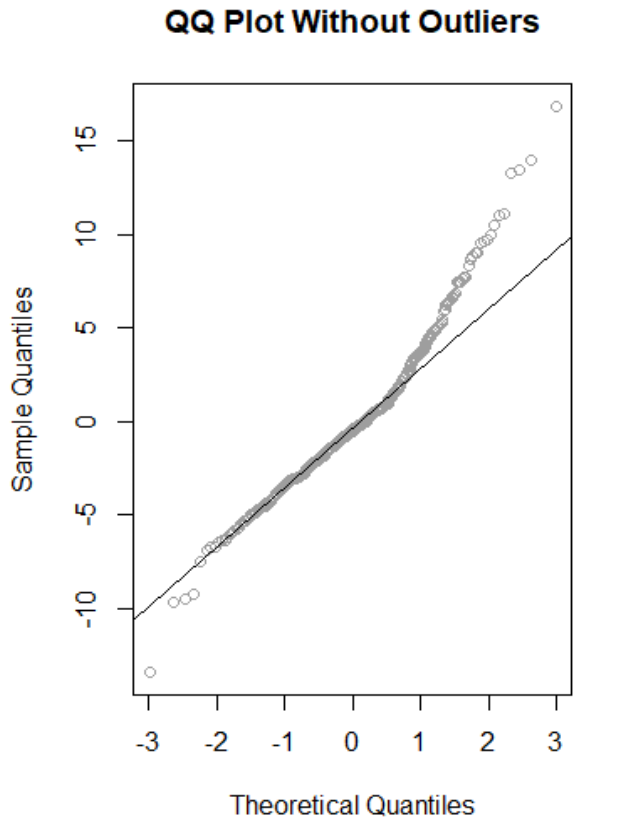
```
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted <- OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted <- OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(ridge),residuals(ridge),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(ridge1),residuals(ridge1),col=5,main="Error vs Predicted MEDV - Without Outli$
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(ridge),col=9,main="QQ Plot With Outliers")
> qqline(residuals(ridge))
>
> qqnorm(residuals(ridge1),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(ridge1))
>
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(ridge)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(ridge1)),main="Density Plot without Outliers",col=5)
```



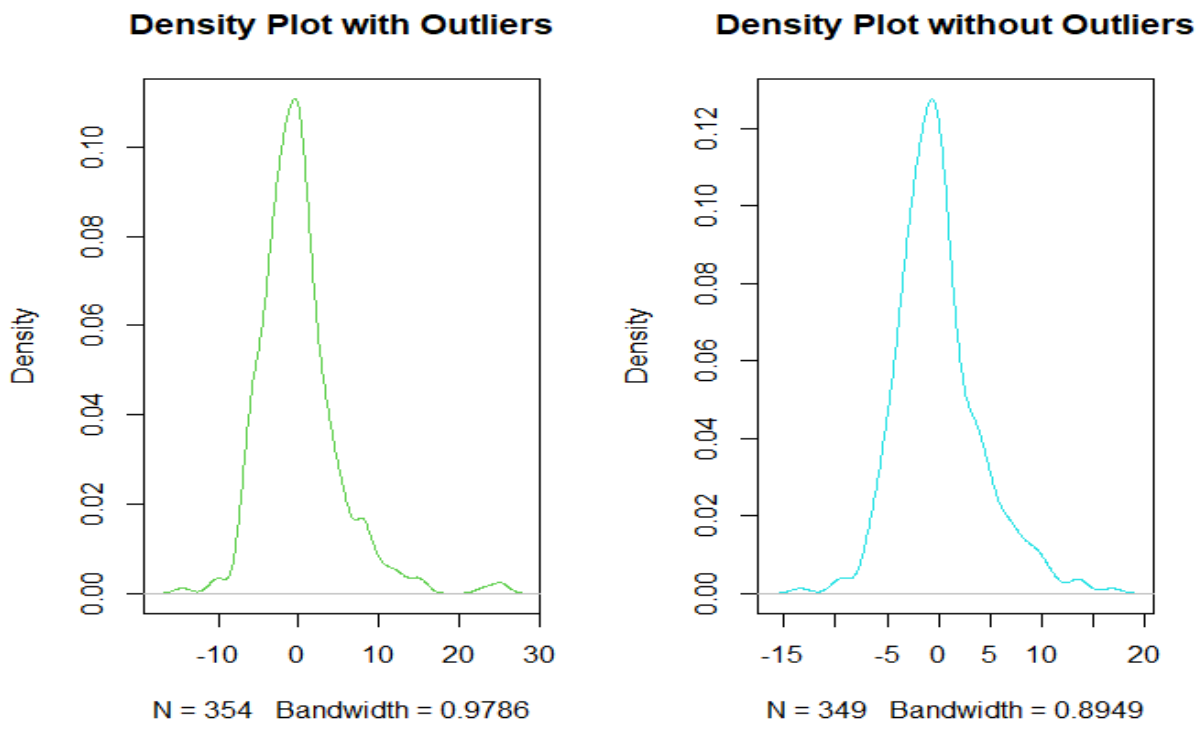


It can be seen that the Residual Plot without outliers follows a horizontal band fashion.

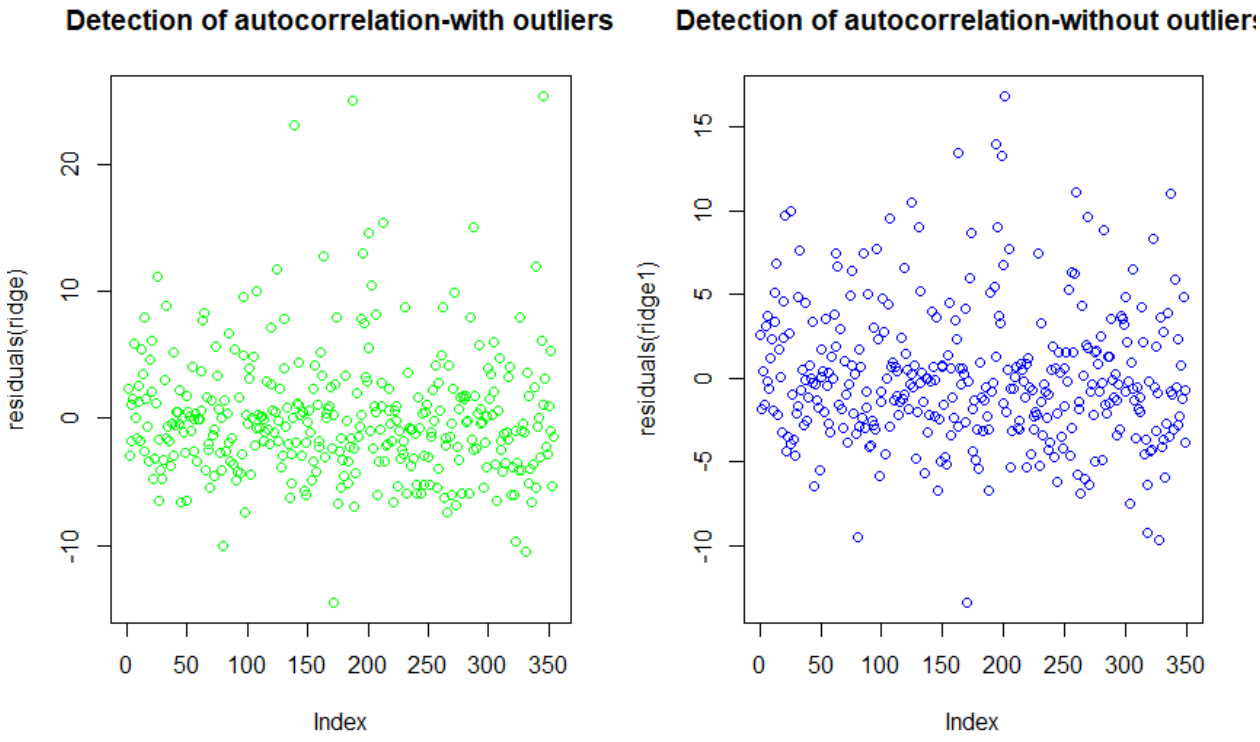




The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.



The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

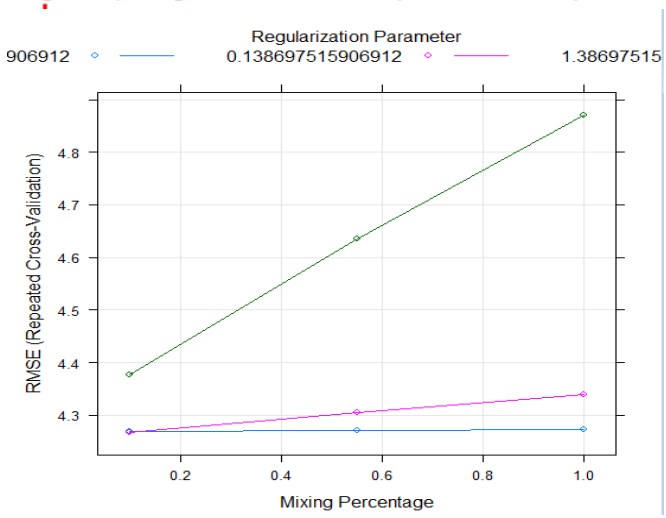


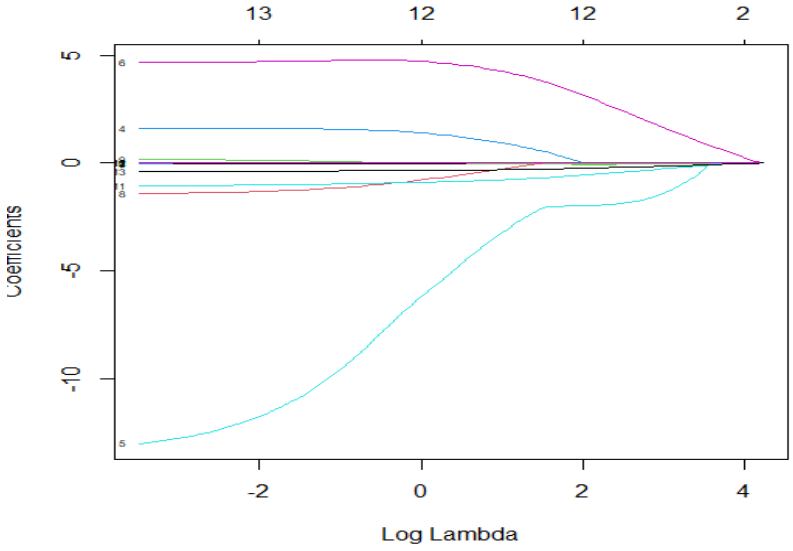
Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

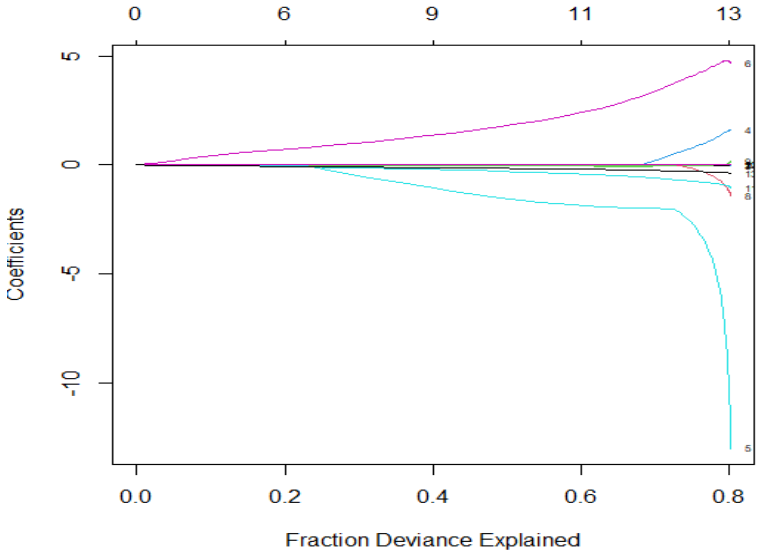
Hence, it can be concluded that Ridge Regression (without outliers) is the best fit when compared to Ridge Regression (with outliers).

PLOTTING RESULTS:

```
> #plotting results
> plot(ridge1)
> plot(ridge1$finalModel,xvar="lambda",label=T)
> plot(ridge1$finalModel,xvar="dev",label=T)
```







8. LASSO REGRESSION MODEL

```
> #LASSO REGRESSION
>
> set.seed(123)
> lasso<-train(MEDV ~ .,
+              train,                    .
+              method='glmnet',
+              tuneGrid= expand.grid(alpha=1,lambda=seq(0.001,0.2,length=5)),
+              trControl=custom)
+ Fold01.Rep1: alpha=1, lambda=0.2
- Fold01.Rep1: alpha=1, lambda=0.2
+ Fold02.Rep1: alpha=1, lambda=0.2
- Fold02.Rep1: alpha=1, lambda=0.2
+ Fold03.Rep1: alpha=1, lambda=0.2
- Fold03.Rep1: alpha=1, lambda=0.2
+ Fold04.Rep1: alpha=1, lambda=0.2
+ Fold10.Rep5: alpha=1, lambda=0.2
- Fold10.Rep5: alpha=1, lambda=0.2
Aggregating results
Selecting tuning parameters
Fitting alpha = 1, lambda = 0.0508 on full training set
> lasso
glmnet

354 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 318, 318, 318, 320, 318, 318, ...
Resampling results across tuning parameters:

  lambda   RMSE      Rsquared   MAE
  0.00100  5.070047  0.7216574  3.650460
  0.05075  5.068020  0.7221822  3.615445
  0.10050  5.092475  0.7202824  3.598487
  0.15025  5.133123  0.7168161  3.602164
  0.20000  5.158359  0.7147392  3.606428

Tuning parameter 'alpha' was held constant at a value of 1
RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 1 and lambda = 0.05075.
```

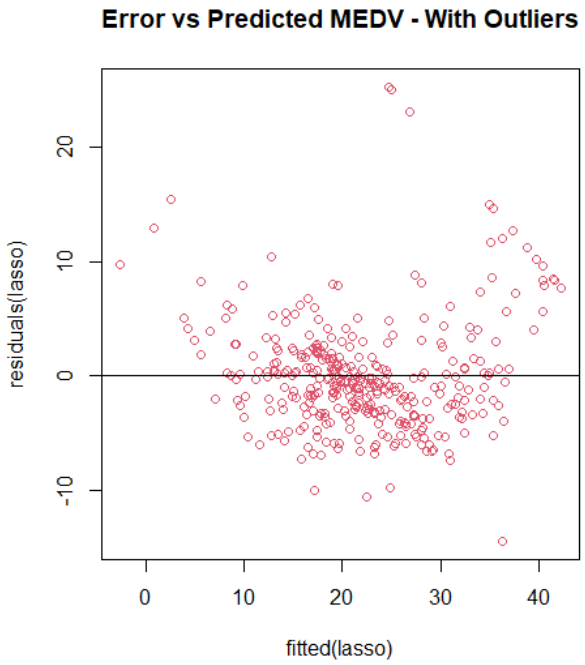
Lasso Regression model after removing the outliers is shown below:

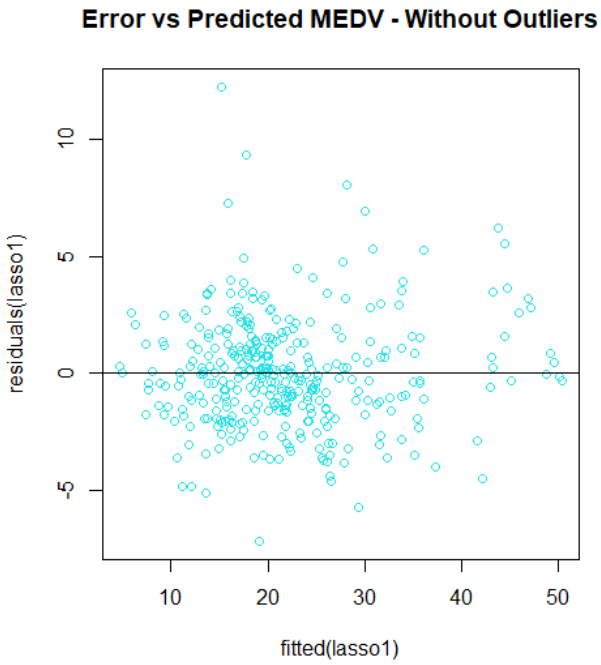
```
> #MODEL AFTER REMOVING OUTLIERS:
>
> error=residuals(lasso)
> n=length(error)
> MSRes= sum(error^2)/(n-14)
> data10=subset(train,(error/sqrt(MSRes))<3)
> lasso1<-train (MEDV~.,
+                data= data10,
+                method='glmnet',
+                tunegrid=expand.grid(alpha=0,
+                                     lambda=seq(0.0001,1,length=5)),
+                trControl=custom)
+ Fold01.Rep1: alpha=0.10, lambda=1.387
- Fold01.Rep1: alpha=0.10, lambda=1.387
+ Fold10.Rep5: alpha=1.00, lambda=1.387
- Fold10.Rep5: alpha=1.00, lambda=1.387
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.139 on full training set
> lasso1
glmnet

349 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 314, 314, 314, 314, 314, 315, ...
Resampling results across tuning parameters:

  alpha  lambda      RMSE      Rsquared   MAE
  0.10   0.01386975  4.214094  0.7913267  3.164777
  0.10   0.13869752  4.213297  0.7914296  3.149913
  0.10   1.38697516  4.337470  0.7850633  3.196815
  0.55   0.01386975  4.215603  0.7911918  3.165924
  0.55   0.13869752  4.251532  0.7880265  3.154397
  0.55   1.38697516  4.603295  0.7708036  3.428020
  1.00   0.01386975  4.216781  0.7910791  3.164176
  1.00   0.13869752  4.283798  0.7854098  3.169629
  1.00   1.38697516  4.840933  0.7620146  3.573047

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1 and lambda = 0.1386975.
```

```
> #Model building using best lambda
> lasso2=glmnet(x_train,y_train,alpha=0.1,lambda=0.1386975)
> coef(lasso2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  44.982898153
CRIM         -0.074154356
ZN            0.036300779
INDUS        -0.032019086
CHAS          3.375348527
NOX         -19.649012724
RM            3.198393685
AGE           0.001473376
DIS          -1.669995645
RAD           0.223722407
TAX          -0.007846644
PTRATIO      -1.085803259
B             0.007695768
LSTAT        -0.567369890
```

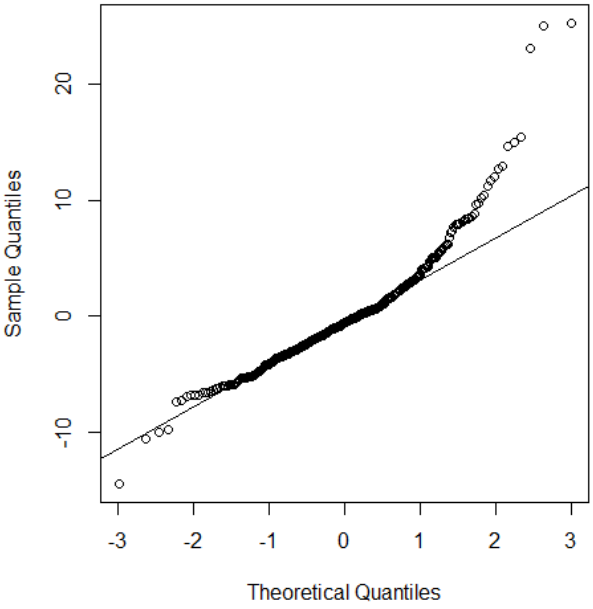
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

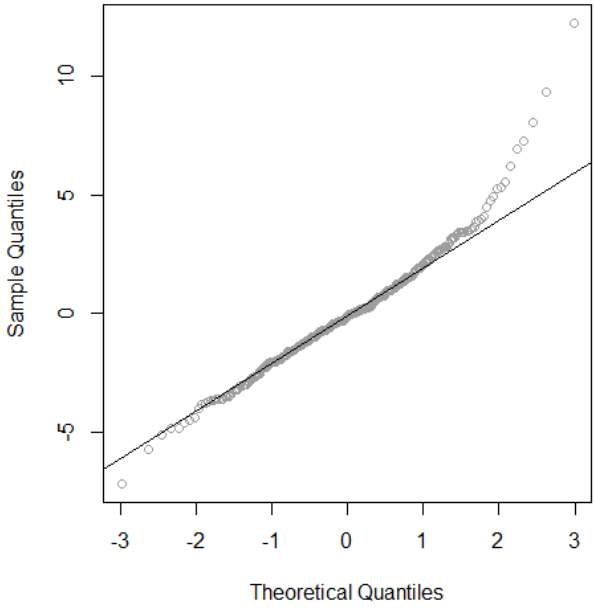
```
> #1. TO CHECK THE NORMALITY OF ERROR
>
> #Error vs Predicted <- OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(lasso),residuals(lasso),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(lasso1),residuals(lasso1),col=5,main="Error vs Predicted MEDV - Without Outliers")
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(lasso),col=9,main="QQ Plot With Outliers")
> qqline(residuals(lasso))
>
> qqnorm(residuals(lasso1),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(lasso1))
>
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(lasso)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(lasso1)),main="Density Plot without Outliers",col=5)
```



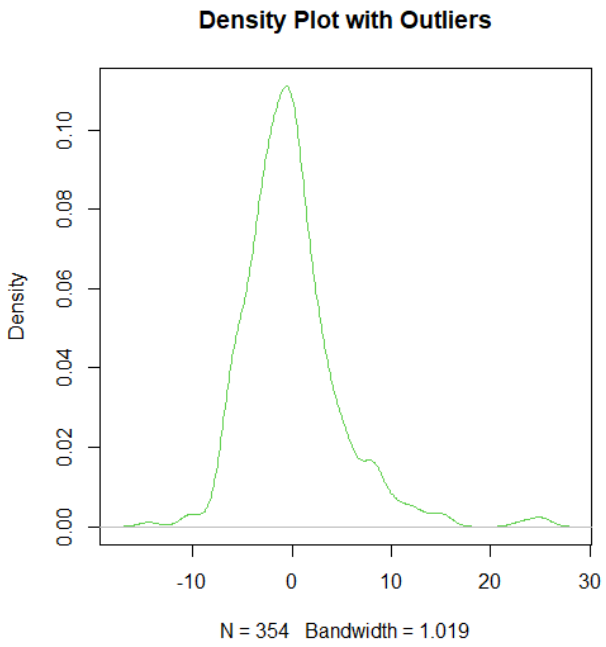


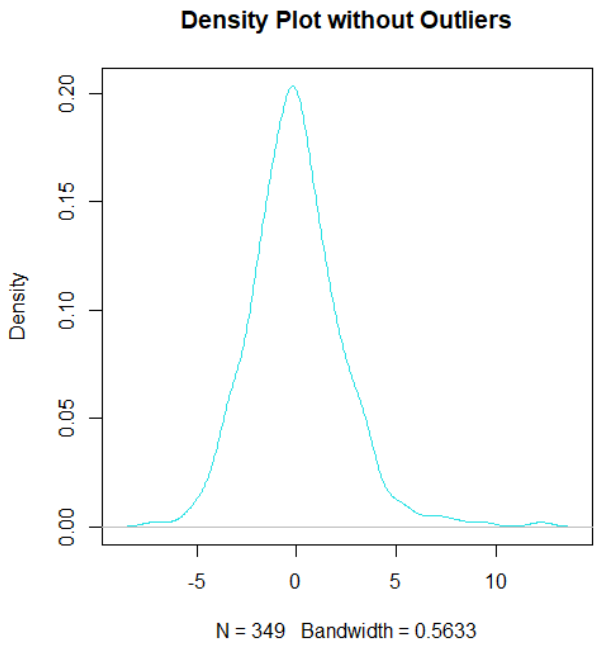
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.





The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.





The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

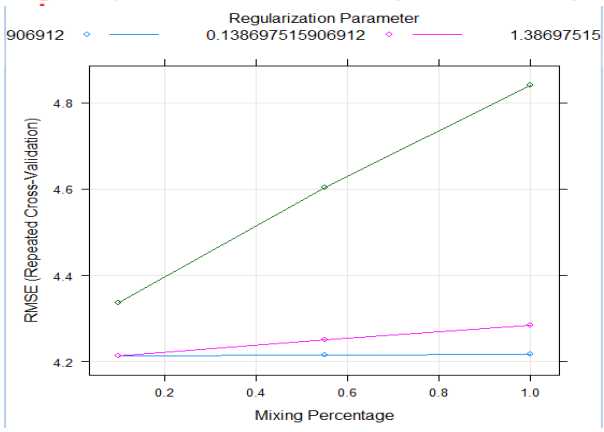
```
> #AUTOCORRELATION
> par(mfrow=c(1,2))
> plot(residuals(lasso),col="red",main="Detection of autocorrelation-with outliers")
> plot(residuals(lassol),col="yellow",main="Detection of autocorrelation-without outliers")
```

Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

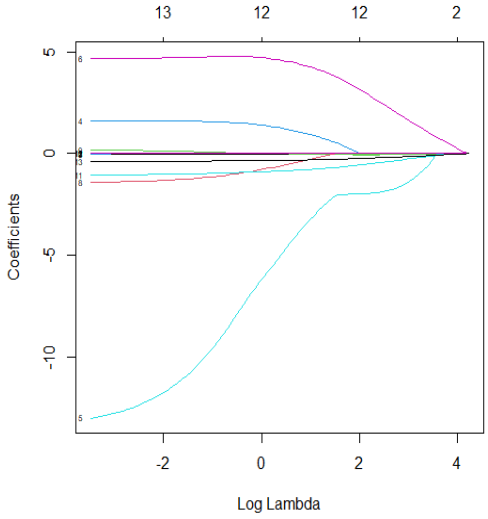
Hence, it can be concluded that Lasso Regression (without outliers) is the best fit when compared to Lasso Regression (with outliers) without any violation of Assumptions.

PLOTTING RESULTS:

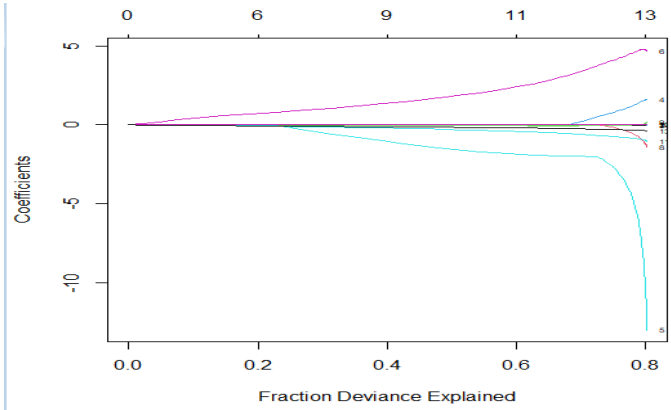
```
> #Plotting results:
> plot(lasso1)
> plot(lasso1$finalModel,xvar='lambda',label=T)
> plot(lasso1$finalModel,xvar='dev',label=T)
```



In y-axis of the above plot the RMSE was calculated using Repeated Cross Validation method .We can see that for higher values of lambda the error increases. So the best value of lambda in this model is 0.13869.



From the above plot we can see that as we relax the lambda the value of coefficients increases. So increasing the value of lambda helps us to reduce the size of the coefficients.



80% of deviance is explained in the above plot.

9. ELASTIC NET REGRESSION MODEL

```
> #Elastic Net Regression
> set.seed(123)
> elast_net<- train(MEDV ~.,
+                   train,
+                   method='glmnet',
+                   tuneGrid=expand.grid(alpha=seq(0,1,length=10),
+                                        lambda=seq(0.0001,1,length=5)),
+                   trControl=custom)
+ Fold01.Rep1: alpha=0.0000, lambda=1
- Fold01.Rep1: alpha=0.0000, lambda=1
- Fold10.Rep5: alpha=0.7778, lambda=1
+ Fold10.Rep5: alpha=0.8889, lambda=1
- Fold10.Rep5: alpha=0.8889, lambda=1
+ Fold10.Rep5: alpha=1.0000, lambda=1
- Fold10.Rep5: alpha=1.0000, lambda=1
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.111, lambda = 0.25 on full training set
> elast_net
glmnet

354 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)

Summary of sample sizes: 318, 318, 318, 320, 318, 318, ...
Resampling results across tuning parameters:

  alpha      lambda    RMSE      Rsquared   MAE
  0.0000000  0.000100  5.062826  0.7225521  3.542875
  0.0000000  0.250075  5.062826  0.7225521  3.542875
  0.0000000  0.500050  5.062826  0.7225521  3.542875
  0.0000000  0.750025  5.065704  0.7224009  3.541583
  0.0000000  1.000000  5.079551  0.7215381  3.534120
  0.1111111  0.000100  5.069583  0.7216451  3.648912
  0.1111111  0.250075  5.058292  0.7229075  3.582303
  0.1111111  0.500050  5.075301  0.7217577  3.554253
  0.1111111  0.750025  5.107279  0.7193436  3.554565
  0.1111111  1.000000  5.143995  0.7166455  3.569771
  0.2222222  0.000100  5.069780  0.7216432  3.648863
  0.2222222  0.250075  5.067340  0.7223198  3.577091
  0.2222222  0.500050  5.113531  0.7186313  3.570454
  0.2222222  0.750025  5.166357  0.7145978  3.591641
  0.2222222  1.000000  5.211871  0.7113366  3.614826
  0.3333333  0.000100  5.069538  0.7216694  3.649366
  0.3333333  0.250075  5.082557  0.7211239  3.576992
  0.3333333  0.500050  5.153718  0.7153417  3.592275
  0.3333333  0.750025  5.213432  0.7106409  3.622494
  0.3333333  1.000000  5.282705  0.7050410  3.670128
  0.4444444  0.000100  5.069835  0.7216628  3.649851
  0.4444444  0.250075  5.104429  0.7192687  3.583996
  0.4444444  0.500050  5.181432  0.7130576  3.607718
  0.4444444  0.750025  5.264404  0.7060023  3.662287
  0.4444444  1.000000  5.360504  0.6977822  3.729266
  0.5555556  0.000100  5.069957  0.7216580  3.650036

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1111111 and lambda = 0.250075.
```

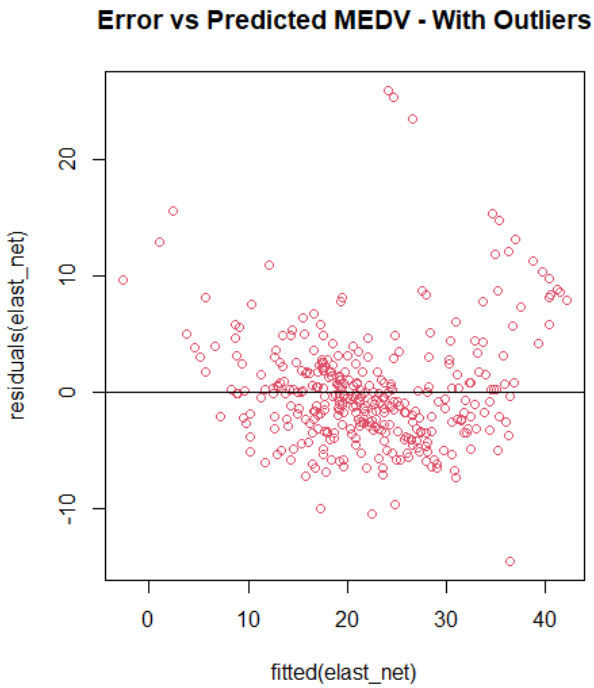
Elastic Net Regression model after removing the outliers is given below:

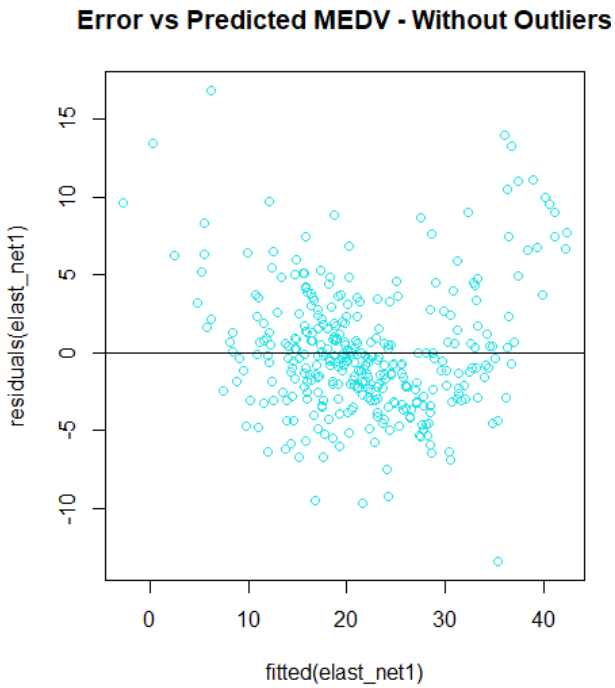
```
> #MODEL AFTER REMOVING OUTLIERS:
>
> error=residuals(elast_net)
> n=length(error)
> MSRes= sum(error^2)/(n-14)
> datall=subset(train,(error/sqrt(MSRes))<3)
> elast_net1<-train (MEDV~.,
+                 data= datall,
+                 method='glmnet',
+                 tunegrid=expand.grid(alpha=0,
+                                      lambda=seq(0.0001,1,length=5)),
+                 trControl=custom)
+ Fold01.Rep1: alpha=0.10, lambda=1.387
- Fold01.Rep1: alpha=0.10, lambda=1.387
+ Fold01.Rep1: alpha=0.55, lambda=1.387
- Fold10.Rep5: alpha=0.55, lambda=1.387
+ Fold10.Rep5: alpha=1.00, lambda=1.387
- Fold10.Rep5: alpha=1.00, lambda=1.387
Aggregating results
Selecting tuning parameters
Fitting alpha = 0.1, lambda = 0.139 on full training set
> elast_net1
glmnet

349 samples
 13 predictor

No pre-processing
Resampling: Cross-Validated (10 fold, repeated 5 times)
Summary of sample sizes: 315, 314, 313, 313, 313, 315, ...
Resampling results across tuning parameters:

  alpha  lambda      RMSE      Rsquared   MAE
  0.10   0.01386975  4.242169  0.7881235  3.169578
  0.10   0.13869752  4.239627  0.7882212  3.154496
  0.10   1.38697516  4.353735  0.7809592  3.203161
  0.55   0.01386975  4.243809  0.7879590  3.171033
  0.55   0.13869752  4.276454  0.7843177  3.160029
  0.55   1.38697516  4.623284  0.7625363  3.427769
  1.00   0.01386975  4.245086  0.7877753  3.169707
  1.00   0.13869752  4.311311  0.7808272  3.172144
  1.00   1.38697516  4.855450  0.7535104  3.571890

RMSE was used to select the optimal model using the smallest value.
The final values used for the model were alpha = 0.1 and lambda = 0.1386975.
```

```
> #Model building using best lambda
> elast_net2=glmnet(x_train,y_train,alpha=0.1,lambda=0.1386975)
> coef(elast_net2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  44.982898153
CRIM         -0.074154356
ZN            0.036300779
INDUS        -0.032019086
CHAS          3.375348527
NOX         -19.649012724
RM            3.198393685
AGE           0.001473376
DIS          -1.669995645
RAD           0.223722407
TAX          -0.007846644
PTRATIO      -1.085803259
B             0.007695768
LSTAT        -0.567369890
```

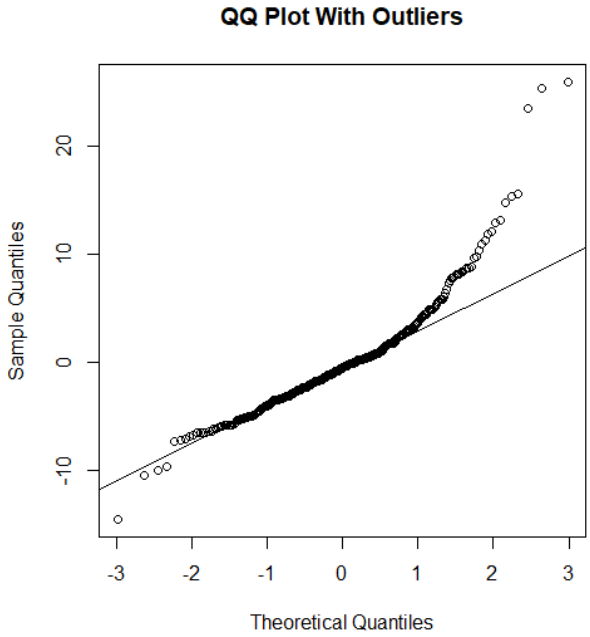
The Residual Plot for error terms before and after removing the outliers can be compared as shown below:

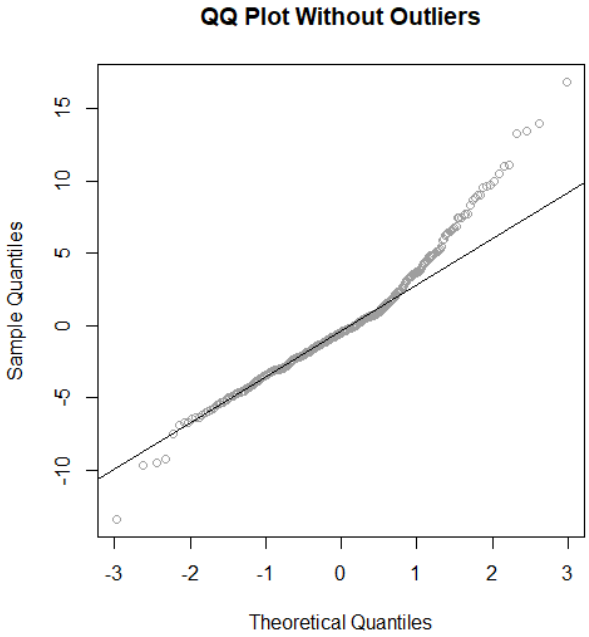
```
> #1. TO CHECK THE NORMALITY OF ERROR
> #Error vs Predicted <- OUTLIERS and HORIZONTAL BAND FASHION
> par(mfrow=c(1,2))
> plot(fitted(elast_net),residuals(elast_net),col=2,main="Error vs Predicted MEDV - With Outliers")
> abline(0,0)
>
> plot(fitted(elast_net1),residuals(elast_net1),col=5,main="Error vs Predicted MEDV - Without Outliers")
> abline(0,0)
>
> #QQ-PLOT of RESIDUALS
> par(mfrow=c(1,2))
>
> qqnorm(residuals(elast_net),col=9,main="QQ Plot With Outliers")
> qqline(residuals(elast_net))
>
> qqnorm(residuals(elast_net1),col=8,main="QQ Plot Without Outliers")
> qqline(residuals(elast_net1))
>
> #NORMALITY OF ERRORS/RESIDUALS
> par(mfrow=c(1,2))
>
> plot(density(residuals(elast_net)),main="Density Plot with Outliers",col=3)
> plot(density(residuals(elast_net1)),main="Density Plot without Outliers",col=5)
```



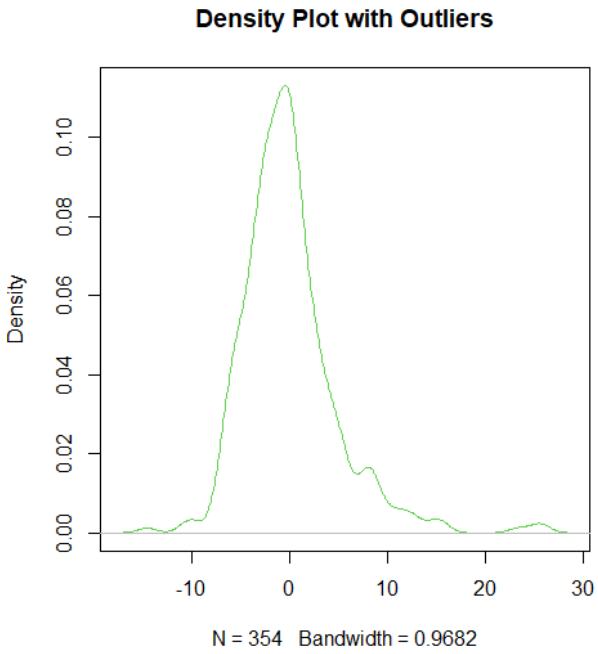


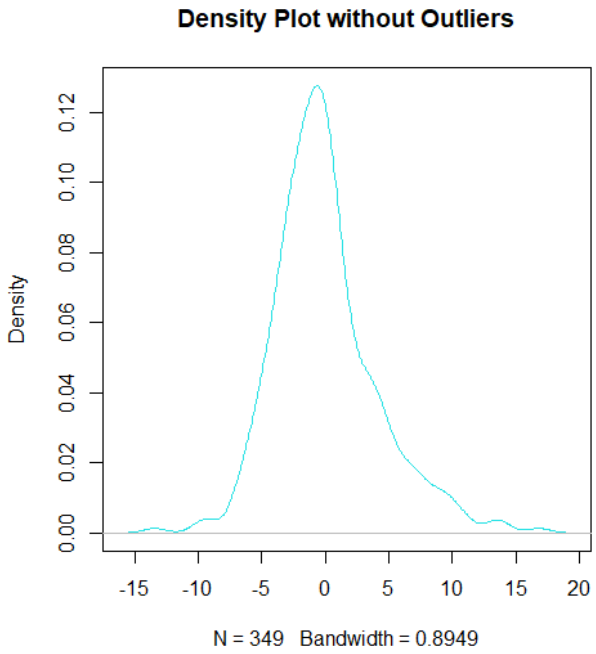
It can be seen that the Residual Plot without outliers follows a horizontal band fashion.





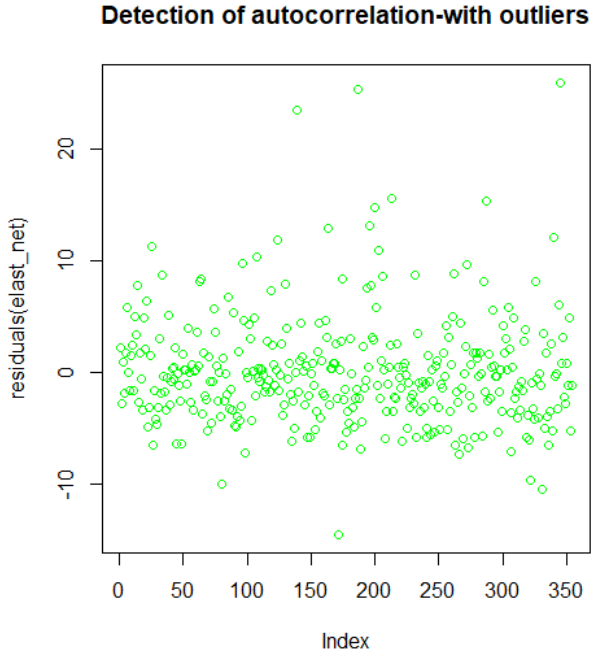
The QQ-Plot with outliers has heavy-tailed distribution, while the QQ-Plot Without outliers follows Normality.

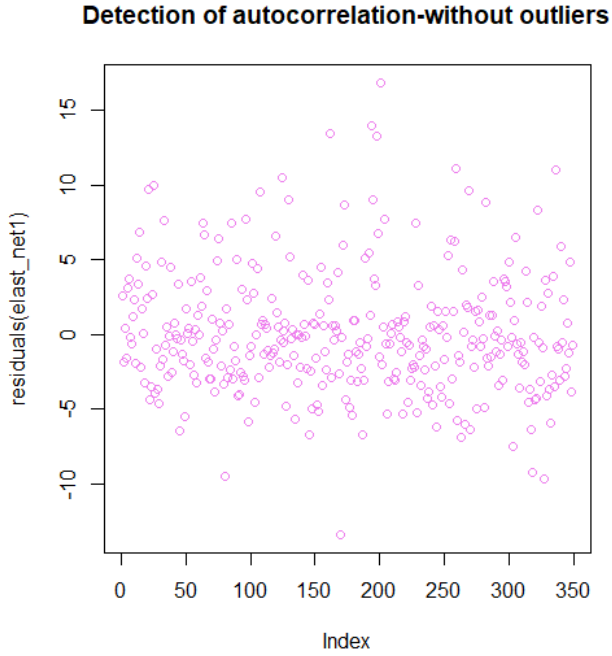




The Density Plot for error with outliers is positively skewed whereas the Density Plot for errors without outliers follows a bell shaped Normal Distribution.

```
> #AUTOCORRELATION
> par(mfrow=c(1,2))
> plot(residuals(elast_net),col="green",main="Detection of autocorrelation-with outliers")
> plot(residuals(elast_net1),col="violet",main="Detection of autocorrelation-without outliers")
```



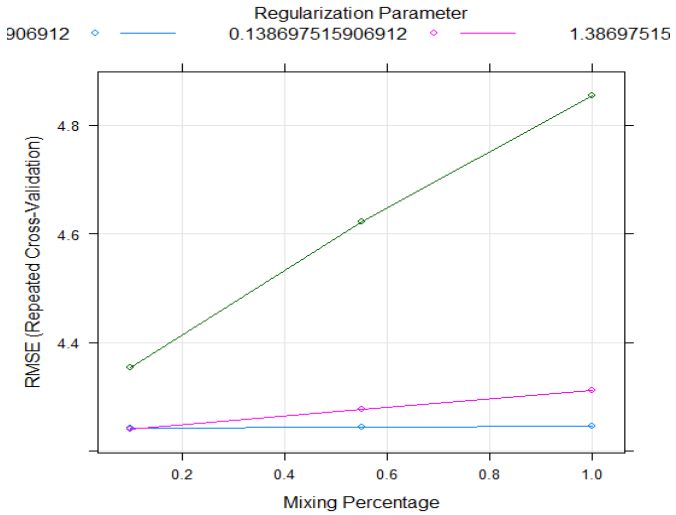


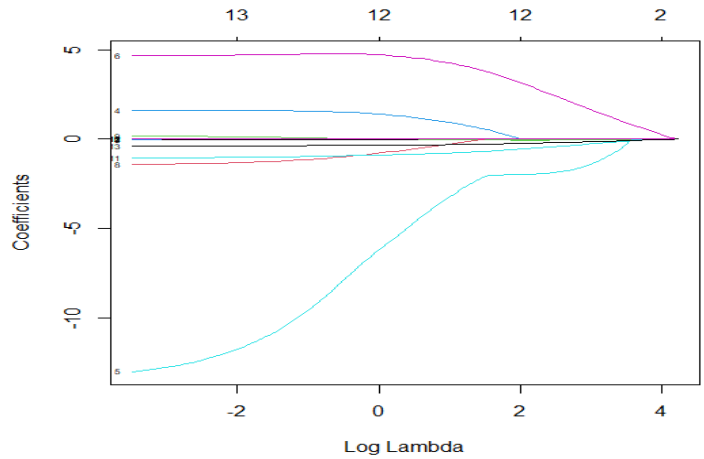
Since the plot of residuals is scattered and doesn't follow any particular pattern, the error terms are independent, that is, there's no autocorrelation.

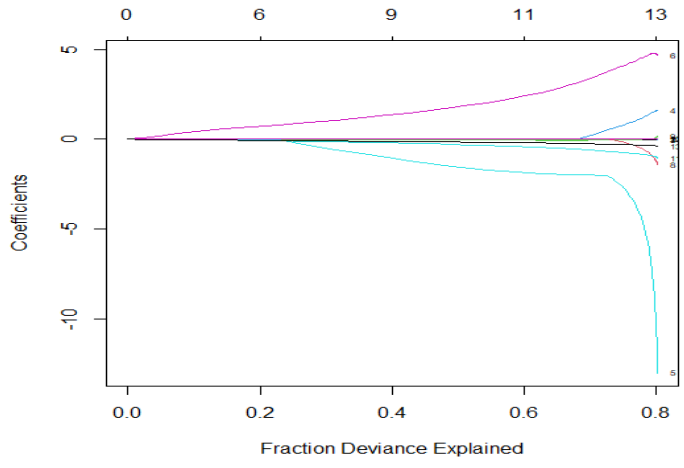
Hence, it can be concluded that Elastic Net Regression (without outliers) is the best fit when compared to Elastic Net Regression (with outliers).

PLOTTING RESULTS

```
> #Plotting results
> plot(elast_net1)
> plot(elast_net1$finalModel, xvar='lambda',label=T)
> plot(elast_net1$finalModel, xvar='dev',label=T)
```







MULTICOLLINEARITY TEST

- **MLRM without Outliers**

```
> imcdiag(model_mlrm1)

Call:
imcdiag(mod = model_mlrm1)


All Individual Multicollinearity Diagnostics Result

             VIF    TOL       Wi       Fi Leamer    CVIF Klein   IND1   IND2
CRIM     1.7455 0.5729  29.0761  31.7872 0.7569 -0.1206     0 0.0147 0.7173
ZN       2.3954 0.4175  54.4197  59.4938 0.6461 -0.1655     0 0.0107 0.9782
INDUS    3.9290 0.2545 114.2299 124.8807 0.5045 -0.2715     0 0.0065 1.2519
CHAS     1.0707 0.9340   2.7572   3.0142 0.9664 -0.0740     0 0.0239 0.1109
NOX      4.4500 0.2247 134.5495 147.0949 0.4740 -0.3075     0 0.0058 1.3019
RM       2.1548 0.4641  45.0389  49.2384 0.6812 -0.1489     0 0.0119 0.9000
AGE      3.3853 0.2954  93.0260 101.6998 0.5435 -0.2340     0 0.0076 1.1832
DIS      4.1053 0.2436 121.1073 132.3994 0.4935 -0.2837     0 0.0062 1.2703
RAD      7.3083 0.1368 246.0234 268.9626 0.3699 -0.5051     1 0.0035 1.4495
TAX      8.5462 0.1170 294.3028 321.7436 0.3421 -0.5906     1 0.0030 1.4828
PTRATIO 1.7599 0.5682  29.6360  32.3993 0.7538 -0.1216     0 0.0146 0.7251
B        1.3392 0.7467  13.2275  14.4608 0.8641 -0.0926     0 0.0191 0.4253
LSTAT    3.5298 0.2833  98.6628 107.8622 0.5323 -0.2439     0 0.0073 1.2036

1 --> COLLINEARITY is detected by the test
0 --> COLLINEARITY is not detected by the test

INDUS , CHAS , AGE , coefficient(s) are non-significant may be due to multicollinearit

R-square of y on all x: 0.8289

* use method argument to check which regressors may be the reason of collinearity
===================================
```

- **MLRM STEP without Outliers**

```
> omcdiag(model_mlrm_step1)

Call:
omcdiag(mod = model_mlrm_step1)


Overall Multicollinearity Diagnostics

                       MC Results detection
Determinant |X'X|:         0.0004         1
Farrar Chi-Square:      2612.5103         1
Red Indicator:             0.4301         0
Sum of Lambda Inverse:    39.8969         0
Theil's Method:           -2.1453         0
Condition Number:         99.4836         1

1 --> COLLINEARITY is detected by the test
0 --> COLLINEARITY is not detected by the test

> imcdiag(model_mlrm_step1)

Call:
imcdiag(mod = model_mlrm_step1)


All Individual Multicollinearity Diagnostics Result

            VIF    TOL       Wi       Fi Leamer    CVIF Klein   IND1   IND2
CRIM     1.9482 0.5133  27.9283  30.8160 0.7164 -0.1467     0 0.0174 0.8265
ZN       2.2537 0.4437  36.9264  40.7445 0.6661 -0.1697     0 0.0151 0.9447
CHAS     1.0665 0.9377   1.9576   2.1600 0.9683 -0.0803     0 0.0318 0.1058
NOX      4.5221 0.2211 103.7433 114.4699 0.4702 -0.3404     0 0.0075 1.3227
RM       2.3625 0.4233  40.1312  44.2806 0.6506 -0.1778     0 0.0144 0.9794
AGE      3.4674 0.2884  72.6748  80.1890 0.5370 -0.2610     0 0.0098 1.2085
DIS      3.7267 0.2683  80.3123  88.6162 0.5180 -0.2805     0 0.0091 1.2425
RAD      6.9610 0.1437 175.5777 193.7316 0.3790 -0.5240     1 0.0049 1.4543
TAX      6.7853 0.1474 170.4029 188.0217 0.3839 -0.5108     1 0.0050 1.4480
PTRATIO  1.7285 0.5786  21.4562  23.6747 0.7606 -0.1301     0 0.0196 0.7157
B        1.4427 0.6931  13.0400  14.3882 0.8325 -0.1086     0 0.0235 0.5211
LSTAT    3.6325 0.2753  77.5400  85.5573 0.5247 -0.2735     0 0.0093 1.2307

1 --> COLLINEARITY is detected by the test
0 --> COLLINEARITY is not detected by the test

* all coefficients have significant t-ratios

R-square of y on all x: 0.8374

* use method argument to check which regressors may be the reason of collinearit
```

From the given multicollinearity diagnosis test and the correlation matrix, it can be seen that the explanatory variables RAD and TAX shows high multicollinearity.

MODEL VALIDATION

The data has been split into train (70%) and test (30%) earlier. Hence, we can predict the test data using the train data now.

1. Multiple Linear Regression Model

```
> #PREDICTING MEDV-TEST using TRAIN
> pred=predict(model_mlrm_train1,test)
> R2=summary(model_mlrm_train)$r.squared
> R2_without_outliers=summary(model_mlrm_train1)$r.squared
> SSR=sum((test$MEDV-pred)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_test_mlrm=1-(SSR/SST)
> RMSE_test_mlrm=sqrt((SSR)/nrow(test))
```

```
> pred
       9       10       17       18       21       23       26       27       38       42       43       52       54       55       56       58
12.586518 18.925313 20.725752 17.013293 12.489236 16.504082 12.990843 15.188219 22.392268 29.098237 25.477678 22.856677 23.747661 15.147572 30.928726 31.220845
      61       62       64       65       66       68       70       73       74       79       80       84       85       86       87       91
16.598782 16.880586 22.962351 24.431183 28.628753 20.763340 20.244062 24.145184 24.235731 21.014046 21.491060 23.862291 24.814223 27.333348 22.181332 26.664864
      97      102      103      115      117      119      121      128      129      134      136      139      142      146      156      159
23.978149 25.396930 18.396152 24.028548 22.839771 19.706459 21.318126 14.856322 19.315815 15.455779 17.857699 14.313565  5.381454 15.017372 21.184290 26.407312
     160      161      165      166      167      174      175      176      177      181      185      192      200      202      209      213
26.229054 30.140248 23.236633 23.479945 37.023341 27.678853 24.863560 30.418663 25.079927 35.124690 20.967541 29.699581 29.873357 27.557663 23.249676 22.366033
     214      215      217      218      219      220      221      222      224      228      231      233      238      244      246      247
25.449243 13.735021 25.828383 27.546961 23.872550 28.182596 31.858736 23.240604 28.168880 31.317704 23.046812 38.411011 32.193329 27.038195 12.918303 19.723252
     250      252      253      254      256      257      258      261      262      267      271      276      278      280      281      284
24.649581 24.808373 25.891482 32.765568 21.191834 36.351138 44.753705 35.083302 37.295904 31.999295 21.822271 32.342492 33.872289 34.317412 38.311931 44.306506
     285      286      288      290      295      301      305      311      313      315      325      326      330      334      351      352
31.788656 27.013166 25.550454 26.494214 23.680504 29.609764 32.926833 17.337686 22.107948 24.972991 24.973546 25.243919 24.968684 22.467767 20.988963 20.790873
     354      366      370      379      381      383      386      389      395      408      409      411      418      421      425      436
24.758173  7.529033 28.514373 15.698479 15.833267 11.913331  7.571681  5.238686 15.978060 16.153158 11.608819 11.015178  5.504688 18.038513 11.062070 13.425174
     443      449      452      454      456      457      461      468      469      471      474      475      476      478      481      482
17.436823 16.370947 18.714963 22.796903 14.646206 10.781073 18.249415 14.784493 15.263868 17.936541 24.335912 12.970984 14.451908  9.373385 20.657781 24.204128
     483      484      485      486      487      489      494      499
25.968024 18.294864 17.570484 20.383941 17.438271 11.594613 20.231873 21.197978
```

```
> R2
[1] 0.7387952
> R2_without_outliers
[1] 0.8374265
> R_square_test_mlrm
[1] 0.7410155
> RMSE_test_mlrm
[1] 4.227935
```

2.Multiple Linear Regression Model – STEP WISE METHOD

```
> pred_step=predict(model_mlrm_step1,test)
> R2=summary(model_mlrm_step)$r.squared
> R2_without_outliers=summary(model_mlrm_step1)$r.squared
> SSR=sum((test$MEDV-pred_step)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_test_step=1-(SSR/SST)
> RMSE_test_step=sqrt((SSR)/nrow(test))
```

```
> pred_step
       9       10       17       18       21       23       26       27       38       42       43       52       54       55       56       58
12.575151 18.941559 20.705729 16.980384 12.437734 16.464486 12.950182 15.157992 22.360465 29.134150 25.506117 22.887402 23.778020 15.079244 30.965362 31.252677
      61       62       64       65       66       68       70       73       74       79       80       84       85       86       87       91
16.608847 16.886061 22.996420 24.449603 28.627194 20.760314 20.240393 24.201486 24.290429 21.058311 21.530371 23.852704 24.792304 27.314143 22.146819 26.595169
      97      102      103      115      117      119      121      128      129      134      136      139      142      146      156      159
23.906294 25.354828 18.330635 23.984004 22.797766 19.651349 21.510363 14.906276 19.376350 15.517985 17.918548 14.357980  5.397027 15.005850 21.176009 26.486973
     160      161      165      166      167      174      175      176      177      181      185      192      200      202      209      213
26.262133 30.207435 23.316639 23.556988 37.129992 27.615997 24.794503 30.366990 25.021511 35.081947 20.896689 29.677491 29.841621 27.521592 23.259433 22.364701
     214      215      217      218      219      220      221      222      224      228      231      233      238      244      246      247
25.481192 13.722644 25.867657 27.618740 23.904480 28.234684 31.820435 23.180186 28.150538 31.305363 23.020528 38.423364 32.196353 27.063994 12.925402 19.749233
     250      252      253      254      256      257      258      261      262      267      271      276      278      280      281      284
24.682501 24.836718 25.943260 32.831133 21.209184 36.392489 44.708800 35.020573 37.239934 31.927337 21.824301 32.364554 33.883665 34.318042 38.331381 44.313462
     285      286      288      290      295      301      305      311      313      315      325      326      330      334      351      352
31.802421 27.003028 25.591262 26.534054 23.784372 29.612571 32.902564 17.306986 22.094829 24.977837 24.975826 25.255374 24.889271 22.474580 20.950727 20.781075
     354      366      370      379      381      383      386      389      395      408      409      411      418      421      425      436
24.854114  7.512847 28.534192 15.703362 15.879805 11.900843  7.547625  5.210799 15.984633 16.160149 11.606891 11.041260  5.485212 18.044241 11.068895 13.411347
     443      449      452      454      456      457      461      468      469      471      474      475      476      478      481      482
17.437100 16.378345 18.725415 22.819415 14.642073 10.770085 18.260308 14.808348 15.302233 17.978341 24.379020 12.994583 14.469124  9.379346 20.722105 24.276701
     483      484      485      486      487      489      494      499
26.045848 18.360915 17.618358 20.444795 17.487136 11.639454 20.176924 21.145453
```

```
> R2
[1] 0.7386736
> R2_without_outliers
[1] 0.8373984
> R_square_test_step
[1] 0.7416991
> RMSE_test_step
[1] 4.222351
```

3. Polynomial Model

```
> #PREDICTING MEDV-TEST using TRAIN
> pred_poly=predict(model_poly1,test)
> R2=summary(model_poly)$r.squared
> R2_without_outliers=summary(model_poly1)$r.squared
> SSR=sum((test$MEDV-pred_poly)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_test_poly=1-(SSR/SST)
> RMSE_poly=sqrt((SSR)/nrow(test))
```

```
> pred
       9       10       17       18       21       23       26       27       38       42       43       52       54       55       56       58
12.586518 18.925313 20.725752 17.013293 12.489236 16.504082 12.990843 15.188219 22.392268 29.098237 25.477678 22.856677 23.747661 15.147572 30.928726 31.220845
      61       62       64       65       66       68       70       73       74       79       80       84       85       86       87       91
16.598782 16.880586 22.962351 24.431183 28.628753 20.763340 20.244062 24.145184 24.235731 21.014046 21.491060 23.862291 24.814223 27.333348 22.181332 26.664864
      97      102      103      115      117      119      121      128      129      134      136      139      142      146      156      159
23.978149 25.396930 18.396152 24.028548 22.839771 19.706459 21.318126 14.856322 19.315815 15.455779 17.857699 14.313565  5.381454 15.017372 21.184290 26.407312
     160      161      165      166      167      174      175      176      177      181      185      192      200      202      209      213
26.229054 30.140248 23.236633 23.479945 37.023341 27.678853 24.863560 30.418663 25.079927 35.124690 20.967541 29.699581 29.873357 27.557663 23.249676 22.366033
     214      215      217      218      219      220      221      222      224      228      231      233      238      244      246      247
25.449243 13.735021 25.828383 27.546961 23.872550 28.182596 31.858736 23.240604 28.168880 31.317704 23.046812 38.411011 32.193329 27.038195 12.918303 19.723252
     250      252      253      254      256      257      258      261      262      267      271      276      278      280      281      284
24.649581 24.808373 25.891482 32.765568 21.191834 36.351138 44.753705 35.083302 37.295904 31.999295 21.822271 32.342492 33.872289 34.317412 38.311931 44.306506
     285      286      288      290      295      301      305      311      313      315      325      326      330      334      351      352
31.788656 27.013166 25.550454 26.494214 23.680504 29.609764 32.926833 17.337686 22.107948 24.972991 24.973546 25.243919 24.968684 22.467767 20.988963 20.790873
     354      366      370      379      381      383      386      389      395      408      409      411      418      421      425      436
24.758173  7.529033 28.514373 15.698479 15.833267 11.913331  7.571681  5.238686 15.978060 16.153158 11.608819 11.015178  5.504688 18.038513 11.062070 13.425174
     443      449      452      454      456      457      461      468      469      471      474      475      476      478      481      482
17.436823 16.370947 18.714963 22.796903 14.646206 10.781073 18.249415 14.784493 15.263868 17.936541 24.335912 12.970984 14.451908  9.373385 20.657781 24.204128
     483      484      485      486      487      489      494      499
25.968024 18.294864 17.570484 20.383941 17.438271 11.594613 20.231873 21.197978
```

```
> R2
[1] 0.9344794
> R2_without_outliers
[1] 0.9667733
> R_square_test_poly
[1] 0.7432177
> RMSE_poly
[1] 4.209921
```

4. Poly-Ridge Model

```
> #PREDICTING MEDV-TEST using TRAIN
> pred_ridge=predict(polyridge1,test)
> SSR=sum((test$MEDV-pred_ridge)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_ridge_test=1-(SSR/SST)
> RMSE_ridge=sqrt((SSR)/nrow(test))
> R_square_ridge_test
[1] 0.8820579
> RMSE_ridge
[1] 2.853158
```

```
> pred_ridge
       9       10       17       18       21       23       26       27       38       42       43       52       54       55       56       58
12.963832 15.135276 20.011312 15.898537 13.168908 15.503020 13.485726 14.048818 22.892169 30.385190 24.986199 20.749307 22.025231 13.554048 33.702122 30.828410
      61       62       64       65       66       68       70       73       74       79       80       84       85       86       87       91
16.597387 15.828771 24.494280 29.560170 26.485782 19.862099 19.319909 23.170167 23.318625 19.749517 20.701714 22.345771 25.387905 27.786490 23.467091 26.343018
      97      102      103      115      117      119      121      128      129      134      136      139      142      146      156      159
23.409378 25.114652 19.307642 22.578321 21.337533 18.758220 21.393633 15.557309 17.908675 15.676305 15.938964 13.940234 13.358016 16.900915 13.419609 29.696793
     160      161      165      166      167      174      175      176      177      181      185      192      200      202      209      213
28.692008 37.067225 22.525078 24.193204 44.242903 26.828507 24.445526 32.632764 25.534371 39.510800 20.275285 28.338144 32.077478 23.327152 19.854006 19.189964
     214      215      217      218      219      220      221      222      224      228      231      233      238      244      246      247
25.356152 29.106478 19.944527 27.062729 19.994372 27.316795 29.889615 19.947429 28.435090 32.927032 23.038159 45.530041 34.901884 25.940549 14.316350 20.083006
     250      252      253      254      256      257      258      261      262      267      271      276      278      280      281      284
25.656124 25.151375 28.927035 40.293757 17.586948 38.964457 54.457453 35.142190 38.401721 30.551526 19.725824 32.362706 31.420519 36.396204 43.957408 51.981892
     285      286      288      290      295      301      305      311      313      315      325      326      330      334      351      352
31.616617 24.804642 22.556518 23.909048 22.480837 28.242009 35.216863 18.133435 20.043680 23.053063 25.284411 25.539246 24.237723 23.390995 21.380745 24.550333
     354      366      370      379      381      383      386      389      395      408      409      411      418      421      425      436
30.440282 35.824141 45.645297  8.343140 13.088922 11.974669  8.252281  9.607076 15.772460 21.713939 12.173972 19.841241  7.878884 15.998725 15.646655  8.535361
     443      449      452      454      456      457      461      468      469      471      474      475      476      478      481      482
16.477468 13.936409 14.922784 15.449163 13.531095 13.498127 15.440257 14.349073 14.461413 18.556038 23.275621 17.040692 12.088773 10.911263 24.146484 28.587484
     483      484      485      486      487      489      494      499
30.018578 23.709294 20.256940 23.128335 19.040605 15.061566 21.203894 21.210951
```

5. Poly-Lasso Model

```
> #TEST DATA
> pred_lasso=predict(polylassol,test)
> head(pred_lasso)
        9        10        17        18        21        23
12.07515 16.15337 19.69475 16.49876 13.55752 16.04290
> SSR=sum((test$MEDV-pred_lasso)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_lasso=1-(SSR/SST)
> RMSE_lasso=sqrt((SSR)/nrow(test))
> R_square_lasso
[1] 0.8420374
> RMSE_lasso
[1] 3.301935
```

```
> pred_lasso
        9        10        17        18        21        23        26        27        38        42        43        52        54        55        56        58
12.075150 16.153370 19.694753 16.498755 13.557518 16.042899 13.526519 14.713809 22.477694 31.664446 24.357806 20.695825 21.667186 20.109045 31.354646 28.075401
       61        62        64        65        66        68        70        73        74        79        80        84        85        86        87        91
16.215376 15.793570 24.696446 30.163118 25.968310 19.562202 19.204743 21.468950 22.405280 19.530327 20.084714 22.171456 25.595950 27.728422 23.793501 26.291394
       97       102       103       115       117       119       121       128       129       134       136       139       142       146       156       159
23.444005 25.769890 16.374798 21.742049 20.764837 17.937168 20.876517 15.769170 18.154628 16.810540 16.523241 13.840122 12.831111 20.285818 17.902922 29.654015
      160       161       165       166       167       174       175       176       177       181       185       192       200       202       209       213
29.511677 41.424089 22.435748 24.751564 45.412180 26.298059 24.083228 32.756616 25.404789 40.591175 20.725709 27.510521 33.293619 23.443976 21.227730 21.411537
      214       215       217       218       219       220       221       222       224       228       231       233       238       244       246       247
25.667092 32.897769 24.415683 28.202391 22.475832 29.302079 28.861537 20.333989 27.830684 32.262295 22.403502 45.607957 34.585910 24.954781 15.049791 20.304408
      250       252       253       254       256       257       258       261       262       267       271       276       278       280       281       284
26.324678 24.315052 30.170372 44.764300 17.706847 38.528187 56.135590 35.979788 38.660143 31.790104 19.758966 32.258883 31.893038 36.951838 44.472567 52.445384
      285       286       288       290       295       301       305       311       313       315       325       326       330       334       351       352
32.811432 24.135842 21.399551 23.609210 22.103846 26.682369 35.797323 15.788570 19.545018 22.947299 25.429220 25.980001 23.333595 23.381075 22.634746 26.273313
      354       366       370       379       381       383       386       389       395       408       409       411       418       421       425       436
27.776493 36.819187 48.092702  9.429078 26.783131 11.590727  8.194188  9.568511 15.018863 21.559395 12.198267 18.850738  7.508614 15.253169 14.121172  9.147250
      443       449       452       454       456       457       461       468       469       471       474       475       476       478       481       482
15.882536 13.557916 14.643426 14.884532 13.883222 13.907193 16.007914 14.336316 16.669742 18.725126 22.731519 18.807308 11.637945 12.230308 24.518137 28.939331
      483       484       485       486       487       489       494       499
30.131639 23.032109 19.837346 23.314677 19.910676 15.926174 21.410664 21.593128
```

6. Poly-Elastic Net Model

```
#TEST DATA
pred_elastic=predict(polyelast_net1,test)
SSR=sum((test$MEDV-pred_elastic)^2)
SST=sum((test$MEDV-mean(test$MEDV))^2)
R_square_elastic=1-(SSR/SST)
RMSE_elastic=sqrt((SSR)/nrow(test))
R_square_elastic
RMSE_elastic
```

```
> pred_elastic
       9       10       17       18       21       23       26       27       38       42       43       52       54       55       56       58
12.336630 15.253519 19.608820 16.101723 13.369984 15.881018 13.272502 14.243772 22.750567 30.543941 24.617437 20.627693 21.842820 15.483217 32.898070 29.501742
      61       62       64       65       66       68       70       73       74       79       80       84       85       86       87       91
16.534898 15.824424 24.800430 30.315966 26.096051 19.645667 19.141453 22.521725 22.844349 19.675105 20.406594 22.082065 25.450129 27.641669 23.654864 26.300019
      97      102      103      115      117      119      121      128      129      134      136      139      142      146      156      159
23.333528 25.073926 17.746071 22.223570 21.177201 18.446964 21.178981 15.597537 17.917684 15.998765 16.065957 13.956425 13.653751 18.160237 14.522189 29.683252
     160      161      165      166      167      174      175      176      177      181      185      192      200      202      209      213
29.135155 38.339923 22.514866 24.434600 44.792113 26.602394 24.213010 32.793792 25.598193 40.102197 20.198232 28.076988 32.486871 23.066978 20.267529 19.727297
     214      215      217      218      219      220      221      222      224      228      231      233      238      244      246      247
25.349849 31.390012 20.803378 27.301021 20.402430 27.918419 29.583637 20.152225 28.255296 32.744114 22.815569 45.666681 34.925647 25.507257 14.360255 20.131598
     250      252      253      254      256      257      258      261      262      267      271      276      278      280      281      284
25.874090 24.757352 29.471256 41.530800 17.644367 38.656546 55.351269 35.609616 38.546811 31.331011 19.696027 32.045966 31.576700 36.391500 43.980530 52.049240
     285      286      288      290      295      301      305      311      313      315      325      326      330      334      351      352
32.324121 24.520422 22.052296 23.772202 22.477550 27.538426 35.661879 17.345699 19.803997 23.006013 25.219001 25.458118 24.034461 23.469942 21.729920 25.266556
     354      366      370      379      381      383      386      389      395      408      409      411      418      421      425      436
30.276501 36.912924 46.670620  8.589339 15.961224 11.801494  8.271668  9.460097 15.607750 21.811991 12.134571 20.169631  7.756543 15.721499 15.232573  8.494667
     443      449      452      454      456      457      461      468      469      471      474      475      476      478      481      482
16.229648 13.708085 14.663014 15.018612 13.478634 13.527789 15.360188 14.174077 15.021831 18.582298 23.118298 17.356721 11.831977 11.020280 24.472699 28.911604
     483      484      485      486      487      489      494      499
30.265554 23.919268 20.279427 23.391984 19.311580 15.156635 21.331676 21.426378
```

7. Ridge Regression Model:

```
> #PREDICTING MEDV-TEST using TRAIN
> pred_ridge1=predict(ridge1,test)
> SSR=sum((test$MEDV-pred_ridge1)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_ridge_test1=1-(SSR/SST)
> RMSE_ridge1=sqrt((SSR)/nrow(test))
> R_square_ridge_test1
[1] 0.7333985
> RMSE_ridge1
[1] 4.289659
> pred_ridge1
        9        10        17        18        21        23        26        27
13.304360 19.538153 20.643970 17.026627 12.790600 16.425302 13.427477 15.314895
       38        42        43        52        54        55        56        58
22.815653 29.085789 25.798087 23.408695 24.177795 15.046515 30.339284 31.410369
       61        62        64        65        66        68        70        73
16.733745 17.093140 22.415631 24.117447 29.131505 21.160705 20.679806 24.390122
       74        79        80        84        85        86        87        91
24.384049 21.301784 21.976851 24.240683 25.186778 27.716864 22.728673 27.360919
       97       102       103       115       117       119       121       128
24.873273 25.262936 19.450862 24.653043 23.359484 20.524339 21.133829 14.681340
      129       134       136       139       142       146       156       159
18.769738 15.182769 17.314331 14.002119  5.356637 15.429998 21.116296 27.479896
      160       161       165       166       167       174       175       176
26.161147 30.280817 24.127935 24.761782 37.056445 28.571318 25.976304 31.026001
      177       181       185       192       200       202       209       213
25.903715 35.181598 22.103358 30.222931 29.786953 28.455343 22.706005 21.951438
      214       215       217       218       219       220       221       222
25.572569 14.080851 25.508438 27.837638 23.633215 27.792827 31.400746 22.991665
      224       228       231       233       238       244       246       247
28.689890 31.643239 23.802884 38.022345 32.241176 27.355367 13.195035 19.712008
      250       252       253       254       256       257       258       261
24.281454 24.699140 25.297574 31.449138 21.560353 36.331919 44.637987 35.636109
      262       267       271       276       278       280       281       284
37.762020 32.608491 22.243501 32.547553 33.094534 35.038010 38.613974 43.565195
      285       286       288       290       295       301       305       311
31.848819 27.655345 25.922346 26.667138 24.312236 30.062034 32.759887 18.411489
      313       315       325       326       330       334       351       352
22.640424 25.158887 24.972462 25.063183 25.649430 22.196733 20.834705 20.626889
      354       366       370       379       381       383       386       389
24.249743  8.724628 27.542841 15.691764 17.372831 11.875781  7.766366  5.645763
      395       408       409       411       418       421       425       436
15.963782 16.702017 12.006389 13.734204  6.795392 17.866081 12.442956 13.482195
      443       449       452       454       456       457       461       468
16.882202 15.957124 18.056726 21.766088 14.861002 11.385390 17.800008 14.837582
```

8. Lasso Regression Model:

```
> #PREDICTING MEDV USING TRAIN DATA
> pred_lassol=predict(lassol,test)
> head(pred_lassol)
       9       10       17       18       21       23
13.30436 19.53815 20.64397 17.02663 12.79060 16.42530
> SSR=sum((test$MEDV-pred_lassol)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_test_lassol=1-(SSR/SST)
> R_square_test_lassol
[1] 0.7333985
> RMSE_test_lassol=sqrt((SSR)/nrow(test))
> RMSE_test_lassol
[1] 4.289659

> pred_lassol
       9       10       17       18       21       23       26       27
13.304360 19.538153 20.643970 17.026627 12.790600 16.425302 13.427477 15.314895
      38       42       43       52       54       55       56       58
22.815653 29.085789 25.798087 23.408695 24.177795 15.046515 30.339284 31.410369
      61       62       64       65       66       68       70       73
16.733745 17.093140 22.415631 24.117447 29.131505 21.160705 20.679806 24.390122
      74       79       80       84       85       86       87       91
24.384049 21.301784 21.976851 24.240683 25.186778 27.716864 22.728673 27.360919
      97      102      103      115      117      119      121      128
24.873273 25.262936 19.450862 24.653043 23.359484 20.524339 21.133829 14.681340
     129      134      136      139      142      146      156      159
18.769738 15.182769 17.314331 14.002119  5.356637 15.429998 21.116296 27.479896
     160      161      165      166      167      174      175      176
26.161147 30.280817 24.127935 24.761782 37.056445 28.571318 25.976304 31.026001
     177      181      185      192      200      202      209      213
25.903715 35.181598 22.103358 30.222931 29.786953 28.455343 22.706005 21.951438
     214      215      217      218      219      220      221      222
25.572569 14.080851 25.508438 27.837638 23.633215 27.792827 31.400746 22.991665
     224      228      231      233      238      244      246      247
```

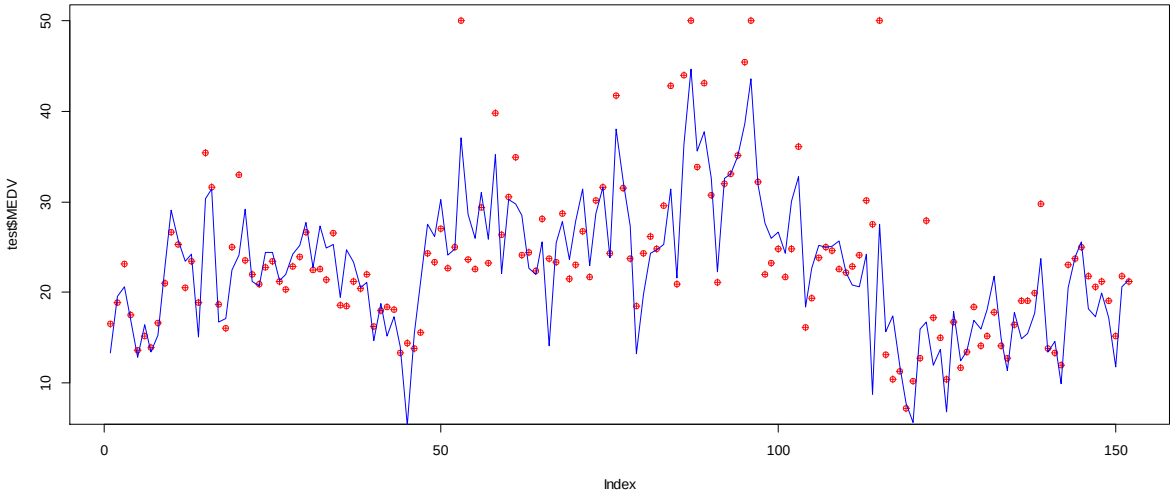
9. Elastic Net Regression:

```
> #PREDICTING MEDV USING TEST DATA
>
> pred_elast_net1=predict(elast_net1,test)
> SSR=sum((test$MEDV-pred_elast_net1)^2)
> SST=sum((test$MEDV-mean(test$MEDV))^2)
> R_square_test_elast1=1-(SSR/SST)
> R_square_test_elast1
[1] 0.7333985
> RMSE_test_elast1=sqrt((SSR)/nrow(test))
> RMSE_test_elast1
[1] 4.289659
pred_elast_net1
       9       10       17       18       21       23       26       27       38
3.304360 19.538153 20.643970 17.026627 12.790600 16.425302 13.427477 15.314895 22.815653
      42       43       52       54       55       56       58       61       62
9.085789 25.798087 23.408695 24.177795 15.046515 30.339284 31.410369 16.733745 17.093140
      64       65       66       68       70       73       74       79       80
2.415631 24.117447 29.131505 21.160705 20.679806 24.390122 24.384049 21.301784 21.976851
      84       85       86       87       91       97      102      103      115
4.240683 25.186778 27.716864 22.728673 27.360919 24.873273 25.262936 19.450862 24.653043
     117      119      121      128      129      134      136      139      142
3.359484 20.524339 21.133829 14.681340 18.769738 15.182769 17.314331 14.002119  5.356637
     146      156      159      160      161      165      166      167      174
5.429998 21.116296 27.479896 26.161147 30.280817 24.127935 24.761782 37.056445 28.571318
     175      176      177      181      185      192      200      202      209
5.976304 31.026001 25.903715 35.181598 22.103358 30.222931 29.786953 28.455343 22.706005
     213      214      215      217      218      219      220      221      222
1.951438 25.572569 14.080851 25.508438 27.837638 23.633215 27.792827 31.400746 22.991665
     224      228      231      233      238      244      246      247      250
8.689890 31.643239 23.802884 38.022345 32.241176 27.355367 13.195035 19.712008 24.281454
     252      253      254      256      257      258      261      262      267
4.699140 25.297574 31.449138 21.560353 36.331919 44.637987 35.636109 37.762020 32.608491
     271      276      278      280      281      284      285      286      288
2.243501 32.547553 33.094534 35.038010 38.613974 43.565195 31.848819 27.655345 25.922346
     290      295      301      305      311      313      315      325      326
6.667138 24.312236 30.062034 32.759887 18.411489 22.640424 25.158887 24.972462 25.063183
     330      334      351      352      354      366      370      379      381
```

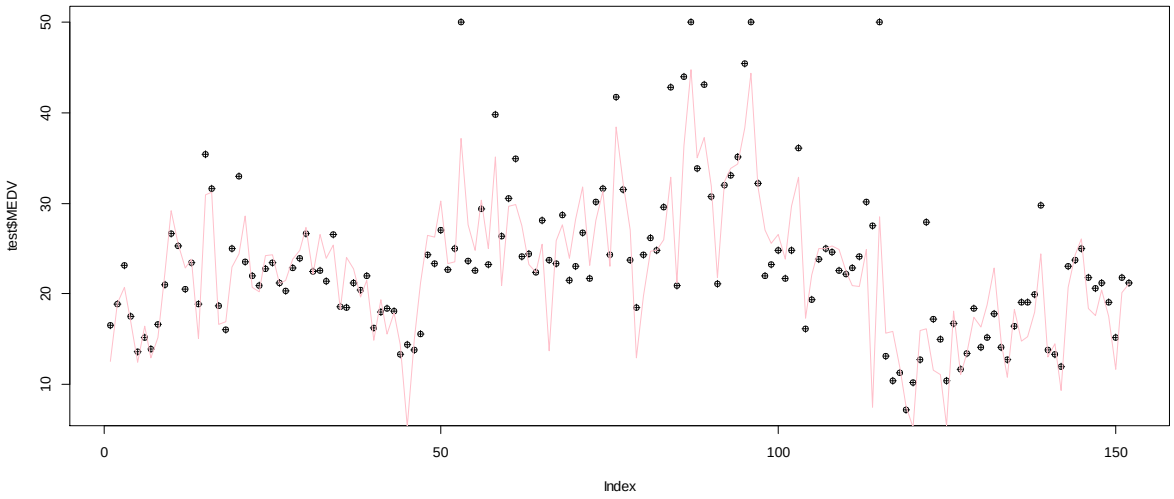
CONCLUSION

```
> #COMPARING PREDICTED vs ACTUAL VALUES
> plot(test$MEDV,pch=10,col="red",main="MULTIPLE LINEAR REGRESSION MODEL")
> lines(pred,col="blue")
>
> plot(test$MEDV,pch=10,col="black",main="MULTIPLE LINEAR REGRESSION MODEL - STEP METHOD")
> lines(pred_step,col="pink")
>
> plot(test$MEDV,pch=10,col="blue",main="POLYNOMIAL REGRESSION MODEL")
> lines(pred_poly,col="orange")
>
> plot(test$MEDV,pch=10,col="red",main="RIDGE REGRESSION MODEL")
> lines(pred_ridge1,col="purple")
>
>
> plot(test$MEDV,pch=10,col="red",main="LASSO REGRESSION MODEL")
> lines(pred_lasso1,col="dark blue")
>
> plot(test$MEDV,pch=10,col="red",main="ELASTIC NET REGRESSION MODEL")
> lines(pred_elastic1,col="dark green")
>
> plot(test$MEDV,pch=10,col="purple",main="POLY RIDGE REGRESSION MODEL")
> lines(pred_ridge,col="black")
>
> plot(test$MEDV,pch=10,col="green",main="POLY LASSO REGRESSION MODEL")
> lines(pred_lasso,col="blue")
>
> plot(test$MEDV,pch=10,col="blue",main="POLY ELASTIC NET REGRESSION MODEL")
> lines(pred_elastic,col="red")
```

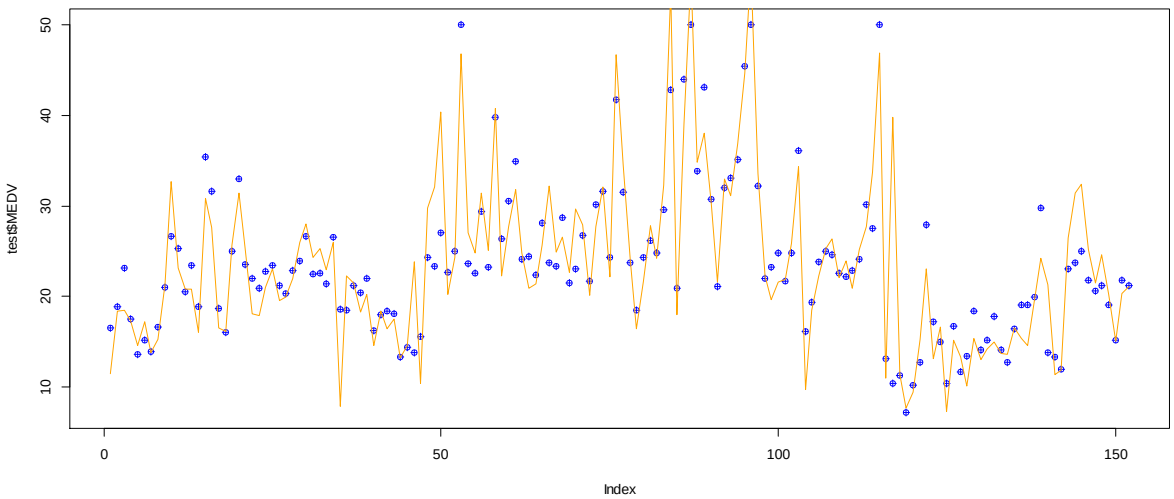
MULTIPLE LINEAR REGRESSION MODEL



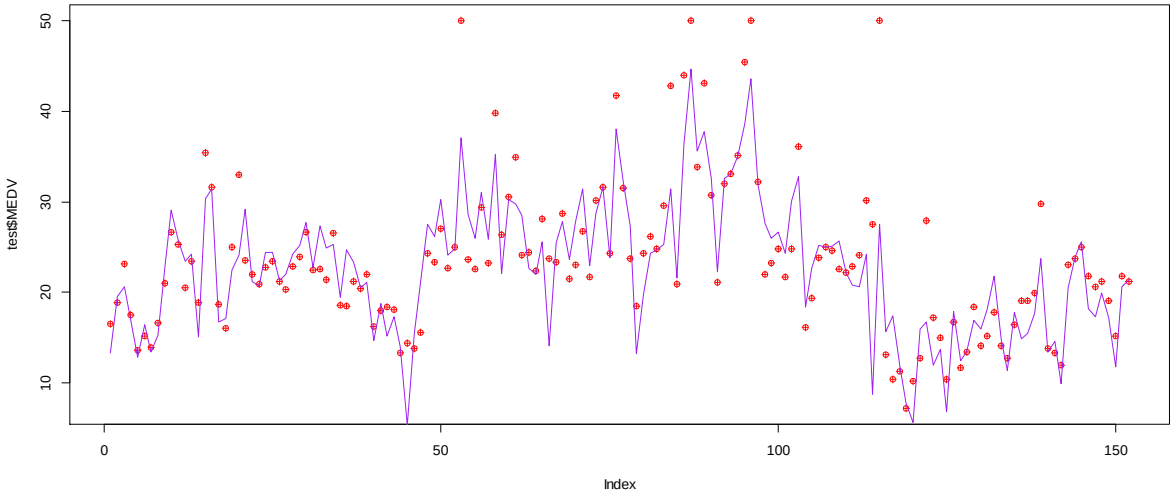
MULTIPLE LINEAR REGRESSION MODEL - STEP METHOD



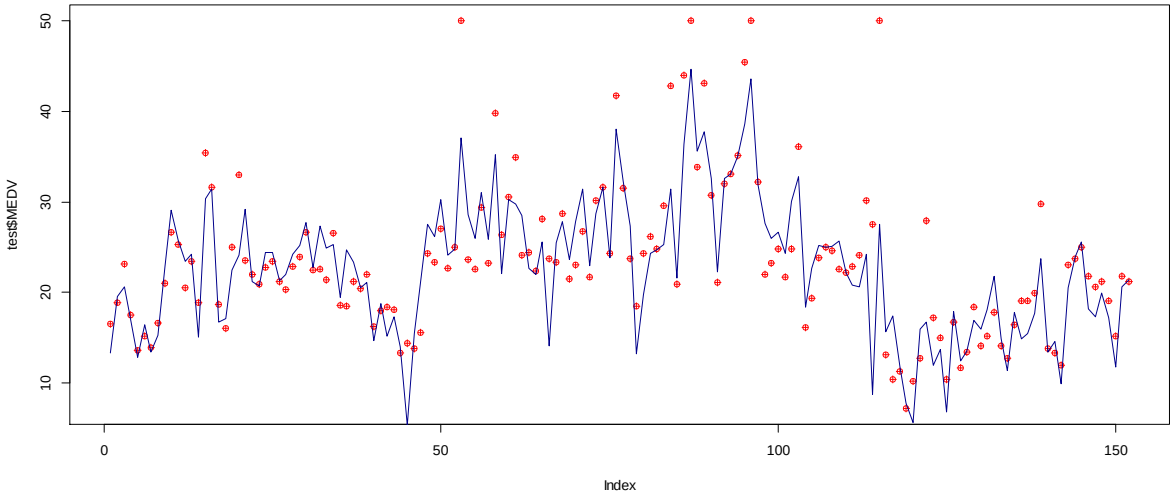
POLYNOMIAL REGRESSION MODEL



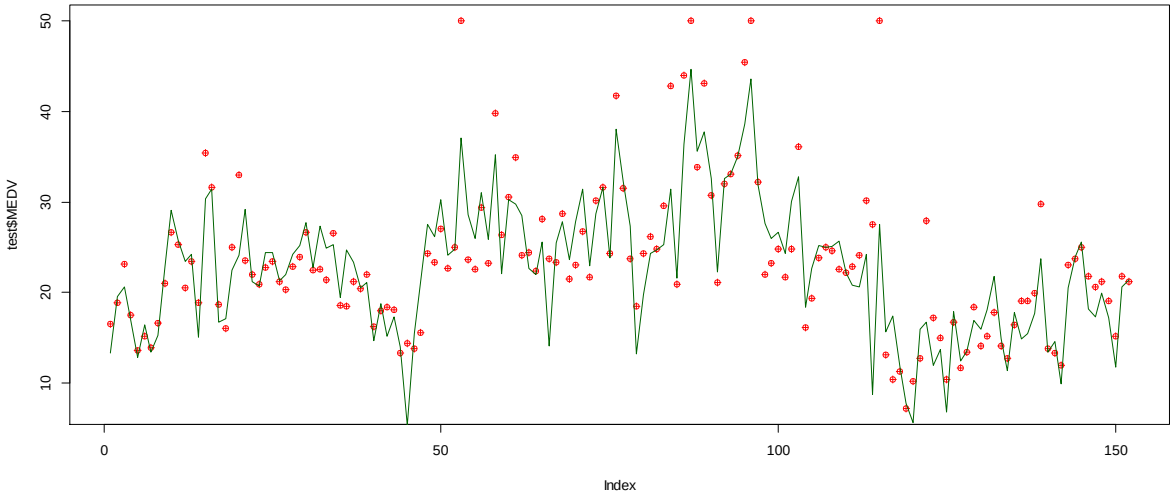
RIDGE REGRESSION MODEL



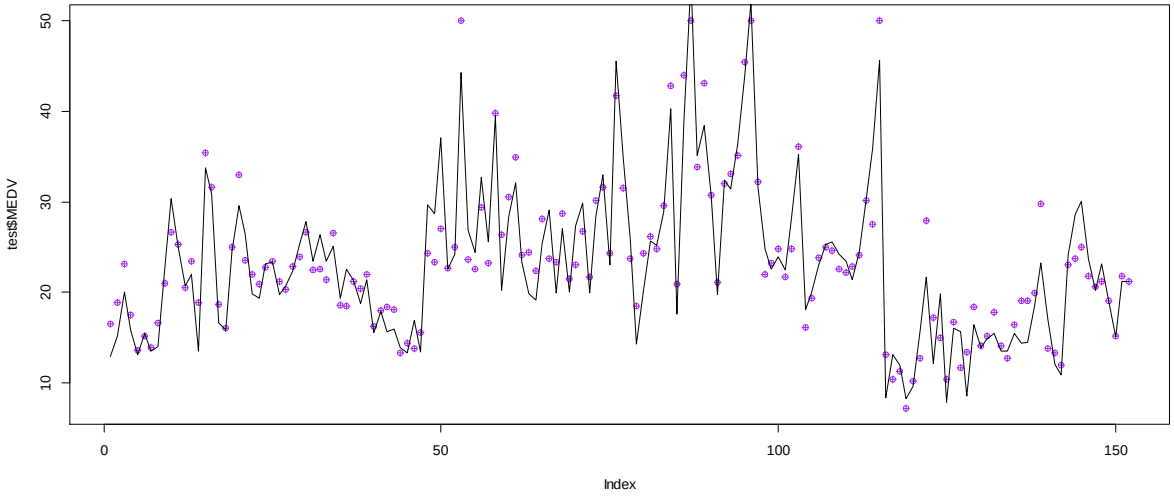
LASSO REGRESSION MODEL



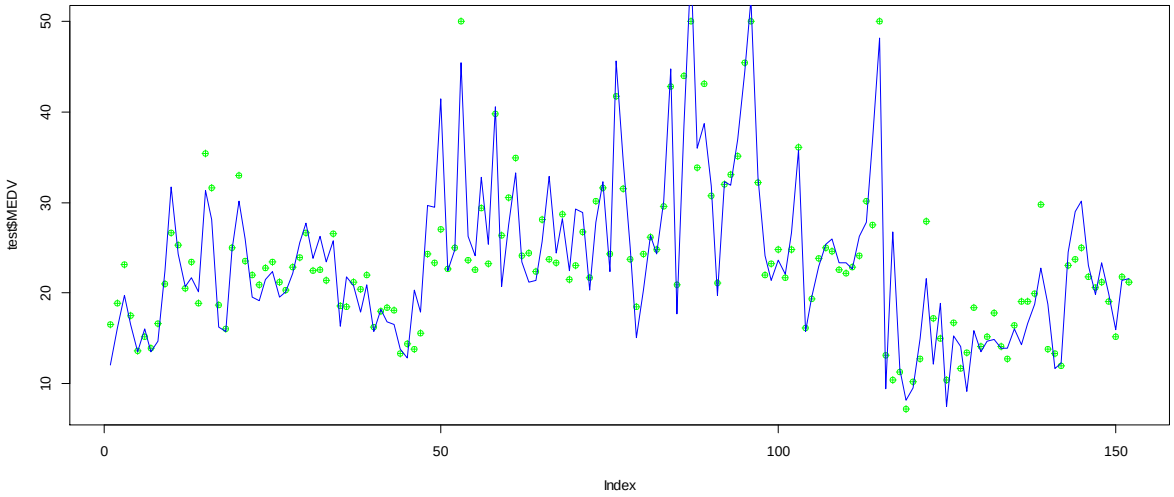
ELASTIC NET REGRESSION MODEL



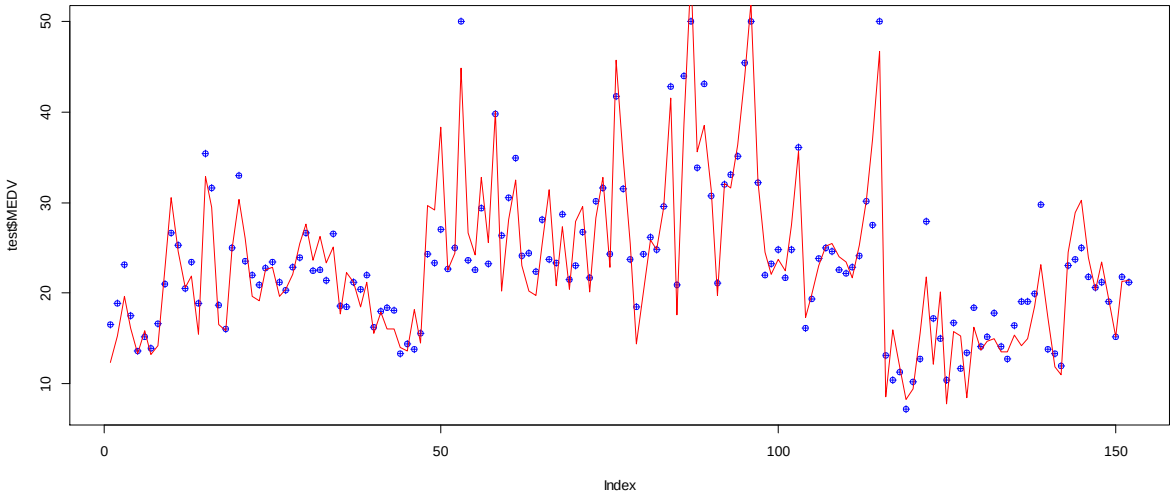
POLY RIDGE REGRESSION MODEL



POLY LASSO REGRESSION MODEL



POLY ELASTIC NET REGRESSION MODEL



Comparing Regularization Regression Model:

```
> #COMPARING MODELS - RIDGE vs LASSO vs ELASTIC NET
> model_list <- list(Ridge=ridgel,Lasso=lassol,ElasticNet=elast_netl,Poly_Ridge=polyridgel,Poly_Lasso=polylassol,Poly_ElasticNet=polyelast_netl)
> res<-resamples(model_list)
> summary(res)

Call:
summary.resamples(object = res)

Models: Ridge, Lasso, ElasticNet, Poly_Ridge, Poly_Lasso, Poly_ElasticNet
Number of resamples: 50

MAE
                    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
Ridge           2.371782 2.880021 3.199581 3.165388 3.390150 4.215969    0
Lasso           2.132617 2.834116 3.149270 3.149913 3.400595 4.638100    0
ElasticNet      1.765344 2.869136 3.146319 3.154496 3.447496 4.254500    0
Poly_Ridge      1.436911 2.048380 2.368423 2.373484 2.610142 3.511196    0
Poly_Lasso      1.644460 2.059360 2.401591 2.422392 2.672407 4.003886    0
Poly_ElasticNet 1.610664 2.110567 2.342003 2.371678 2.680612 3.476453    0

RMSE
                    Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
Ridge           3.126513 3.798600 4.259754 4.266912 4.615252 5.734090    0
Lasso           2.680263 3.758394 4.137121 4.213297 4.661620 6.789730    0
ElasticNet      2.358261 3.753877 4.235759 4.239627 4.694807 5.571404    0
Poly_Ridge      1.967115 2.648817 3.043898 3.354775 3.997404 6.354375    0
Poly_Lasso      2.130399 2.702183 3.257869 3.451000 3.746557 7.199353    0
Poly_ElasticNet 2.018851 2.776942 3.273610 3.379455 3.923725 5.496252    0

Rsquared
                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max. NA's
Ridge           0.6453977 0.7413499 0.7981667 0.7881790 0.8376325 0.8969088    0
Lasso           0.5658806 0.7549938 0.8096524 0.7914296 0.8480875 0.9029806    0
ElasticNet      0.4970317 0.7583093 0.8003982 0.7882212 0.8376582 0.9241802    0
Poly_Ridge      0.6719116 0.8249746 0.8907999 0.8707924 0.9227850 0.9644756    0
Poly_Lasso      0.4535397 0.8282174 0.8839761 0.8624330 0.9243330 0.9537353    0
Poly_ElasticNet 0.7440782 0.8427262 0.8750770 0.8716965 0.9139368 0.9601423    0
```

Comparing Train Data

```
> F_MLRM=summary(model_mlrm1)$fstatistic[1]
> F_MLRM_STEP=summary(model_mlrm_step1)$fstatistic[1]
> F_POLY=summary(model_poly1)$fstatistic[1]
> F_RIDGE=''
> F_LASSO=''
> F_ELASTIC=''
> F_POLY_RIDGE=''
> F_POLY_LASSO=''
> F_POLY_ELASTIC=''
>
> Model=c("MLRM","MLRM - Step Wise", "Polynomial", "Ridge", "Lasso", "Elastic-Net","Poly-Ridge", "Poly-Lasso","Poly-Elastic Net")
> R_Square = c(R_MLRM,R_MLRM_STEP,R_POLY,R_RIDGE,R_LASSO,R_ELASTIC,R_POLY_RIDGE,R_POLY_LASSO,R_POLY_ELASTIC)
> Adj_R_Square = c(ADJ_R_MLRM,ADJ_R_MLRM_STEP,ADJ_R_POLY,ADJ_R_RIDGE,ADJ_R_LASSO,ADJ_R_ELASTIC,ADJ_R_POLY_RIDGE,ADJ_R_POLY_LASSO,ADJ_R_POLY_ELASTIC)
> RMSE = c(RMSE_MLRM,RMSE_MLRM_STEP,RMSE_POLY,RMSE_RIDGE,RMSE_LASSO,RMSE_ELASTIC,RMSE_POLY_RIDGE,RMSE_POLY_LASSO,RMSE_POLY_ELASTIC)
> F_Statistic=c(F_MLRM,F_MLRM_STEP,F_POLY,F_RIDGE,F_LASSO,F_ELASTIC,F_POLY_RIDGE,F_POLY_LASSO,F_POLY_ELASTIC)
>
> #Table-TRAIN
> train_table=data.frame(Model,R_Square,Adj_R_Square,RMSE,F_Statistic)
> comparison_train=list()
> comparison_train[[1]] = "Comparison of Different Models for Train Data"
> comparison_train[[2]]=train_table
```

```
> comparison_train
[[1]]
[1] "Comparison of Different Models for Train Data"

[[2]]
             Model  R_Square      Adj_R_Square     RMSE      F_Statistic
1             MLRM 0.8288663 0.824102375930295 3.341323 173.989234959232
2 MLRM - Step Wise 0.8373984 0.831357493371114 3.457026 138.620879143773
3       Polynomial 0.9667733 0.952085084297712 2.050677 65.8197102943855
4            Ridge 0.7881790                   4.266912
5            Lasso 0.7914296                   4.213297
6      Elastic-Net 0.7882212                   4.239627
7       Poly-Ridge 0.8707138                   3.351363
8       Poly-Lasso 0.8624330                   3.738446
9 Poly-Elastic Net 0.8716965                   3.379455
```

Since, we can see that MLRM,MLRM-Step Wise and Polynomial has the issue of overfitting, i.e, the training data has good performance while there's a poor generalization on the test data, we go for regularization of the Model.

Comparing Test Data

```
> #CONCLUSION - TEST DATA
>
> Model_Test=c("MLRM","MLRM - Step Wise", "Polynomial", "Ridge", "Lasso", "Elastic-Net","Poly-Ridge", "Poly-Lasso","Poly-Elastic Net")
> R_Square_test= c(R_square_test_mlrm,R_square_test_step,R_square_test_poly,R_square_ridge_test1,
+                  R_square_test_lasso1,R_square_test_elast1,R_square_ridge_test,R_square_lasso,R_square_elastic)
> RMSE_test= c(RMSE_test_mlrm,RMSE_test_step,RMSE_poly,RMSE_ridge1,RMSE_test_lasso1,RMSE_test_elast1,RMSE_ridge,RMSE_lasso,RMSE_elastic)
>
> #Table-TRAIN
> test_table=data.frame(Model_Test,R_Square_test,RMSE_test)
> comparison_test=list()
> comparison_test[[1]] = "Comparison of Different Models for Test Data"
> comparison_test[[2]]=test_table
```

```
> comparison_test
[[1]]
[1] "Comparison of Different Models for Test Data"

[[2]]
        Model_Test R_Square_test RMSE_test
1             MLRM     0.7410155  4.227935
2 MLRM - Step Wise     0.7416991  4.222351
3       Polynomial     0.7432177  4.209921
4            Ridge     0.7333985  4.289659
5            Lasso     0.7333985  4.289659
6      Elastic-Net     0.7333985  4.289659
7       Poly-Ridge     0.8820579  2.853158
8       Poly-Lasso     0.8420374  3.301935
9 Poly-Elastic Net     0.8741650  2.947082
```

From the above table, it can be seen that the poly_ridge model is the best fit model as we have minimum RMSE value and maximum R-Squared value in both the Train and the Test data.

Model: POLYNOMIAL RIDGE REGRESSION MODEL

The estimates of Poly Ridge Model are as follows:

```
> coef(polyridge2)
14 x 1 sparse Matrix of class "dgCMatrix"
                      s0
(Intercept)  48.688856865
CRIM         -0.083539523
ZN            0.041003733
INDUS        -0.018113913
CHAS          3.371533643
NOX         -21.958401782
RM            3.037703508
AGE           0.005628192
DIS          -1.783495823
RAD           0.280230583
TAX          -0.010075074
PTRATIO      -1.133022699
B             0.007766838
LSTAT        -0.584926444
```

Therefore, the Model can be written as

MEDV = 48.69 – (0.08*CRIM) + (0.04*ZN) – (0.02*INDUS) + (3.37*CHAS) – (21.96*NOX) + (3.04*RM) + (0.005*AGE) – (1.78*DIS) + (0.28*RAD) – (0.01*TAX) – (1.13*PTRATO) + (0.008*B) – (0.58*LSTAT)

From the given model the following can be concluded about the dataset:

- It can be seen that for *higher RM*, the MEDV Prices would be *higher*. This might be because more rooms would imply more space, thereby costing more, taking all other factors constant.
- For a *higher LSTAT*, the MEDV Prices would be *lower*. The area with more "lower class" citizens would be having lower price mainly because they feel relatively unsafe compared to area with more "upper class" citizens.
- For *higher PTRATIO*, the MEDV Prices would be *lower*. This is because there would be a lower teacher-to-student ratio resulting in less attention dedicated to each student that may impair their performance in school. The prices of houses around such schools are generally lower than those around schools with high teacher-to-student ratio... Hence one would expect a lower price given a high student-to-teacher ratio due to a lower demand for houses in such areas.
- For *higher NOX*, the MEDV Prices would be *very low*. This is because higher the Nitric Oxide concentration, lower would be the demand for the houses in such an environment.
- For *higher DIS*, the MEDV Prices would be *low*. This is because longer the distance between Boston and the employment centre, lower would be the demand for the houses.

REFERENCES:

- Introduction to Linear Regression Analysis – Douglas Montgomery, Elizabeth A. Peck and G. Geoffrey Vining
- https://www.kaggle.com/shreayan98c/boston-house-price-prediction
- https://www.datacamp.com/community/tutorials/tutorial-ridge-lasso-elastic-net
- https://towardsdatascience.com/polynomial-regression-bbe8b9d97491