Project-1

Aim : Property Price Prediction using Linear Regression

Trainer: Indrani Sen

Submitted by:

Dr. Shilpa Joshi

Linear Regression: Linear regression is used to predict the value of an outcome variable *Y* based on one or more input predictor variables *X*. The aim is to establish a linear relationship (a mathematical formula) between the predictor variable(s) and the response variable, so that, we can use this formula to estimate the value of the response *Y*, when only the predictors (*Xs*) values are known.

The aim of linear regression is to model a continuous variable *Y* as a mathematical function of one or more *X* variable(s), so that we can use this regression model to predict the *Y* when only the *X* is known. This mathematical equation can be generalized as follows:

$$Y = \beta_1 + \beta_2 X + \epsilon$$

where, β_1 is the intercept and β_2 is the slope. Collectively, they are called *regression coefficients*. ϵ is the error term, the part of *Y* the regression model is unable to explain.

R-script for Case Study

Case Study: Boston Dataset

1)For this analysis, we will use the *Boston* dataset that comes with R by default. Boston is a standard built-in dataset, that makes it convenient to demonstrate linear regression in a simple and easy to understand fashion. We can access this dataset simply by calling MASS library and then importing Boston day in R console.

2) You will find that it consists of 506 observations(rows) and 14 variables (columns).

Lets print out the first six observations here.

head(d_b)

```
> head(d_b) ## Displays Ist 6 Observations (6)###
     crim zn indus chas   nox    rm  age    dis rad tax ptratio
1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3
2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8
3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8
4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7
5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7
6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7
   black lstat medv
1 396.90  4.98 24.0
2 396.90  9.14 21.6
3 392.83  4.03 34.7
4 394.63  2.94 33.4
5 396.90  5.33 36.2
6 394.12  5.21 28.7
>
```

3) Graphical Analysis

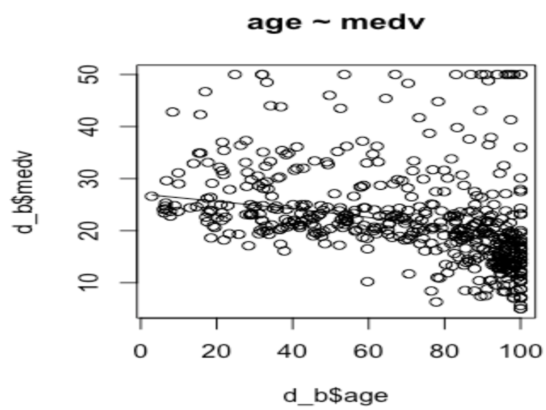
1. **Scatter plot**: Visualize the **linear relationship** between the predictor and response
2. **Box plot**: To spot any outlier observations in the variable. Having outliers in your predictor can drastically affect the predictions as they can easily affect the direction/slope of the line of best fit.

3. **Density plot**: To see the distribution of the predictor variable. Ideally, a close to **normal distribution** (a bell shaped curve), without being skewed to the left or right is preferred. Let us see how to make each one of them.

Scatter Plot

Scatter plots can help visualize any linear relationships between the dependent (response) variable and independent (predictor) variables. Ideally, if you are having multiple predictor variables, a scatter plot is drawn for each one of them against the response, along with the line of best as seen below.

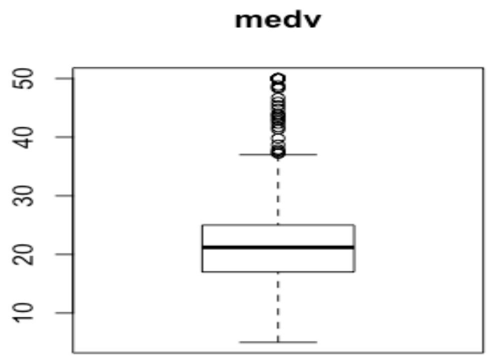
>scatter.smooth(x=d_b$age, y=d_b$medv, main="age ~ medv") # scatterplot



The scatter plot along with the smoothing line above suggests a linearly increasing relationship between the 'age of property' and 'price (medv)' variables. This is a good thing, because, one of the underlying assumptions in linear regression is that the relationship between the response and predictor variables is linear and additive.

BoxPlot – Check for outliers

Generally, any datapoint that lies outside the 1.5 * interquartile-range (1.5 * *IQR*) is considered an outlier, where, IQR is calculated as the distance between the 25th percentile and 75th percentile values for that variable.



outliers

Correlation

Correlation is a statistical measure that suggests the level of linear dependence between two variables, that occur in pair – just like what we have here in **age and medv**. Correlation can take values between -1 to +1. If we observe for every instance where speed increases, the distance also increases along with it, then there is a high positive correlation between them and therefore the correlation between them will be closer to 1. The opposite is true for an inverse relationship, in which case, the correlation between the variables will be close to -1.

A value closer to 0 suggests a weak relationship between the variables. A low correlation ($-0.2 < x < 0.2$) probably suggests that much of variation of the response variable (*Y*) is unexplained by the predictor (*X*), in which case, we should probably look for better explanatory variables.

Build Linear Model

Now that we have seen the linear relationship pictorially in the scatter plot and by computing the correlation, lets see the syntax for building the linear model. The function used for building linear models is lm(). The lm() function takes in two main arguments, namely: 1. Formula 2. Data. The data is typically a data.frame and the formula is a object of class formula. But the most common convention is to write out the formula directly in place of the argument as written below.

```
>linearMod = lm(age ~ medv, data=d_b)  # build linear regression model on full data
> print(linearMod)
```

```
Call:
lm(formula = age ~ medv, data = d_b)

Coefficients:
(Intercept)        medv
     94.571      -1.154
```

Now that we have built the linear model, we also have established the relationship between the predictor and response in the form of a mathematical formula for Distance (dist) as a function for speed. For the above output, you can notice the 'Coefficients' part having two components: *Intercept*: 94.571, *medv*: -1.154 These are also called the beta coefficients. In other words,

$$medv = 94.571 + (-1.154) * age$$

Linear Regression Diagnostics

```
summary(linearMod)

Call:
lm(formula = age ~ medv, data = d_b)

Residuals:
   Min     1Q Median     3Q    Max
-61.37 -21.37   7.31  18.94  63.11

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  94.5713     3.0728  30.777   <2e-16 ***
medv         -1.1537     0.1263  -9.137   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 26.1 on 504 degrees of freedom
```

Multiple R-squared: 0.1421,   Adjusted R-squared: 0.1404
F-statistic: 83.48 on 1 and 504 DF, p-value: < 2.2e-16

The p Value: Checking for statistical significance

The summary statistics above tells us a number of things. One of them is the model p-Value (bottom last line) and the p-Value of individual predictor variables (extreme right column under 'Coefficients'). The p-Values are very important because, We can consider a linear model to be statistically significant only when both these p-Values are less that the pre-determined statistical significance level, which is ideally **0.05**. This is visually interpreted by the significance stars at the end of the row. The more the stars beside the variable's p-Value, the more significant the variable.

Null and alternate hypothesis

When there is a p-value, there is a hull and alternative hypothesis associated with it. In Linear Regression, the Null Hypothesis is that the coefficients associated with the variables is equal to zero. The alternate hypothesis is that the coefficients are not equal to zero (i.e. there exists a relationship between the independent variable in question and the dependent variable).

t-value

We can interpret the t-value something like this. **A larger *t-value* indicates that it is less likely that the coefficient is not equal to zero purely by chance**. So, higher the t-value, the better. *Pr(>|t|)* or *p-value* is the probability that you get a t-value as high or higher than the observed value when the Null Hypothesis (the β coefficient is equal to zero or that there is no relationship) is true. So if the *Pr(>|t|)* is low, the coefficients are significant (significantly different from zero). If the *Pr(>|t|)* is high, the coefficients are not significant.

when p Value is less than significance level (< 0.05), we can safely reject the null hypothesis that the co-efficient β of the predictor is zero. In our case, linearMod, both these p-Values are well below the 0.05 threshold, so we can conclude our model is indeed statistically significant.
It is absolutely important for the model to be statistically significant before we can go ahead and use it to predict (or estimate) the dependent variable, otherwise, the confidence in predicted values from that model reduces and may be construed as an event of chance.

Calculation of the t Statistic and p-Values?

Capture summary model as an object and then calculate model coefficients from that. When the model co-efficients and standard error are known, the formula for calculating t Statistic and p-Value is as foloows

t−Statistic=β−coefficient/Std.Error

modelSummary =summary(linearMod) # capture model summary as an object

> modelCoeffs <- modelSummary$coefficients # model coefficients

> modelCoeffs

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	94.571350	3.0727616	30.77731	7.882546e-118
medv	-1.153716	0.1262741	-9.13660	1.569982e-18

>beta.estimate = modelCoeffs["medv", "Estimate"] # get beta estimate for speed

> beta.estimate

[1] -1.153716

>std.error = modelCoeffs["medv", "Std. Error"] # get std.error for speed

> std.error

[1] 0.1262741

>t_value <- beta.estimate/std.error # calc t statistic

> t_value

[1] -9.1366

>f = summary(linearMod)$fstatistic # parameters for model p-value calc

> f

```
   value     numdf     dendf
 83.47746   1.00000 504.00000
```

> model_p <- pf(f[1], f[2], f[3], lower=FALSE)

> model_p

value

1.569982e-18

T is simply the calculated difference represented in units of standard error. The greater the magnitude of **T**, the greater the evidence against the null hypothesis. T and P values are interlinked and they go hand in hand.

Many other parameters are also important while analyzing structure of data, finding missing values if any in data set. Replacing them..

Find Structure of the data

> str(d_b)

```
'data.frame':	506 obs. of  14 variables:
 $ crim   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
 $ zn     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
 $ indus  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
 $ chas   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ nox    : num  0.538 0.469 0.469 0.458 0.458 0.458 0.524 0.524 0.524 0.524 ...
 $ rm     : num  6.58 6.42 7.18 7 7.15 ...
 $ age    : num  65.2 78.9 61.1 45.8 54.2 58.7 66.6 96.1 100 85.9 ...
 $ dis    : num  4.09 4.97 4.97 6.06 6.06 ...
 $ rad    : int  1 2 2 3 3 3 5 5 5 5 ...
 $ tax    : num  296 242 242 222 222 222 311 311 311 311 ...
 $ ptratio: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
 $ black  : num  397 397 393 395 397 ...
 $ lstat  : num  4.98 9.14 4.03 2.94 5.33 ...
 $ medv   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

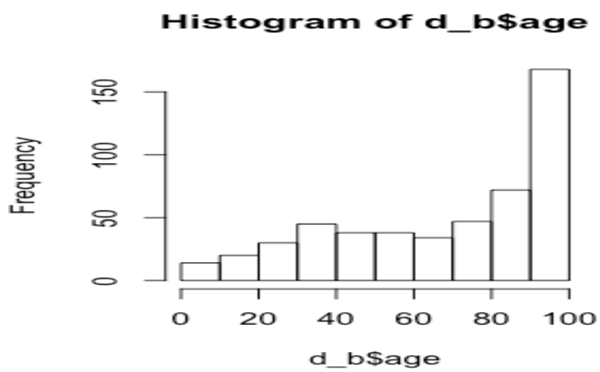
> *###Find out Missing Values###*

> sum(is.na(d_b)) ### Missing Values Calculation ####

[1] 0

Plotting of Histogram to see the range

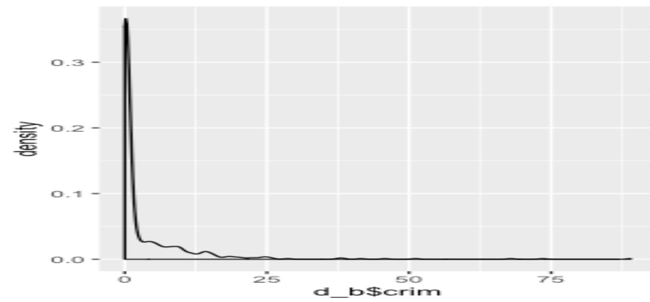
hist(d_b$age,binwidth=10)



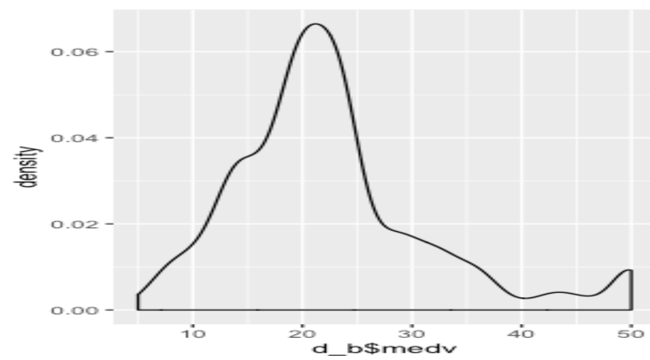
The factors like Kurtosis and skewness are calculated, **Skewness** is a measure of symmetry, or more precisely, **the** lack of symmetry. A distribution, or data set, is symmetric if it looks **the** same to **the** left and right of **the**center point. **Kurtosis** is a measure of whether **the** data are heavy-tailed or light-tailed relative to a normal distribution.

```
###### Find Out Kurtosis and Skewness###
library(moments)
### Kurtosis###
>skewness(d_b$crim)
##5.2076##
>skewness(d_b$medv)
##1.104###
>skewness(d_b$lstat)
###.90377###
>kurtosis(d_b$crim)
###39.75###
#### Plotting of Crim,medv and lstat###
>library(ggplot2)  ## Calling ggplot2 Library for plotting###
```

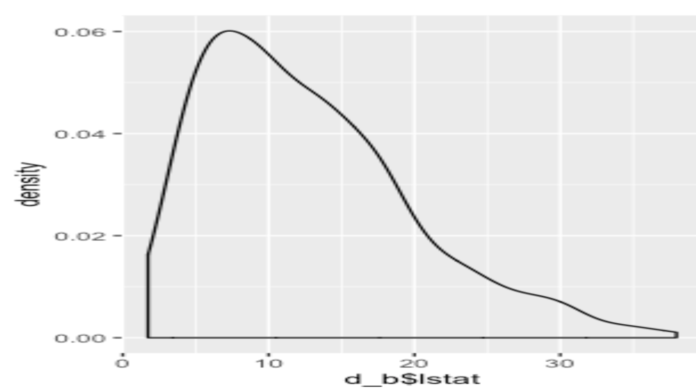
>ggplot(d_b,aes(x=d_b$crim))+geom_density()



>ggplot(d_b,aes(x=d_b$medv))+geom_density()



>ggplot(d_b,aes(x=d_b$lstat))+geom_density()



If data is highly skewed then log transformation is done.

Observe the Crim.. graph..Crim is highally skewed so Log Transformation is done

>d_log$crim=log1p(d_b$crim)

Replace original Crim with log Transformed crim###

>d_b$crim=log1p(d_b$crim)

>d_b$crim #### New log transformed Crim###

Statiscal Analysis For Log Transformed Variables:Crim###

kurtosis(d_b$crim)

###3.487747###

skewness(d_b$crim)

1.265435###

Plotting of handling of variables can be done by Two ways (tapply and grou_by)

zone and price table by method-1 using group_by###

>tapply(d_bzn, d_bmedv, mean)

```
>tapply(d_b$zn, d_b$medv, mean)
       5      5.6      6.3        7      7.2      7.4      7.5
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
     8.1      8.3      8.4      8.5      8.7      8.8      9.5
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
     9.6      9.7     10.2     10.4     10.5     10.8     10.9
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
      11     11.3     11.5     11.7     11.8     11.9       12
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
    12.1     12.3     12.5     12.6     12.7     12.8       13
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
    13.1     13.2     13.3     13.4     13.5     13.6     13.8
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
    13.9       14     14.1     14.2     14.3     14.4     14.5
 0.000000  0.000000  0.000000  0.000000  0.000000  0.000000  0.000000
    14.6     14.8     14.9       15     15.1     15.2     15.3
 0.000000  0.000000  0.000000  4.166667  0.000000  0.000000  0.000000
    15.4     15.6     15.7       16     16.1     16.2     16.3
 0.000000  0.000000  0.000000 25.000000  0.000000  0.000000  0.000000
    16.4     16.5     16.6     16.7     16.8       17     17.1
 0.000000  6.250000  0.000000  0.000000  0.000000  0.000000 11.666667
    17.2     17.3     17.4     17.5     17.6     17.7     17.8
```

0.000000	0.000000	4.166667	0.000000	22.000000	0.000000	0.000000
17.9	18	18.1	18.2	18.3	18.4	18.5
0.000000	0.000000	0.000000	26.666667	0.000000	0.000000	5.500000
18.6	18.7	18.8	18.9	19	19.1	19.2
30.000000	8.333333	0.000000	25.000000	0.000000	0.000000	0.000000
19.3	19.4	19.5	19.6	19.7	19.8	19.9
0.000000	19.166667	0.000000	5.000000	10.500000	0.000000	0.000000
20	20.1	20.2	20.3	20.4	20.5	20.6
0.000000	22.000000	0.000000	0.000000	0.000000	14.333333	18.000000
20.7	20.8	20.9	21	21.1	21.2	21.4
10.000000	0.000000	46.250000	0.000000	10.000000	0.000000	0.000000
21.5	21.6	21.7	21.8	21.9	22	22.1
0.000000	0.000000	1.785714	0.000000	26.666667	18.785714	0.000000
22.2	22.3	22.4	22.5	22.6	22.7	22.8
11.000000	26.250000	0.000000	23.333333	0.000000	0.000000	5.000000
22.9	23	23.1	23.2	23.3	23.4	23.5
26.375000	0.000000	12.142857	13.125000	13.750000	10.500000	80.000000
23.6	23.7	23.8	23.9	24	24.1	24.2
0.000000	15.000000	0.000000	16.000000	9.000000	47.500000	0.000000
24.3	24.4	24.5	24.6	24.7	24.8	25
7.333333	10.500000	34.000000	0.000000	28.333333	42.375000	9.250000
25.1	25.2	25.3	26.2	26.4	26.5	26.6
0.000000	20.000000	0.000000	22.000000	17.000000	0.000000	13.333333
26.7	27	27.1	27.5	27.9	28	28.1
0.000000	0.000000	6.250000	0.000000	40.000000	25.000000	0.000000
28.2	28.4	28.5	28.6	28.7	29	29.1
33.000000	16.500000	80.000000	0.000000	0.000000	35.000000	50.000000
29.4	29.6	29.8	29.9	30.1	30.3	30.5
0.000000	11.000000	22.500000	0.000000	36.666667	80.000000	45.000000
30.7	30.8	31	31.1	31.2	31.5	31.6
20.000000	75.000000	20.000000	60.000000	55.000000	0.000000	50.000000

```
    31.7        32      32.2      32.4      32.5      32.7      32.9 
 0.000000 42.500000 90.000000 40.000000  0.000000 35.000000 95.000000 
      33      33.1      33.2      33.3      33.4      33.8      34.6 
17.500000 37.000000 20.000000 80.000000 16.500000 20.000000 80.000000 
    34.7      34.9      35.1      35.2      35.4        36      36.1 
 0.000000 71.666667 20.000000 20.000000 55.000000 20.000000 33.000000 
    36.2      36.4      36.5        37      37.2      37.3      37.6 
 0.000000 45.000000 20.000000 45.000000  0.000000 80.000000  0.000000 
    37.9      38.7      39.8      41.3      41.7      42.3      42.8 
 0.000000  0.000000  0.000000  0.000000  0.000000 82.500000 22.000000 
    43.1      43.5      43.8        44      44.8      45.4        46 
20.000000 20.000000  0.000000 90.000000  0.000000 20.000000 20.000000 
    46.7      48.3      48.5      48.8        50 
 0.000000  0.000000 95.000000 20.000000 19.062500 
>
```

```
#### zone and price table by method-2 which is good for Printing using ###
library(dplyr)    #  loads %>%
d_b%>% group_by(zn)%>%summarise(mean(medv))###  It will Print Table##
# A tibble: 26 x 2
      zn `mean(medv)`
   <dbl>        <dbl>
 1   0          20.5
 2  12.5         20.1
 3  17.5         33
 4  18           24
 5  20           35.5
 6  21           22.2
 7  22           25.3
 8  25           22.4
 9  28           22.8
```

10 30   22.5
… with 16 more rows

Part-II: Predicting Linear Models

So far we have seen how to build a linear regression model using the whole dataset. If we build it that way, there is no way to tell how the model will perform with new data. So the preferred practice is to split your dataset into a 80:20 sample (training:test), then, build the model on the 80% sample and then use the model thus built to predict the dependent variable on test data.

Doing it this way, we will have the model predicted values for the 20% data (test) as well as the actuals (from the original dataset). By calculating accuracy measures (like min_max accuracy) and error rates (MAPE or MSE), we can find out the prediction accuracy of the model. Now, lets see how to actually do this..

Step 1: Create the training (development) and test (validation) data samples from original data after sampling

>s=sample(nrow(d_b),.80*nrow(d_b)) ###Sampling###

> s

```
  [1] 390 378  60 186   5 110  43 327 420 241 454 497   4 318 275 348 202  75
 [19] 311 330  86  97  15 205 253 180 323 506 352 387 136 158 287 141 233 342
 [37]  92 316  57 437 450 403 182 359 483 207 254 391 174 479 426 177 163 463
 [55] 201  61 247 428  77 351 462  22 232 473 466 389  32 271  46 248 281 234
 [73] 432 381 239 195 246 286 396 152 263 401  65 472  45 357 363 103 382 265
 [91] 278  67  27 361 255 106 332 373 135 315 375  78 112  68 364 486 140  62
[109] 309 397 477 170 502 279 146  80 161 166  81 505 325 159 267  55 284 150
[127]  91 298  23 430 371 192 231 297 377 415 372 240 259 449 149  98 301 481
[145] 461 221  28 273 499 331 264  20 175 501 185  53 226 164  94  99 306 353
[163] 245 484  84 335 101 168  41 485 345  42 200 383  44 329 300  13 458  12
[181] 211  31 445  59 237  14 453  76 157 190 117  19 125 395 123   8 296 172
[199] 276 421 134 230  93 496  37 113 269 250 347 120 108 394 460 154 310  48
```

```
[217] 442 369  39 102 289 137 302 413 418 322 360 144 105 127 312 474   7 406
[235] 283 362 142 261 277 400 274 337 194 328 242 407 270 282  66 111 452 355
[253] 405 162 131 133   1 356 431  87   6 118 109 122 189  89 419 438 299 199
[271] 334   3 304 398 384  96 197 457   9  51 165 295 260 386 498 343 392 236
[289] 491  11 366 464 196 470 388 338 358 193 107 385 434 262 155 147 321 435
[307] 244  26 143  85 294 139 500  63 459 492 218  34 427 183 412 222 488 121
[325]  47  35 475 425 266 148  16  24  73 433 224 423 493 354 305 198 451 333
[343] 317 495 455 379 487 209 429 404 167 214 219  79 227 169 258 482 344 235
[361] 367 374 417 468 324 243 489 503 341  54 467 138 171 252  49 402 444 132
[379] 210 114 411 272 160 256 184 251  18 446  33 399  25 208 145  52 376 292
[397] 291 124 380 116 346 223 393 439
>
```

```
#### Preapre Training Data Set###
> df_tr=d_b[s,]
> df_tr
        crim   zn indus chas    nox    rm   age     dis rad tax ptratio  black
390  8.15174  0.0 18.10    0 0.7000 5.390  98.9  1.7281  24 666    20.2 396.90
378  9.82349  0.0 18.10    0 0.6710 6.794  98.8  1.3580  24 666    20.2 396.90
60   0.10328 25.0  5.13    0 0.4530 5.927  47.2  6.9320   8 284    19.7 396.90
186  0.06047  0.0  2.46    0 0.4880 6.153  68.8  3.2797   3 193    17.8 387.11
5    0.06905  0.0  2.18    0 0.4580 7.147  54.2  6.0622   3 222    18.7 396.90
110  0.26363  0.0  8.56    0 0.5200 6.229  91.2  2.5451   5 384    20.9 391.23
43   0.14150  0.0  6.91    0 0.4480 6.169   6.6  5.7209   3 233    17.9 383.37
327  0.30347  0.0  7.38    0 0.4930 6.312  28.9  5.4159   5 287    19.6 396.90
420 11.81230  0.0 18.10    0 0.7180 6.824  76.5  1.7940  24 666    20.2  48.45
241  0.11329 30.0  4.93    0 0.4280 6.897  54.3  6.3361   6 300    16.6 391.25
454  8.24809  0.0 18.10    0 0.7130 7.393  99.3  2.4527  24 666    20.2 375.87
497  0.28960  0.0  9.69    0 0.5850 5.390  72.9  2.7986   6 391    19.2 396.90
4    0.03237  0.0  2.18    0 0.4580 6.998  45.8  6.0622   3 222    18.7 394.63
318  0.24522  0.0  9.90    0 0.5440 5.782  71.7  4.0317   4 304    18.4 396.90
```

275	0.05644	40.0	6.41	1	0.4470	6.758	32.9	4.0776	4	254	17.6	396.90
348	0.01870	85.0	4.15	0	0.4290	6.516	27.7	8.5353	4	351	17.9	392.43
202	0.03445	82.5	2.03	0	0.4150	6.162	38.4	6.2700	2	348	14.7	393.77
75	0.07896	0.0	12.83	0	0.4370	6.273	6.0	4.2515	5	398	18.7	394.92
311	2.63548	0.0	9.90	0	0.5440	4.973	37.8	2.5194	4	304	18.4	350.45
330	0.06724	0.0	3.24	0	0.4600	6.333	17.2	5.2146	4	430	16.9	375.21
86	0.05735	0.0	4.49	0	0.4490	6.630	56.1	4.4377	3	247	18.5	392.30
97	0.11504	0.0	2.89	0	0.4450	6.163	69.6	3.4952	2	276	18.0	391.83
15	0.63796	0.0	8.14	0	0.5380	6.096	84.5	4.4619	4	307	21.0	380.02
205	0.02009	95.0	2.68	0	0.4161	8.034	31.9	5.1180	4	224	14.7	390.55
253	0.08221	22.0	5.86	0	0.4310	6.957	6.8	8.9067	7	330	19.1	386.09
180	0.05780	0.0	2.46	0	0.4880	6.980	58.4	2.8290	3	193	17.8	396.90
323	0.35114	0.0	7.38	0	0.4930	6.041	49.9	4.7211	5	287	19.6	396.90
506	0.04741	0.0	11.93	0	0.5730	6.030	80.8	2.5050	1	273	21.0	396.90
352	0.07950	60.0	1.69	0	0.4110	6.579	35.9	10.7103	4	411	18.3	370.78
387	24.39380	0.0	18.10	0	0.7000	4.652	100.0	1.4672	24	666	20.2	396.90
136	0.55778	0.0	21.89	0	0.6240	6.335	98.2	2.1107	4	437	21.2	394.67
158	1.22358	0.0	19.58	0	0.6050	6.943	97.4	1.8773	5	403	14.7	363.43
287	0.01965	80.0	1.76	0	0.3850	6.230	31.5	9.0892	1	241	18.2	341.60
141	0.29090	0.0	21.89	0	0.6240	6.174	93.6	1.6119	4	437	21.2	388.08
233	0.57529	0.0	6.20	0	0.5070	8.337	73.3	3.8384	8	307	17.4	385.91
342	0.01301	35.0	1.52	0	0.4420	7.241	49.3	7.0379	1	284	15.5	394.74
92	0.03932	0.0	3.41	0	0.4890	6.405	73.9	3.0921	2	270	17.8	393.55
316	0.25356	0.0	9.90	0	0.5440	5.705	77.7	3.9450	4	304	18.4	396.42
57	0.02055	85.0	0.74	0	0.4100	6.383	35.7	9.1876	2	313	17.3	396.90
437	14.42080	0.0	18.10	0	0.7400	6.461	93.3	2.0026	24	666	20.2	27.49
450	7.52601	0.0	18.10	0	0.7130	6.417	98.3	2.1850	24	666	20.2	304.21
403	9.59571	0.0	18.10	0	0.6930	6.404	100.0	1.6390	24	666	20.2	376.11
182	0.06888	0.0	2.46	0	0.4880	6.144	62.2	2.5979	3	193	17.8	396.90
359	5.20177	0.0	18.10	1	0.7700	6.127	83.4	2.7227	24	666	20.2	395.43
483	5.73116	0.0	18.10	0	0.5320	7.061	77.0	3.4106	24	666	20.2	395.28

207	0.22969	0.0	10.59	0	0.4890	6.326	52.5	4.3549	4	277	18.6	394.87
254	0.36894	22.0	5.86	0	0.4310	8.259	8.4	8.9067	7	330	19.1	396.90
391	6.96215	0.0	18.10	0	0.7000	5.713	97.0	1.9265	24	666	20.2	394.43
174	0.09178	0.0	4.05	0	0.5100	6.416	84.1	2.6463	5	296	16.6	395.50
479	10.23300	0.0	18.10	0	0.6140	6.185	96.7	2.1705	24	666	20.2	379.70
426	15.86030	0.0	18.10	0	0.6790	5.896	95.4	1.9096	24	666	20.2	7.68
177	0.07022	0.0	4.05	0	0.5100	6.020	47.2	3.5549	5	296	16.6	393.23
163	1.83377	0.0	19.58	1	0.6050	7.802	98.2	2.0407	5	403	14.7	389.61
463	6.65492	0.0	18.10	0	0.7130	6.317	83.0	2.7344	24	666	20.2	396.90
201	0.01778	95.0	1.47	0	0.4030	7.135	13.9	7.6534	3	402	17.0	384.30
61	0.14932	25.0	5.13	0	0.4530	5.741	66.2	7.2254	8	284	19.7	395.11
247	0.33983	22.0	5.86	0	0.4310	6.108	34.9	8.0555	7	330	19.1	390.18
428	37.66190	0.0	18.10	0	0.6790	6.202	78.7	1.8629	24	666	20.2	18.82
77	0.10153	0.0	12.83	0	0.4370	6.279	74.5	4.0522	5	398	18.7	373.66
351	0.06211	40.0	1.25	0	0.4290	6.490	44.4	8.7921	1	335	19.7	396.90
462	3.69311	0.0	18.10	0	0.7130	6.376	88.4	2.5671	24	666	20.2	391.43
22	0.85204	0.0	8.14	0	0.5380	5.965	89.2	4.0123	4	307	21.0	392.53
232	0.46296	0.0	6.20	0	0.5040	7.412	76.9	3.6715	8	307	17.4	376.14
473	3.56868	0.0	18.10	0	0.5800	6.437	75.0	2.8965	24	666	20.2	393.37
466	3.16360	0.0	18.10	0	0.6550	5.759	48.2	3.0665	24	666	20.2	334.40
389	14.33370	0.0	18.10	0	0.7000	4.880	100.0	1.5895	24	666	20.2	372.92
32	1.35472	0.0	8.14	0	0.5380	6.072	100.0	4.1750	4	307	21.0	376.73
271	0.29916	20.0	6.96	0	0.4640	5.856	42.1	4.4290	3	223	18.6	388.65
46	0.17142	0.0	6.91	0	0.4480	5.682	33.8	5.1004	3	233	17.9	396.90
248	0.19657	22.0	5.86	0	0.4310	6.226	79.2	8.0555	7	330	19.1	376.14
281	0.03578	20.0	3.33	0	0.4429	7.820	64.5	4.6947	5	216	14.9	387.31

	lstat	medv
390	20.85	11.5
378	21.24	13.3
60	9.22	19.6
186	13.15	29.6

5	5.33	36.2
110	15.55	19.4
43	5.81	25.3
327	6.15	23.0
420	22.74	8.4
241	11.38	22.0
454	16.74	17.8
497	21.14	19.7
4	2.94	33.4
318	15.94	19.8
275	3.53	32.4
348	6.36	23.1
202	7.43	24.1
75	6.78	24.1
311	12.64	16.1
330	7.34	22.6
86	6.53	26.6
97	11.34	21.4
15	10.26	18.2
205	2.88	50.0
253	3.53	29.6
180	5.04	37.2
323	7.70	20.4
506	7.88	11.9
352	5.49	24.1
387	28.28	10.5
136	16.96	18.1
158	4.59	41.3
287	12.93	20.1
141	24.16	14.0
233	2.47	41.7

342	5.49	32.7
92	8.20	22.0
316	11.50	16.2
57	5.77	24.7
437	18.05	9.6
450	19.31	13.0
403	20.31	12.1
182	9.45	36.2
359	11.48	22.7
483	7.01	25.0
207	10.97	24.4
254	3.54	42.8
391	17.11	15.1
174	9.04	23.6
479	18.03	14.6
426	24.39	8.3
177	10.11	23.2
163	1.92	50.0
463	13.99	19.5
201	4.45	32.9
61	13.15	18.7
247	9.16	24.3
428	14.52	10.9
77	11.97	20.0
351	5.98	22.9
462	14.65	17.7
22	13.83	19.6
232	5.25	31.7
473	14.36	23.2
466	14.13	19.9
389	30.62	10.2

```
32  13.04 14.5
271 13.00 21.1
46  10.21 19.3
248 10.15 20.5
281  3.76 45.4
 [ reached 'max' / getOption("max.print") -- omitted 333 rows ]
```

> ####Prepare Testing Data Set###

> df_ts=d_b[-s,]###-sign stands for remaimng data##

> df_ts

```
       crim    zn indus chas    nox    rm  age    dis rad tax ptratio  black
2   0.02731   0.0  7.07    0 0.4690 6.421 78.9 4.9671   2 242    17.8 396.90
10  0.17004  12.5  7.87    0 0.5240 6.004 85.9 6.5921   5 311    15.2 386.71
17  1.05393   0.0  8.14    0 0.5380 5.935 29.3 4.4986   4 307    21.0 386.85
21  1.25179   0.0  8.14    0 0.5380 5.570 98.1 3.7979   4 307    21.0 376.57
29  0.77299   0.0  8.14    0 0.5380 6.495 94.4 4.4547   4 307    21.0 387.94
30  1.00245   0.0  8.14    0 0.5380 6.674 87.3 4.2390   4 307    21.0 380.23
36  0.06417   0.0  5.96    0 0.4990 5.933 68.2 3.3603   5 279    19.2 396.90
38  0.08014   0.0  5.96    0 0.4990 5.850 41.5 3.9342   5 279    19.2 396.90
40  0.02763  75.0  2.95    0 0.4280 6.595 21.8 5.4011   3 252    18.3 395.63
50  0.21977   0.0  6.91    0 0.4480 5.602 62.0 6.0877   3 233    17.9 396.90
56  0.01311  90.0  1.22    0 0.4030 7.249 21.9 8.6966   5 226    17.9 395.93
58  0.01432 100.0  1.32    0 0.4110 6.816 40.5 8.3248   5 256    15.1 392.90
64  0.12650  25.0  5.13    0 0.4530 6.762 43.4 7.9809   8 284    19.7 395.58
69  0.13554  12.5  6.07    0 0.4090 5.594 36.8 6.4980   4 345    18.9 396.90
70  0.12816  12.5  6.07    0 0.4090 5.885 33.0 6.4980   4 345    18.9 396.90
71  0.08826   0.0 10.81    0 0.4130 6.417  6.6 5.2873   4 305    19.2 383.73
72  0.15876   0.0 10.81    0 0.4130 5.961 17.5 5.2873   4 305    19.2 376.94
74  0.19539   0.0 10.81    0 0.4130 6.245  6.2 5.2873   4 305    19.2 377.17
82  0.04462  25.0  4.86    0 0.4260 6.619 70.4 5.4007   4 281    19.0 395.63
83  0.03659  25.0  4.86    0 0.4260 6.302 32.2 5.4007   4 281    19.0 396.90
```

88	0.07151	0.0	4.49	0	0.4490	6.121	56.8	3.7476	3	247	18.5	395.15
90	0.05302	0.0	3.41	0	0.4890	7.079	63.1	3.4145	2	270	17.8	396.06
95	0.04294	28.0	15.04	0	0.4640	6.249	77.3	3.6150	4	270	18.2	396.90
100	0.06860	0.0	2.89	0	0.4450	7.416	62.5	3.4952	2	276	18.0	396.90
104	0.21161	0.0	8.56	0	0.5200	6.137	87.4	2.7147	5	384	20.9	394.47
115	0.14231	0.0	10.01	0	0.5470	6.254	84.2	2.2565	6	432	17.8	388.74
119	0.13058	0.0	10.01	0	0.5470	5.872	73.1	2.4775	6	432	17.8	338.63
126	0.16902	0.0	25.65	0	0.5810	5.986	88.4	1.9929	2	188	19.1	385.02
128	0.25915	0.0	21.89	0	0.6240	5.693	96.0	1.7883	4	437	21.2	392.11
129	0.32543	0.0	21.89	0	0.6240	6.431	98.8	1.8125	4	437	21.2	396.90
130	0.88125	0.0	21.89	0	0.6240	5.637	94.7	1.9799	4	437	21.2	396.90
151	1.65660	0.0	19.58	0	0.8710	6.122	97.3	1.6180	5	403	14.7	372.80
153	1.12658	0.0	19.58	1	0.8710	5.012	88.0	1.6102	5	403	14.7	343.28
156	3.53501	0.0	19.58	1	0.8710	6.152	82.6	1.7455	5	403	14.7	88.01
173	0.13914	0.0	4.05	0	0.5100	5.572	88.5	2.5961	5	296	16.6	396.90
176	0.06664	0.0	4.05	0	0.5100	6.546	33.1	3.1323	5	296	16.6	390.96
178	0.05425	0.0	4.05	0	0.5100	6.315	73.4	3.3175	5	296	16.6	395.60
179	0.06642	0.0	4.05	0	0.5100	6.860	74.4	2.9153	5	296	16.6	391.27
181	0.06588	0.0	2.46	0	0.4880	7.765	83.3	2.7410	3	193	17.8	395.56
187	0.05602	0.0	2.46	0	0.4880	7.831	53.6	3.1992	3	193	17.8	392.63
188	0.07875	45.0	3.44	0	0.4370	6.782	41.1	3.7886	5	398	15.2	393.87
191	0.09068	45.0	3.44	0	0.4370	6.951	21.5	6.4798	5	398	15.2	377.68
203	0.02177	82.5	2.03	0	0.4150	7.610	15.7	6.2700	2	348	14.7	395.38
204	0.03510	95.0	2.68	0	0.4161	7.853	33.2	5.1180	4	224	14.7	392.78
206	0.13642	0.0	10.59	0	0.4890	5.891	22.3	3.9454	4	277	18.6	396.90
212	0.37578	0.0	10.59	1	0.4890	5.404	88.6	3.6650	4	277	18.6	395.24
213	0.21719	0.0	10.59	1	0.4890	5.807	53.8	3.6526	4	277	18.6	390.94
215	0.28955	0.0	10.59	0	0.4890	5.412	9.8	3.5875	4	277	18.6	348.93
216	0.19802	0.0	10.59	0	0.4890	6.182	42.4	3.9454	4	277	18.6	393.63
217	0.04560	0.0	13.89	1	0.5500	5.888	56.0	3.1121	5	276	16.4	392.80
220	0.11425	0.0	13.89	1	0.5500	6.373	92.4	3.3633	5	276	16.4	393.74

225	0.31533	0.0	6.20	0	0.5040	8.266	78.3	2.8944	8	307	17.4	385.05
228	0.41238	0.0	6.20	0	0.5040	7.163	79.9	3.2157	8	307	17.4	372.08
229	0.29819	0.0	6.20	0	0.5040	7.686	17.0	3.3751	8	307	17.4	377.51
238	0.51183	0.0	6.20	0	0.5070	7.358	71.6	4.1480	8	307	17.4	390.07
249	0.16439	22.0	5.86	0	0.4310	6.433	49.1	7.8265	7	330	19.1	374.71
257	0.01538	90.0	3.75	0	0.3940	7.454	34.2	6.3361	3	244	15.9	386.34
268	0.57834	20.0	3.97	0	0.5750	8.297	67.0	2.4216	5	264	13.0	384.54
280	0.21038	20.0	3.33	0	0.4429	6.812	32.2	4.1007	5	216	14.9	396.90
285	0.00906	90.0	2.97	0	0.4000	7.088	20.8	7.3073	1	285	15.3	394.72
288	0.03871	52.5	5.32	0	0.4050	6.209	31.3	7.3172	6	293	16.6	396.90
290	0.04297	52.5	5.32	0	0.4050	6.565	22.9	7.3172	6	293	16.6	371.72
293	0.03615	80.0	4.95	0	0.4110	6.630	23.4	5.1167	4	245	19.2	396.90
303	0.09266	34.0	6.09	0	0.4330	6.495	18.4	5.4917	7	329	16.1	383.61
307	0.07503	33.0	2.18	0	0.4720	7.420	71.9	3.0992	7	222	18.4	396.90
308	0.04932	33.0	2.18	0	0.4720	6.849	70.3	3.1827	7	222	18.4	396.90
313	0.26169	0.0	9.90	0	0.5440	6.023	90.4	2.8340	4	304	18.4	396.30
314	0.26938	0.0	9.90	0	0.5440	6.266	82.8	3.2628	4	304	18.4	393.39
319	0.40202	0.0	9.90	0	0.5440	6.382	67.2	3.5325	4	304	18.4	395.21
320	0.47547	0.0	9.90	0	0.5440	6.113	58.8	4.0019	4	304	18.4	396.23
326	0.19186	0.0	7.38	0	0.4930	6.431	14.7	5.4159	5	287	19.6	393.68

	lstat	medv
2	9.14	21.6
10	17.10	18.9
17	6.58	23.1
21	21.02	13.6
29	12.80	18.4
30	11.98	21.0
36	9.68	18.9
38	8.77	21.0
40	4.32	30.8
50	16.20	19.4

56	4.81	35.4
58	3.95	31.6
64	9.50	25.0
69	13.09	17.4
70	8.79	20.9
71	6.72	24.2
72	9.88	21.7
74	7.54	23.4
82	7.22	23.9
83	6.72	24.8
88	8.44	22.2
90	5.70	28.7
95	10.59	20.6
100	6.19	33.2
104	13.44	19.3
115	10.45	18.5
119	15.37	20.4
126	14.81	21.4
128	17.19	16.2
129	15.39	18.0
130	18.34	14.3
151	14.10	21.5
153	12.12	15.3
156	15.02	15.6
173	14.69	23.1
176	5.33	29.4
178	6.29	24.6
179	6.92	29.9
181	7.56	39.8
187	4.45	50.0
188	6.68	32.0

191	5.10	37.0
203	3.11	42.3
204	3.81	48.5
206	10.87	22.6
212	23.98	19.3
213	16.03	22.4
215	29.55	23.7
216	9.47	25.0
217	13.51	23.3
220	10.50	23.0
225	4.14	44.8
228	6.36	31.6
229	3.92	46.7
238	4.73	31.5
249	9.52	24.5
257	3.11	44.0
268	7.44	50.0
280	4.85	35.1
285	7.85	32.2
288	7.14	23.2
290	9.51	24.8
293	4.70	27.9
303	8.67	26.4
307	6.47	33.4
308	7.53	28.2
313	11.72	19.4
314	7.90	21.6
319	10.36	23.1
320	12.73	21.0
326	5.08	24.6

[reached 'max' / getOption("max.print") -- omitted 31 rows]

>head(df_tr)### Display ist six rows##

```
       crim zn indus chas   nox    rm  age    dis rad tax ptratio  black lstat
390 8.15174  0 18.10    0 0.700 5.390 98.9 1.7281  24 666    20.2 396.90 20.85
378 9.82349  0 18.10    0 0.671 6.794 98.8 1.3580  24 666    20.2 396.90 21.24
60  0.10328 25  5.13    0 0.453 5.927 47.2 6.9320   8 284    19.7 396.90  9.22
186 0.06047  0  2.46    0 0.488 6.153 68.8 3.2797   3 193    17.8 387.11 13.15
5   0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90  5.33
110 0.26363  0  8.56    0 0.520 6.229 91.2 2.5451   5 384    20.9 391.23 15.55
    medv
390 11.5
378 13.3
60  19.6
186 29.6
5   36.2
110 19.4
```

Step 2: Develop the model on the training data and use it to predict the distance on test data

```
>Linrearmodel=lm(medv~.,data=df_tr)#####
> summary(Linrearmodel)

Call:
lm(formula = medv ~ ., data = df_tr)

Residuals:
    Min      1Q  Median      3Q     Max 
-10.953  -2.756  -0.575   1.763  25.924 
```

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  32.904538   5.775851   5.697 2.41e-08 ***
crim         -0.109353   0.033705  -3.244 0.001278 **
zn            0.044175   0.016087   2.746 0.006311 **
indus         0.034628   0.066837   0.518 0.604681
chas          3.331052   0.978222   3.405 0.000730 ***
nox         -13.954187   4.314143  -3.235 0.001322 **
rm            3.666942   0.480633   7.629 1.83e-13 ***
age           0.010954   0.015362   0.713 0.476233
dis          -1.274753   0.222227  -5.736 1.94e-08 ***
rad           0.276809   0.073523   3.765 0.000192 ***
tax          -0.012474   0.004181  -2.983 0.003033 **
ptratio      -0.889940   0.142852  -6.230 1.21e-09 ***
black         0.009894   0.002898   3.414 0.000706 ***
lstat        -0.555489   0.056434  -9.843  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.748 on 390 degrees of freedom
Multiple R-squared:  0.7409,  Adjusted R-squared:  0.7323
F-statistic: 85.79 on 13 and 390 DF,  p-value: < 2.2e-16
```

Analysis of Model: ###here crim ,zn,dis,rad are significant *###**

###Plot Residual###

Residual plots are used to look for underlying patterns in the residuals that may mean that the model has a problem. When using the plot() function, the first plot is the Residuals vs Fitted plot and gives an indication if there are non-linear patterns.

```
Linrearmodel$residuals
       390        378         60        186          5        110
```

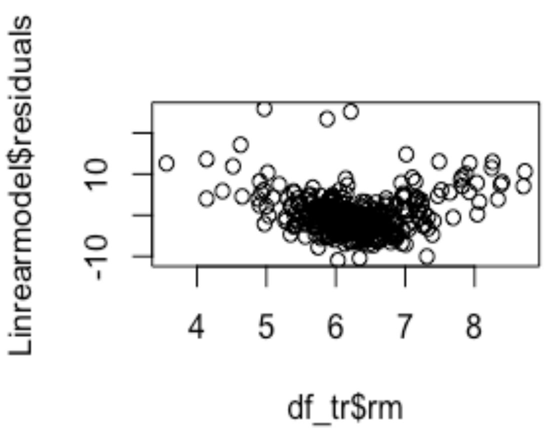
-2.72119807	-6.54549548	-1.61373066	5.18367021	8.15957088	-0.33920144
43	327	420	241	454	497
0.46252320	-0.67767976	-5.60139449	-4.91963258	-4.72983165	5.77839715
4	318	275	348	202	75
4.68878834	1.30965582	-3.76640112	-2.18023045	-4.64732350	-0.70966252
311	330	86	97	15	205
-2.09242215	-1.09274119	-0.99717309	-2.95801748	-1.65031201	7.83394763
253	180	323	506	352	387
4.62992454	4.68833297	-2.53345058	-10.95250335	3.05187396	4.54377271
136	158	287	141	233	342
0.31758424	7.83914083	0.10048493	0.25803919	3.89113829	2.44910446
92	316	57	437	450	403
-5.23275860	-4.64494441	-0.32179596	-4.79403873	-4.22356200	-5.99915697
182	359	483	207	254	391
8.86859347	-0.65331123	-3.24493263	0.84458480	12.96799759	-2.21507276
174	479	426	177	163	463
-5.20321800	-4.31714159	-1.13976890	-1.97419085	8.58226957	-0.45643154
201	61	247	428	77	351
2.79651839	0.54002257	4.16538918	-2.74727385	-2.74027582	1.92999759
462	22	232	473	466	389
-2.64834024	1.48817921	-1.38235607	1.14486439	2.29926659	4.00060658
32	271	46	248	281	234
-4.14262991	-0.83021790	-2.52704913	0.12060360	7.11053177	11.47950992
432	381	239	195	246	286
-3.83512271	-3.51642314	-4.16318542	-1.83536983	4.85514025	-4.93206719
396	152	263	401	65	472
-7.14631653	0.17470518	8.01343194	-5.96172163	9.13615559	-3.07137455
45	357	363	103	382	265
-1.62073564	-2.79565093	1.91519934	-1.07141337	-7.25741608	0.83606306
278	67	27	361	255	106
-1.87214116	-5.54404963	0.56672579	1.75749582	-2.17399853	0.99688184

332	373	135	315	375	78
-2.65227844	23.41010355	1.78009431	-1.96118223	13.59910928	-2.14486691
112	68	364	486	140	62
-3.61275363	1.13687106	-4.40166936	-0.74702959	0.94186715	-2.91416588
309	397	477	170	502	279
-6.15245770	-6.64586627	-3.49634715	-4.45905577	-1.37791550	-0.59894167
146	80	161	166	81	505
1.32384763	-1.78366273	-6.54527323	-0.50221754	-0.05966792	-4.63190649
325	159	267	55	284	150
-0.07630901	-4.99687431	-0.14490446	3.57691635	5.65948874	-0.18888161
91	298	23	430	371	192
-4.23784065	0.82519720	-0.91867482	-3.05216125	14.80156600	0.66241656
231	297	377	415	372	240
0.10279354	-0.20309386	-3.32650233	11.89847590	25.19669387	-4.76225302
259	449	149	98	301	481
-0.38119131	-3.55492407	7.45654015	3.27252124	-5.66666788	-0.02353696
461	221	28	273	499	331
-2.66575705	-6.82838076	-0.26570781	-3.87498345	-0.01301394	-1.48852088
264	20	175	501	185	53
-3.22467748	-0.51305540	-3.56621035	-3.70487934	3.81985571	-2.43424880
226	164	94	99	306	353
10.65013358	7.43758387	-3.61718974	9.38746050	-1.77129694	1.35464049
245	484	84	335	101	168
0.78566467	1.19613040	-1.95626765	-1.23723185	2.90243271	0.74756883
41	485	345	42	200	383
1.29623813	1.60690622	2.83606801	-0.96132415	5.30583626	-2.01609672
44	329	300	13	458	12
0.50728516	-1.39253076	-2.22716544	1.10534103	0.72727728	-3.04274216
211	31	445	59	237	14
-1.37765378	0.96219572	-0.63690061	1.36558126	-5.52830584	0.37158290
453	76	157	190	117	19

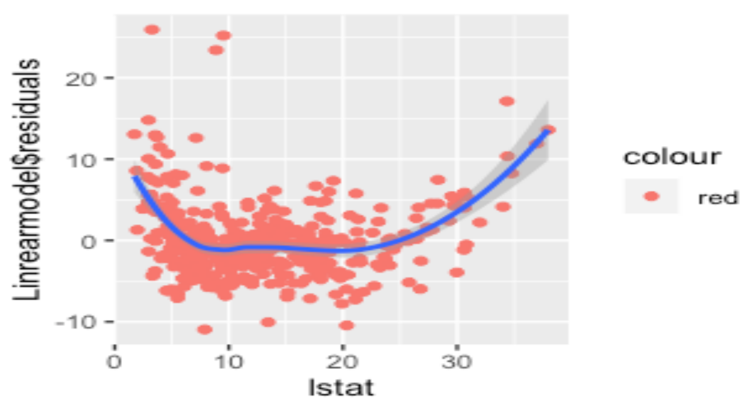
-2.57856390	-2.24716770	-1.39550329	1.21880428	-1.99920097	4.08600134
125	395	123	8	296	172
-2.12309440	-5.25668715	-0.38268930	7.33210233	0.38790532	-5.40645409
276	421	134	230	93	496
-1.35518854	-3.00356212	1.94943115	0.92977687	-5.73710226	6.69191572
37	113	269	250	347	120
-2.13668462	-1.83929921	4.72217985	2.23143850	2.05468391	-1.25898453
108	394	460	154	310	48
-0.22876212	-6.41524831	1.28247783	1.16670813	-3.42468012	-1.49950446
442	369	39	102	289	137
-0.29904167	25.92443412	2.18552916	0.91446194	-4.71822944	1.03027979
302	413	418	322	360	144
-6.32252034	17.11944181	4.03403421	-1.82001830	2.98805207	2.74725511
105	127	312	474	7	406
-1.37358517	1.04444762	-4.67681610	4.64871652	-0.18137274	-2.95405998
283	362	142	261	277	400
5.53010175	0.59327730	10.35614063	-0.73557080	-2.62387403	-3.93193112
274	337	194	328	242	407
-0.57169850	-0.85035834	-0.25964298	2.88154869	-3.46457572	4.00562209
270	282	66	111	452	355
-5.34884554	1.64744262	-6.23777551	1.32933460	-4.28926818	3.27354569
405	162	131	133	1	356
1.60260236	13.05729495	-1.48057489	2.20905976	-6.01121277	3.39498218
431	87	6	118	109	122
-2.94351631	0.74424402	3.19603467	-4.49580790	-2.92176507	-2.53437976
189	89	419	438	299	199
-1.78468663	-7.08700337	3.07560114	0.18155703	-6.12381202	0.53526538
334	3	304	398	384	96
-0.35082126	4.07172956	1.07092838	-7.78016203	-0.61507865	0.20753899
197	457	9	51	165	295
-2.22357468	0.13877928	4.93802658	-1.43878996	-2.29768976	-2.47605646

260	386	498	343	392	236
-4.91334122	-0.50337329	-0.87086111	-5.80199720	6.00476233	-0.92754476
491	11	366	464	196	470
4.31566635	-4.22059080	12.60827475	-2.58108279	10.04645503	2.03421288
388	338	358	193	107	385
2.21235059	-1.15958213	-2.10962336	3.98513210	2.44187863	5.87937645
434	262	155	147	321	435
-2.62319555	6.10295394	-6.95321267	-1.09327343	-1.09967590	-4.26067459
244	26	143	85	294	139
-3.24384157	0.10152776	-2.52312863	-0.61891928	-1.45787226	-0.87891221
500	63	459	492	218	34
-0.94899718	-2.18745864	-2.34540593	-0.56820788	0.22034937	-1.51663439
427	183	412	222	488	121
-5.24158403	4.12162500	1.05839612	-2.23828558	-0.32455111	-0.16672642
47	35	475	425	266	148
-0.01245550	-0.37210738	-2.32010328	-2.13305403	-5.15686686	5.43417468
16	24	73	433	224	423
0.21409911	0.28866397	-1.30835635	-4.79752518	0.53579917	2.55969554
493	354	305	198	451	333
3.71914793	4.03802287	3.55845181	-1.65245412	-2.92240713	-3.73079592
317	495	455	379	487	209
0.05440191	4.21834045	-0.19364981	-2.29035946	-0.37021153	0.42804146
429	404	167	214	219	79
-2.71025052	-4.64822262	12.68516076	3.26082348	-4.01471234	0.16065482
227	169	258	482	344	235
0.23093311	-2.68422160	7.06335436	-3.05722906	-3.71728952	-3.05565664
367	374	417	468	324	243
6.11663059	8.24462468	-5.16250530	2.59821446	-1.18572548	-1.74323272
489	503	341	54	467	138
2.92836386	-2.14852284	-2.96133921	-0.40701818	4.87829205	-2.78669364
171	252	49	402	444	132

```
-5.24647887  -0.07782058   5.37016052 -10.45088991  -2.78321539  -0.50299750
        210          114          411          272          160          256
 2.31737038  -1.89465520   0.21762206  -1.51300412  -3.46305694  -0.71729756
        184          251           18          446           33          399
 1.49304038   0.38263885   0.20811819   0.10816415   4.54472726  -1.15304459
         25          208          145           52          376          292
-0.57832016   4.70914001   2.40933305  -3.60796407 -10.05466464   3.52807253
        291          124          380          116          346          223
-4.35426235   0.81408183  -6.31821853  -1.95261099   0.57079778  -4.95870059
        393          439
-0.09120371   4.14166510
```



>



Step 3: Prediction based on Model

```
> p1=predict(Linrearmodel,df_ts[,-14])
> p1
        2        10        17        21        29        30        36
25.223425 19.257386 20.674775 12.838593 20.083696 21.291397 23.726945
       38        40        50        56        58        64        69
22.902285 30.797274 17.251368 30.913450 32.902169 22.725778 17.216855
       70        71        72        74        82        83        88
20.631719 24.648721 21.265761 23.481506 26.982801 25.693125 25.583739
       90        95       100       104       115       119       126
```

```
30.587051 26.922639 31.790930 20.353845 25.054046 20.022465 22.838271
      128       129       130       151       153       156       173
15.694686 19.440735 14.571405 21.830587 21.865167 21.413989 22.690669
      176       178       179       181       187       188       191
30.120431 28.992722 31.120737 34.361177 35.393429 32.473385 30.163972
      203       204       206       212       213       215       216
36.225427 41.020393 22.237328 17.541213 23.044534  9.932291 24.263180
      217       220       225       228       229       238       249
27.001878 30.532667 37.940761 32.131952 34.579162 32.598333 21.439985
      257       268       280       285       288       290       293
36.873096 40.103650 34.466884 30.954940 26.729202 26.376521 31.196762
      303       307       308       313       314       319       320
28.067853 35.015200 32.211401 23.442115 25.795653 24.343334 21.352095
      326       336       339       340       349       350       365
24.533219 20.869470 22.191679 21.330925 27.481439 22.491879 38.067523
      368       370       408       409       410       414       416
10.837330 33.274834 19.956528 13.013395 19.120979 11.331606  8.920260
      422       424       436       440       441       443       447
18.225889 12.373367 13.140410 13.100627 12.711464 18.988445 17.807595
      448       456       465       469       471       476       478
18.368968 15.623419 20.183244 16.471463 19.866658 15.500701 11.137518
      480       490       494       504
21.583137  8.478326 20.492767 28.106226
```

Actuals and Predicted Difference##

```
> d_pred=data.frame(cbind(actuals=df_ts$medv,predicted=p1))
> d_pred
    actuals predicted
2      21.6 25.223425
10     18.9 19.257386
17     23.1 20.674775
```

21	13.6	12.838593
29	18.4	20.083696
30	21.0	21.291397
36	18.9	23.726945
38	21.0	22.902285
40	30.8	30.797274
50	19.4	17.251368
56	35.4	30.913450
58	31.6	32.902169
64	25.0	22.725778
69	17.4	17.216855
70	20.9	20.631719
71	24.2	24.648721
72	21.7	21.265761
74	23.4	23.481506
82	23.9	26.982801
83	24.8	25.693125
88	22.2	25.583739
90	28.7	30.587051
95	20.6	26.922639
100	33.2	31.790930
104	19.3	20.353845
115	18.5	25.054046
119	20.4	20.022465
126	21.4	22.838271
128	16.2	15.694686
129	18.0	19.440735
130	14.3	14.571405
151	21.5	21.830587
153	15.3	21.865167
156	15.6	21.413989

173	23.1	22.690669
176	29.4	30.120431
178	24.6	28.992722
179	29.9	31.120737
181	39.8	34.361177
187	50.0	35.393429
188	32.0	32.473385
191	37.0	30.163972
203	42.3	36.225427
204	48.5	41.020393
206	22.6	22.237328
212	19.3	17.541213
213	22.4	23.044534
215	23.7	9.932291
216	25.0	24.263180
217	23.3	27.001878
220	23.0	30.532667
225	44.8	37.940761
228	31.6	32.131952
229	46.7	34.579162
238	31.5	32.598333
249	24.5	21.439985
257	44.0	36.873096
268	50.0	40.103650
280	35.1	34.466884
285	32.2	30.954940
288	23.2	26.729202
290	24.8	26.376521
293	27.9	31.196762
303	26.4	28.067853
307	33.4	35.015200

308	28.2	32.211401
313	19.4	23.442115
314	21.6	25.795653
319	23.1	24.343334
320	21.0	21.352095
326	24.6	24.533219
336	21.1	20.869470
339	20.6	22.191679
340	19.0	21.330925
349	24.5	27.481439
350	26.6	22.491879
365	21.9	38.067523
368	23.1	10.837330
370	50.0	33.274834
408	27.9	19.956528
409	17.2	13.013395
410	27.5	19.120979
414	16.3	11.331606
416	7.2	8.920260
422	14.2	18.225889
424	13.4	12.373367
436	13.4	13.140410
440	12.8	13.100627
441	10.5	12.711464
443	18.4	18.988445
447	14.9	17.807595
448	12.6	18.368968
456	14.1	15.623419
465	21.4	20.183244
469	19.1	16.471463
471	19.9	19.866658

```
476    13.3 15.500701
478    12.0 11.137518
480    21.4 21.583137
490     7.0  8.478326
494    21.8 20.492767
504    23.9 28.106226
```

Step:4 Calculation of Accuracy

>mma=mean(apply(d_pred,1,min)/apply(d_pred,1,max))

> mma###.9599###

[1] 0.8814823

The accuracy is 88%. To improve the accuracy, we can generate other model, by eliminating non significant rows from first model

Step:5 Generation of another model by eliminating nonsignificant variables from model1

```
d_pre2=lm(medv~.,-zn-indus-age,data=df_tr)
> summary(d_pre2)

Call:
lm(formula = medv ~ ., data = df_tr, subset = -zn - indus - age)

Residuals:
     Min       1Q   Median       3Q      Max
-11.3414  -2.8386  -0.5527   1.8319  24.8775

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  37.733726   6.582713   5.732 2.44e-08 ***
crim         -0.100197   0.041366  -2.422  0.01603 *
zn            0.053385   0.017566   3.039  0.00259 **
indus         0.043918   0.071050   0.618  0.53697
```

```
chas         2.312570   1.138898   2.031  0.04320 *
nox        -16.004602   4.851099  -3.299  0.00109 **
rm           3.229035   0.549185   5.880 1.11e-08 ***
age          0.009800   0.017446   0.562  0.57471
dis         -1.496728   0.254832  -5.873 1.15e-08 ***
rad          0.250605   0.079071   3.169  0.00169 **
tax         -0.011231   0.004335  -2.591  0.01005 *
ptratio     -0.879841   0.158510  -5.551 6.33e-08 ***
black        0.008648   0.003251   2.660  0.00823 **
lstat       -0.570380   0.062755  -9.089  < 2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.633 on 296 degrees of freedom
Multiple R-squared:  0.743,    Adjusted R-squared:  0.7317
F-statistic: 65.82 on 13 and 296 DF,  p-value: < 2.2e-16
```

> **Step 5: Develop the prediction based on this new model**

>p2=predict(d_pre2,df_ts[,-14])

```
> p2
        2        10        17        21        29        30        36
24.948907 18.682799 20.763474 12.962873 19.765246 20.974552 23.937108
       38        40        50        56        58        64        69
23.065905 31.217304 17.008073 30.392244 32.733862 21.830284 17.270943
       70        71        72        74        82        83        88
20.626726 24.630870 21.397068 23.536380 26.871530 25.770534 25.753917
       90        95       100       104       115       119       126
30.399546 27.332568 31.525290 20.672933 25.438562 20.527072 23.224138
      128       129       130       151       153       156       173
16.357123 19.792838 15.199937 21.762328 21.338376 20.661096 23.085772
      176       178       179       181       187       188       191
```

```
30.180030 29.045681 31.019285 33.843857 34.829639 32.922173 30.007799
      203       204       206       212       213       215       216
36.229726 41.059185 22.453079 16.746522 22.238562 10.234661 24.353795
      217       220       225       228       229       238       249
26.167427 29.432356 37.250587 31.835612 34.119511 32.015849 20.843867
      257       268       280       285       288       290       293
36.905714 39.470451 34.395679 30.944366 26.655603 26.152836 31.729262
      303       307       308       313       314       319       320
28.076202 34.868665 32.282202 23.669773 25.891067 24.308397 21.304560
      326       336       339       340       349       350       365
24.243894 20.416739 21.979923 21.123166 27.220868 21.807199 36.124761
      368       370       408       409       410       414       416
12.562653 32.551798 20.663788 13.576242 19.469731 12.482311  9.286831
      422       424       436       440       441       443       447
18.429278 13.009377 13.158131 13.258154 12.910384 18.899125 17.738695
      448       456       465       469       471       476       478
18.276832 15.773624 20.168298 16.772777 19.909209 15.725148 11.708416
      480       490       494       504
21.920130  9.640224 20.973068 28.177519
```

Step:6 Calculation of Accuracy

```
>d_pm2=data.frame(cbind(actuals=df_ts$medv,predicted=p2))
> d_pm2
    actuals predicted
2      21.6 24.948907
10     18.9 18.682799
17     23.1 20.763474
21     13.6 12.962873
29     18.4 19.765246
30     21.0 20.974552
```

36	18.9	23.937108
38	21.0	23.065905
40	30.8	31.217304
50	19.4	17.008073
56	35.4	30.392244
58	31.6	32.733862
64	25.0	21.830284
69	17.4	17.270943
70	20.9	20.626726
71	24.2	24.630870
72	21.7	21.397068
74	23.4	23.536380
82	23.9	26.871530
83	24.8	25.770534
88	22.2	25.753917
90	28.7	30.399546
95	20.6	27.332568
100	33.2	31.525290
104	19.3	20.672933
115	18.5	25.438562
119	20.4	20.527072
126	21.4	23.224138
128	16.2	16.357123
129	18.0	19.792838
130	14.3	15.199937
151	21.5	21.762328
153	15.3	21.338376
156	15.6	20.661096
173	23.1	23.085772
176	29.4	30.180030
178	24.6	29.045681

179	29.9	31.019285
181	39.8	33.843857
187	50.0	34.829639
188	32.0	32.922173
191	37.0	30.007799
203	42.3	36.229726
204	48.5	41.059185
206	22.6	22.453079
212	19.3	16.746522
213	22.4	22.238562
215	23.7	10.234661
216	25.0	24.353795
217	23.3	26.167427
220	23.0	29.432356
225	44.8	37.250587
228	31.6	31.835612
229	46.7	34.119511
238	31.5	32.015849
249	24.5	20.843867
257	44.0	36.905714
268	50.0	39.470451
280	35.1	34.395679
285	32.2	30.944366
288	23.2	26.655603
290	24.8	26.152836
293	27.9	31.729262
303	26.4	28.076202
307	33.4	34.868665
308	28.2	32.282202
313	19.4	23.669773
314	21.6	25.891067

319	23.1	24.308397
320	21.0	21.304560
326	24.6	24.243894
336	21.1	20.416739
339	20.6	21.979923
340	19.0	21.123166
349	24.5	27.220868
350	26.6	21.807199
365	21.9	36.124761
368	23.1	12.562653
370	50.0	32.551798
408	27.9	20.663788
409	17.2	13.576242
410	27.5	19.469731
414	16.3	12.482311
416	7.2	9.286831
422	14.2	18.429278
424	13.4	13.009377
436	13.4	13.158131
440	12.8	13.258154
441	10.5	12.910384
443	18.4	18.899125
447	14.9	17.738695
448	12.6	18.276832
456	14.1	15.773624
465	21.4	20.168298
469	19.1	16.772777
471	19.9	19.909209
476	13.3	15.725148
478	12.0	11.708416
480	21.4	21.920130

```
490     7.0  9.640224
494    21.8 20.973068
504    23.9 28.177519
> mma=mean(apply(d_pm2,1,min)/apply(d_pm2,1,max))
> mma
####.96026###
```

So now the accuracy is 96%. So we can develop such models to increase the accuracy.

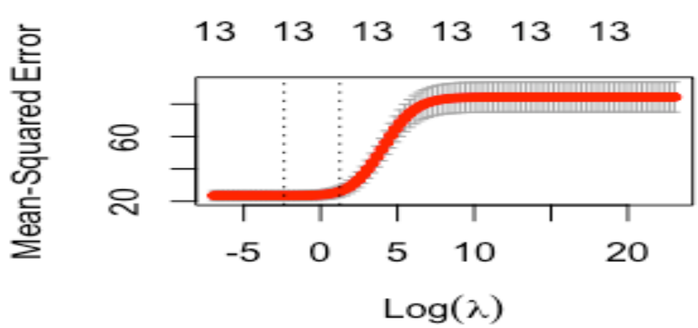
Step-7:Lasso and Ridge regression

Ridge and **Lasso regression** are powerful techniques generally used for creating parsimonious models in presence of a 'large' number of features. Here 'large' can typically mean either of two things: Large enough to enhance the tendency of a model to overfit (as low as 10 variables might cause overfitting. The key **difference between** these two is the penalty term. **Ridge regression** adds “squared magnitude” of coefficient as penalty term to the loss function. ... **Lasso Regression** (Least Absolute Shrinkage and Selection Operator) adds “absolute value of magnitude” of coefficient as penalty term to the loss function.
Ridge regression puts constraint on the coefficients (w). The penalty term (**lambda**) regularizes the coefficients such that if the coefficients take large values the optimization function is penalized. So, **ridge regression** shrinks the coefficients and it helps to reduce the model complexity and multi-collinearity

```
>install.packages("glmnet")
>library(glmnet) ## Import this library for Ridge and lasso Regression###
>lambda=10^seq(-3,10,length.out = 100)
> lambda
  [1] 1.000000e-03 1.353048e-03 1.830738e-03 2.477076e-03 3.351603e-03
  [6] 4.534879e-03 6.135907e-03 8.302176e-03 1.123324e-02 1.519911e-02
 [11] 2.056512e-02 2.782559e-02 3.764936e-02 5.094138e-02 6.892612e-02
 [16] 9.326033e-02 1.261857e-01 1.707353e-01 2.310130e-01 3.125716e-01
```

```
[21] 4.229243e-01 5.722368e-01 7.742637e-01 1.047616e+00 1.417474e+00
[26] 1.917910e+00 2.595024e+00 3.511192e+00 4.750810e+00 6.428073e+00
[31] 8.697490e+00 1.176812e+01 1.592283e+01 2.154435e+01 2.915053e+01
[36] 3.944206e+01 5.336699e+01 7.220809e+01 9.770100e+01 1.321941e+02
[41] 1.788650e+02 2.420128e+02 3.274549e+02 4.430621e+02 5.994843e+02
[46] 8.111308e+02 1.097499e+03 1.484968e+03 2.009233e+03 2.718588e+03
[51] 3.678380e+03 4.977024e+03 6.734151e+03 9.111628e+03 1.232847e+04
[56] 1.668101e+04 2.257020e+04 3.053856e+04 4.132012e+04 5.590810e+04
[61] 7.564633e+04 1.023531e+05 1.384886e+05 1.873817e+05 2.535364e+05
[66] 3.430469e+05 4.641589e+05 6.280291e+05 8.497534e+05 1.149757e+06
[71] 1.555676e+06 2.104904e+06 2.848036e+06 3.853529e+06 5.214008e+06
[76] 7.054802e+06 9.545485e+06 1.291550e+07 1.747528e+07 2.364489e+07
[81] 3.199267e+07 4.328761e+07 5.857021e+07 7.924829e+07 1.072267e+08
[86] 1.450829e+08 1.963041e+08 2.656088e+08 3.593814e+08 4.862602e+08
[91] 6.579332e+08 8.902151e+08 1.204504e+09 1.629751e+09 2.205131e+09
[96] 2.983647e+09 4.037017e+09 5.462277e+09 7.390722e+09 1.000000e+10
```

Set alpha=0 in Ridge###



>ridgmod=cv.glmnet(x,y,alpha=0,lambda=lambda,standardize=TRUE,nfold=10)

Step-8 Calculation of Accuracy using Ridge Mode

>d_ridge=data.frame(cbind(actuals=df_ts$medv,predicted=p4))## Data Farme##

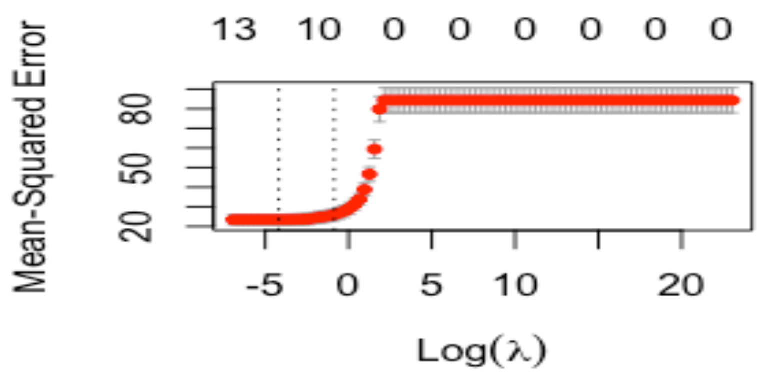
> mma=mean(apply(d_ridge,1,min)/apply(d_ridge,1,max))

> mma

[1] 0.8699422

Set alpha=1in Lasso###

```
>lassomod=cv.glmnet(x,y,alpha=1,lambda=lambda,standardize=TRUE,nfolds=10
> plot(lassomod)
```



Step-9 Calculation of Accuracy using Lasso Mode

```
> p4lasso=predict(lassomod,as.matrix(df_ts[,-14]))
> d_lasso=data.frame(cbind(actuals=df_ts$medv,predicted=p4lasso))
> mma=mean(apply(d_lasso,1,min)/apply(d_lasso,1,max))
> mma
[1] 0.8533709
```

So the accuracy calculated by Simple Regression,Ridge Regression and Lasso Regression are calculated. Based on that simple regression gives maximum accuracy.

Summary: The model development is done based on the various factors based on tvalue and pvalue. Non significant variables were detected and another model was developed. The prediction is done based on this model. So for Boston Data set as there are no missing values,the simple linear regression turns out be best for maximum accuracy(96%)

R-Script for the case study

```
####Project-1####
##Boston###
library(MASS)
d_b=Boston
View(d_b)
head(d_b)### Displays Ist 6 Observations (Rows)####
scatter.smooth(x=d_b$age, y=d_b$medv, main="age ~ medv")  # scatterplot
boxplot(d_b$medv, main="medv", sub=paste("Outlier rows: ",
boxplot.stats(d_b$medv)$out))  # box plot for medv
c=cor(d_b$age, d_b$medv)  # calculate correlation between age and medv ##
##-0.3769546##

### Build Linear Regression Model###
linearMod = lm(age ~ medv, data=d_b)  # build linear regression model on full data
print(linearMod)
summary(linearMod)

modelSummary =summary(linearMod)  # capture model summary as an object
modelCoeffs = modelSummary$coefficients  # model coefficients
beta.estimate = modelCoeffs["medv", "Estimate"]  # get beta estimate for speed
std.error = modelCoeffs["medv", "Std. Error"]  # get std.error for speed
std.error
t_value =beta.estimate/std.error  # calc t statistic
t_value
p_value = 2*pt(-abs(t_value), df=nrow(cars)-ncol(cars))  # calc p Value
f_statistic = linearMod$fstatistic[1]  # fstatistic
f_statistic
f = summary(linearMod)$fstatistic  # parameters for model p-value calc
f
model_p =pf(f[1], f[2], f[3], lower=FALSE)
```

```
model_p

###Training Data Set with 80% Values****
s=sample(nrow(d_b),.80*nrow(d_b))###Sampling###
s
#### Preapre Training Data Set###
df_tr=d_b[s,]
df_tr
####Prepare Testing Data Set###
df_ts=d_b[-s,]###-sign stands for remaimng data##
df_ts
###Find out Missing Values###
sum(is.na(d_b))### Missing Values Calculation ####

#### Find Structure of the data ####
str(d_b)
### Filtering---Print Records who has age between 20 to 40--preapre New Data Frame####
d_b1=subset(d_b,'age'>=20 &'age'<=40,selec=c(age,medv))
d_b1
head(d_b1)
boxplot(d_b$age)
range(d_b$age)
diff(range(d_b$age))
#### Print Histogram ###
hist(d_b$age,binwidth=10)
###### Perform Univarite Statiscal ANalysis###
boxplot(d_b$crim)
boxplot(d_b$medv)
boxplot(d_b$lstat)
###### Find Out Kurtosis and Skewness###
library(moments)
```

```
### Kurtosis###
skewness(d_b$crim)
##5.2076##
skewness(d_b$medv)
##1.104###
skewness(d_b$lstat)
###.90377###
kurtosis(d_b$crim)
###39.75###
kurtosis(d_b$medv)
###4.46##
kurtosis(d_b$lstat)
###3.4765###


#### Plotting of Crim,medv and lstat###
library(ggplot2)
ggplot(d_b,aes(x=d_b$crim))+geom_density()
ggplot(d_b,aes(x=d_b$medv))+geom_density()
ggplot(d_b,aes(x=d_b$lstat))+geom_density()

### Observe the graph..Crim is highally skewed Do Log Transformation###
d_log$crim=log1p(d_b$crim)
#### Replace original Crim with log Transformed crim...You can createnew dataframe as well as can keep the same###
d_b$crim=log1p(d_b$crim)
d_b$crim #### New log transformed Crim###
#ggplot(d_b,aes(x=d_b$crim))+geom_density()#
### Statiscal Analysis For Log Transformed Variables:Crim###
kurtosis(d_b$crim)
###3.487747###
```

```
skewness(d_b$crim)
## 1.265435###

### Pi-Plot 5 Prices####

d_b[order(-d_b$medv),]
d_order=d_b[order(-d_b$medv),]#### Decending Order###
head(d_order)
### Order Medv###
sum(duplicared(d_order$medv))### How many duplicated values in medv##
length(unique(d_b$medv))

### Order zn###
sum(duplicated(d_order$zn))
length(unique(d_b$zn))

#### zone and price table by method-1 using group_by###
tapply(d_b$zn, d_b$medv, mean)

#### zone and price table by method-2 which is good for Printing using ###
library(dplyr)    # alternatively, this also loads %>%
d_b%>% group_by(zn)%>%summarise(mean(medv))###  It will Print Table##


###### Prediction####
head(df_tr)
head(d_b)
df_tr=d_b[s,]
df_ts=d_b[-s,]
Linrearmodel=lm(medv~.,data=df_tr)
summary(Linrearmodel)
```

```
###here crim ,zn,dis,rad are significant ***###
###Plot Residual###
Linrearmodel$residuals
plot(df_tr$rm,Linrearmodel$residuals)
ggplot(df_tr,aes(lstat,Linrearmodel$residuals))+geom_point(aes(col="red"))+geom_smoot
h()
#### Prediction###
p1=predict(Linrearmodel,df_ts[,-14])
p1

### Actuals and Prdicted Difference##
d_pred=data.frame(cbind(actuals=df_ts$medv,predicted=p1))
d_pred
mma=mean(apply(d_pred,1,min)/apply(d_pred,1,max))
mma###.9599###
####Linmodel 2 ---elimniting rows####
d_pre2=lm(medv~.,-zn-indus-age,data=df_tr)
summary(d_pre2)
p2=predict(d_pre2,df_ts[,-14])
p2
d_pm2=data.frame(cbind(actuals=df_ts$medv,predicted=p2))
d_pm2
mma=mean(apply(d_pm2,1,min)/apply(d_pm2,1,max))
mma####.96026###

#### Remove rm#####
#### VIF ####
library(car)
vif(d_pre2)###with model-2##
##model 3 after removing columns with high multicololineranit..columns VIF>5###
d_pre3=lm(medv~.,-crim-rad-tax-zn-indus-age,data=df_tr)
```

```
summary(d_pre3)
p3=predict(d_pre3,df_ts[,-14])
p3
d_pred3=data.frame(cbind(actuals=df_ts$medv,predicted=p3))
mma=mean(apply(d_pred3,1,min)/apply(d_pred3,1,max))
mma
###0.9602686###

#####Ridge Regression####mma



#####Ridge Regression####
library(glmnet)
install.packages("glmnet")
library(glmnet)
lambda=10^seq(-3,10,length.out = 100)
lambda
x=as.matrix(df_tr[,-14])
y=as.matrix(df_ts[,14])



ridgmod=cv.glmnet(x,y,alpha=0,lambda=lambda,standardize=TRUE,nfold=10)
ridgmod=cv.glmnet(x,y,alpha=0,lambda=lambda,standardize=TRUE,nfolds=10)
ridgmod=cv.glmnet(x,y,alpha=0,lambda=lambda,standardize=TRUE,nfolds=10)
x=as.matrix(df_tr[,-14])
y=as.matrix(df_tr[,14])
ridgmod=cv.glmnet(x,y,alpha=0,lambda=lambda,standardize=TRUE,nfolds=10)
plot(ridgmod)
p4=predict(ridgmod,as.matrix(df_ts[,-14]))
p4
d_ridge=data.frame(cbind(actuals=df_ts$medv,predicted=p4))
```

```
mma=mean(apply(d_ridge,1,min)/apply(d_ridge,1,max))
mma
lassomod=cv.glmnet(x,y,alpha=1,lambda=lambda,standardize=TRUE,nfolds=10)
plot(lassomod)
p4lasso=predict(lassomod,as.matrix(df_ts[,-14]))
d_lasso=data.frame(cbind(actuals=df_ts$medv,predicted=p4lasso))
mma=mean(apply(d_lasso,1,min)/apply(d_lasso,1,max))
mma
```