Machine Learning Models for Predicting House Price of Boston

Provakar Ghose

2023-03-14

Introduction & Objective:

The purpose of this project to find the best regression model that could be used for predicting the median value of owner-occupied homes in various suburbs of Boston. I would be using the data set called "Boston-Housing_randomized - 9" dataset for this. The BostonHousing_randomized – 9 dataset is a popular dataset used in machine learning for regression tasks. The dataset contains information about different properties of houses in Boston, such as the number of rooms, crime rate, and distance from employment centers, and the target variable is the median value of owner-occupied homes in thousands of dollars. I will be using three regression models - Linear Regression, SVR, and Random Forest Regression model will be applying them to predict the housing prices in Boston. Also, to evaluate and finding best-performing model R-squared and Mean Squared Error (MSE) matrix have been used.

Reason for choosing the Models:

Linear regression model is being used initially to determine what is the best variable when MEDV is the dependent variable. Once the independent variables are determined, i will be using other models for predicting the housing values.

SVR(Support vector regression) can be used for this Boston housing data set as the data is not linear.

Random forest regression is a powerful machine learning algorithm that is used for a wide range of prediction tasks

Loading the Libraries:

```
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```
library(ggplot2)
library(caTools)
library(e1071)
library(rpart)
library(randomForest)
```

```
## randomForest 4.7-1.1
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(readxl)
```

Loading the Dataset:

```
getwd()
```

```
## [1] "C:/Users/dibak/Desktop/R Script/DM Mid Project"
```

```
setwd("C:/Users/dibak/Desktop/R Script/DM Mid Project")
Boston<- read_excel("C:\\Users\\dibak\\Desktop\\R Script\\DM Mid Project\\BostonHousing_randomized_9.xls
```

Finding Missing Values in the Dataset:

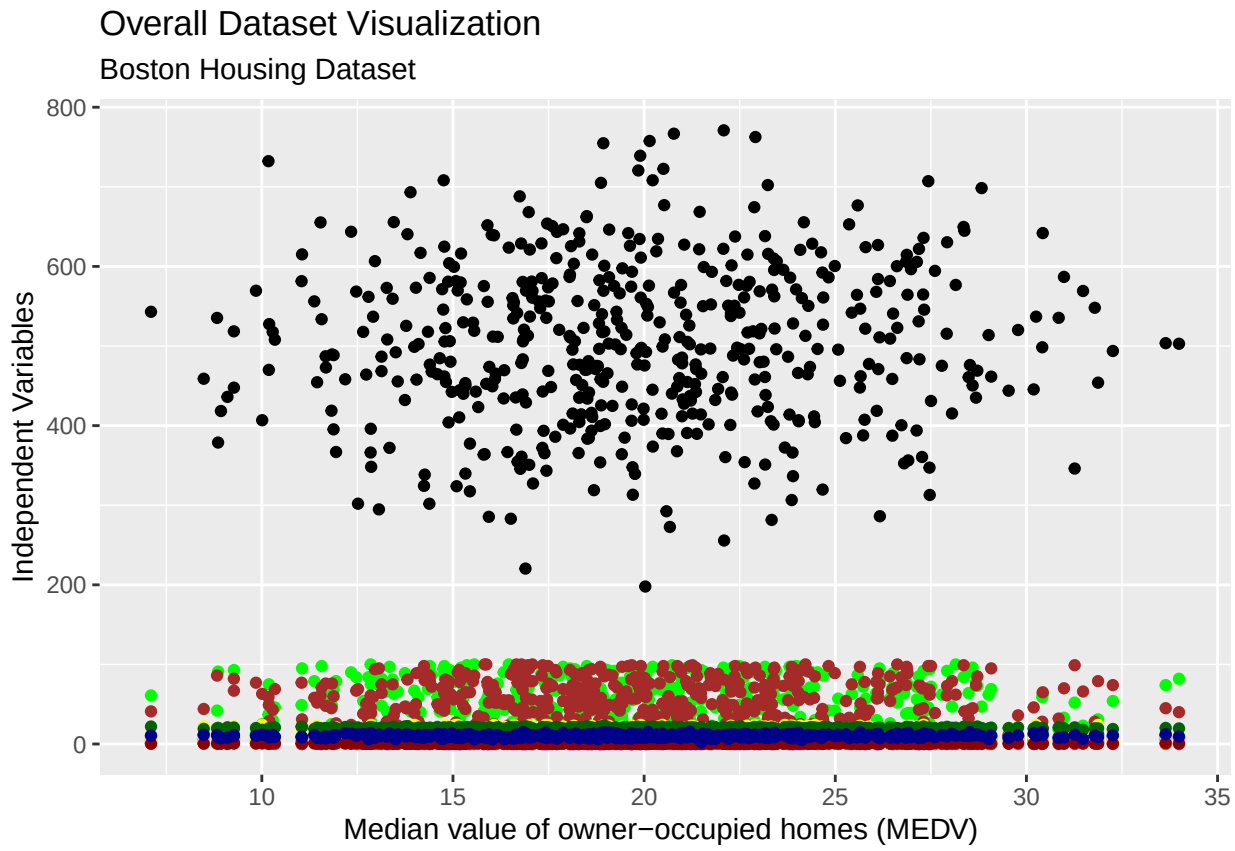
```
sum(is.na(Boston))
```

```
## [1] 0
```

We found there is no missing value in our Dataset Boston

Visualization of Overall Dataset:

```
ggplot(Boston, aes(x = medv)) +
  geom_point(aes(y = crim), colour = 'red') +
  geom_point(aes(y = zn), colour = 'green') +
  geom_point(aes(y = indus), colour = 'blue') +
  geom_point(aes(y = chas), colour = 'purple') +
  geom_point(aes(y = nox), colour = 'orange') +
  geom_point(aes(y = rm), colour = 'pink') +
  geom_point(aes(y = age), colour = 'brown') +
  geom_point(aes(y = dis), colour = 'gray') +
  geom_point(aes(y = rad), colour = 'yellow') +
  geom_point(aes(y = tax), colour = 'black') +
  geom_point(aes(y = ptratio), colour = 'dark green') +
  geom_point(aes(y = black), colour = 'dark red') +
  geom_point(aes(y = lstat), colour = 'dark blue') +
  labs(x = "Median value of owner-occupied homes (MEDV) ", y = "Independent Variables",
       title = "Overall Dataset Visualization",
       subtitle = "Boston Housing Dataset")
```



This visualization illustrate overall data position of our Dataset.

Correlation Summary :

```
cor(Boston)
```

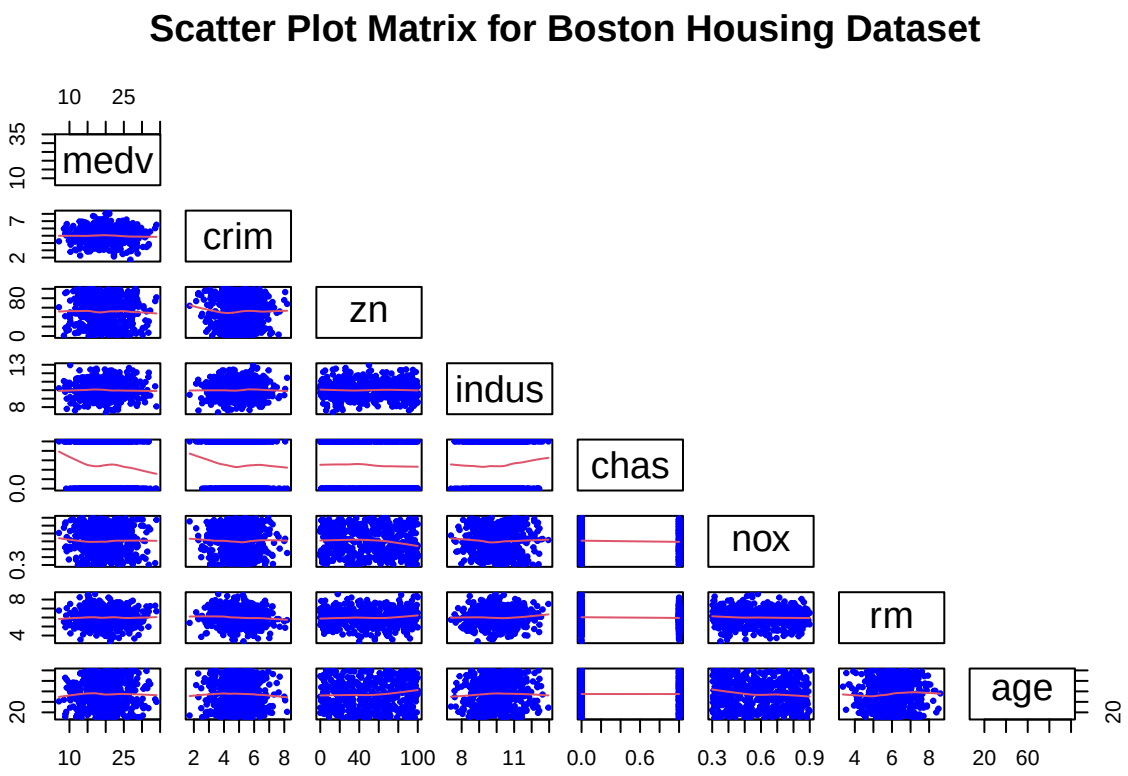
```
##                crim           zn        indus         chas           nox
## crim     1.000000000  0.008848415  0.040200954 -0.040058187 -0.0079385324
## zn       0.008848415  1.000000000  0.001717435 -0.036011889 -0.1067904769
## indus    0.040200954  0.001717435  1.000000000  0.051905649 -0.0052799505
## chas    -0.040058187 -0.036011889  0.051905649  1.000000000 -0.0405190286
## nox     -0.007938532 -0.106790477 -0.005279951 -0.040519029  1.0000000000
## rm      -0.073854866  0.062767464  0.012268534 -0.047887689 -0.0483361541
## age     -0.020776873  0.090743294  0.019470887 -0.001207518 -0.1047313250
## dis      0.006128499 -0.036501129 -0.033074010  0.035561418 -0.0128303946
## rad      0.003574835 -0.068270664  0.006536469  0.007951057 -0.0108939446
## tax      0.024263122  0.025314066  0.024968109  0.013933426  0.0262165173
## ptratio  0.028755277  0.058270461 -0.017156375  0.035115980 -0.0006231371
## black    0.048255171  0.011311314  0.018703480  0.032170248 -0.0386997654
## lstat    0.032912373  0.026022907 -0.015652977  0.002088604  0.0298933006
## medv    -0.033491220 -0.007202397 -0.019616345 -0.073673565  0.0098534585
##                  rm          age          dis          rad         tax
## crim    -0.07385487 -0.020776873  0.006128499  0.003574835  0.02426312
## zn       0.06276746  0.090743294 -0.036501129 -0.068270664  0.02531407
## indus    0.01226853  0.019470887 -0.033074010  0.006536469  0.02496811
## chas    -0.04788769 -0.001207518  0.035561418  0.007951057  0.01393343
## nox     -0.04833615 -0.104731325 -0.012830395 -0.010893945  0.02621652
## rm       1.00000000  0.068221509 -0.016323197  0.088336634 -0.02773658
## age      0.06822151  1.000000000  0.049135579  0.003137082 -0.01425844
## dis     -0.01632320  0.049135579  1.000000000  0.037995835  0.02981028
## rad      0.08833663  0.003137082  0.037995835  1.000000000  0.03335664
## tax     -0.02773658 -0.014258436  0.029810276  0.033356644  1.00000000
## ptratio -0.06381415  0.029981903 -0.017034834 -0.132543114 -0.05406864
## black   -0.06440492  0.081350731  0.037419359 -0.089351337 -0.07116298
## lstat    0.02057637  0.024214792 -0.045876809 -0.031351858 -0.05973254
## medv    -0.01097985 -0.005295311 -0.012338321 -0.005700536  0.05155010
##               ptratio       black        lstat         medv
## crim     0.0287552766  0.04825517  0.032912373 -0.033491220
## zn       0.0582704607  0.01131131  0.026022907 -0.007202397
## indus   -0.0171563749  0.01870348 -0.015652977 -0.019616345
## chas     0.0351159796  0.03217025  0.002088604 -0.073673565
## nox     -0.0006231371 -0.03869977  0.029893301  0.009853458
## rm      -0.0638141532 -0.06440492  0.020576371 -0.010979854
## age      0.0299819025  0.08135073  0.024214792 -0.005295311
## dis     -0.0170348340  0.03741936 -0.045876809 -0.012338321
## rad     -0.1325431136 -0.08935134 -0.031351858 -0.005700536
## tax     -0.0540686370 -0.07116298 -0.059732541  0.051550099
## ptratio  1.0000000000  0.05095949  0.119365663  0.026625475
## black    0.0509594851  1.00000000 -0.041712855 -0.061723134
## lstat    0.1193656626 -0.04171285  1.000000000  0.070530755
## medv     0.0266254748 -0.06172313  0.070530755  1.000000000
```

Above Correlation summary provides us relationship percentage between independent variables and dependent variables. Here “lstat” found with highest positive and “chas” found highest negative correlation with our target variable “medv”.

Scatter plot matrix to visualize data Correlation:

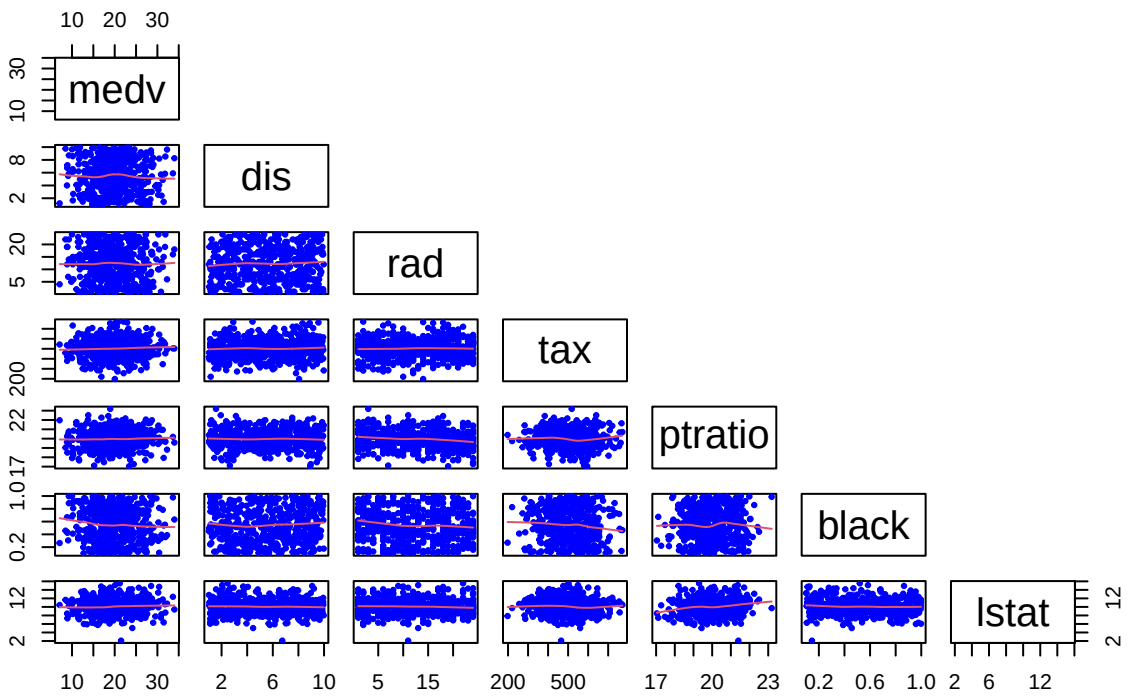
Scatter plot matrix 1:

```
pairs(data=Boston,~medv+crim+zn+indus+chas+nox+rm+age,
        main = "Scatter Plot Matrix for Boston Housing Dataset",
      upper.panel = NULL,
      lower.panel = panel.smooth,
      diag.panel = NULL,
      pch = 20,
      col = "blue",
      cex = 0.7)
```



Scatter plot matrix 2:

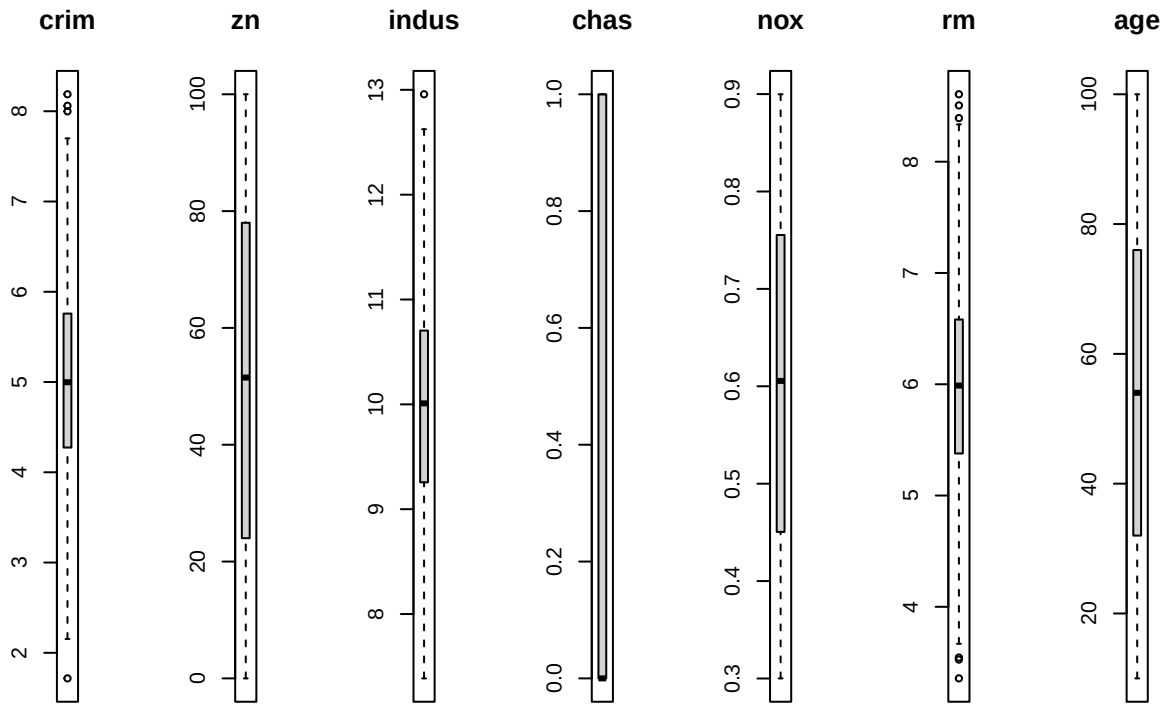
```
pairs(data=Boston,~medv+dis+rad+tax+ptratio+black+lstat,
        main = "Scatter Plot Matrix for Boston Housing Dataset",
      upper.panel = NULL,
      lower.panel = panel.smooth,
      diag.panel = NULL,
      pch = 20,
      col = "blue",
      cex = 0.7)
```



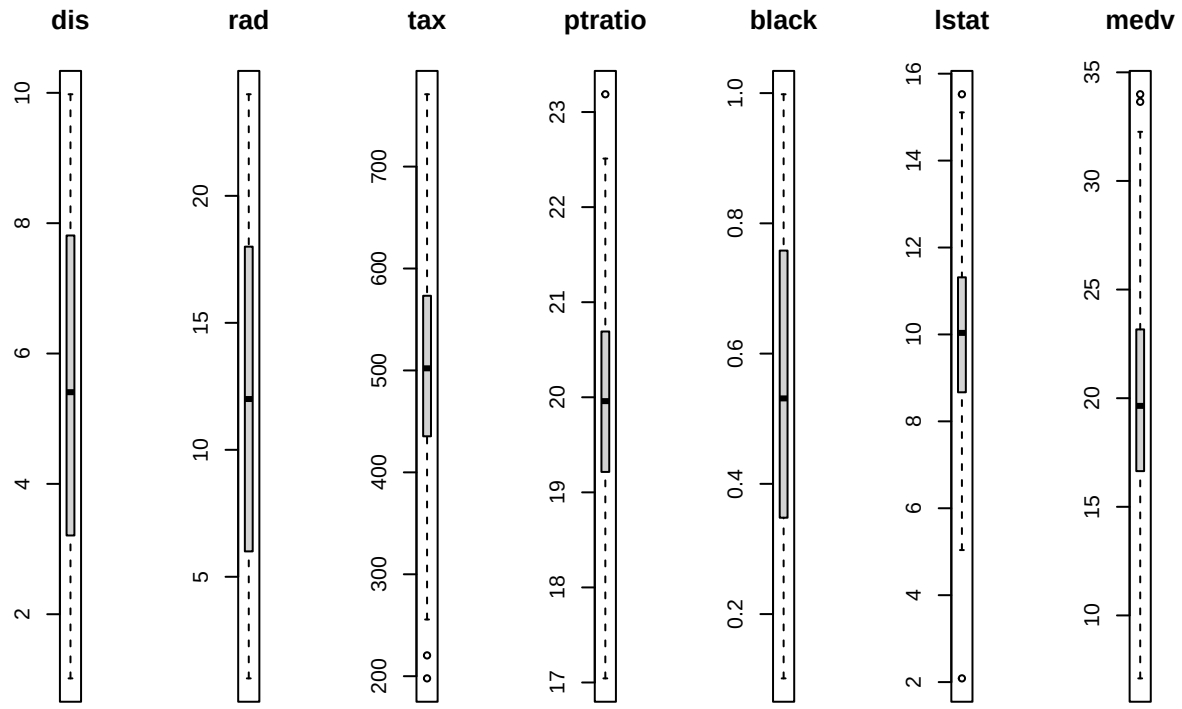
Scatter plot matrix 1 and 2 used to see correlation of all independent variables with target variable "medv'. Here we found data are highly scattered.

Checking for Outliers:

```
# Part 1
par(mfrow=c(1,7))
for(i in 1:7) {
  boxes <- boxplot(Boston[,i], main=names(Boston)[i])}
```




```
# Part 2
par(mfrow=c(1,7))
for(i in 8:14) {
  boxes <- boxplot(Boston[,i], main=names(Boston)[i])}
```



By outliers checking we found independent variables crim, indus, rm, tax, ptratio, and lstat have outliers means data points that are significantly different from the other data points in the dataset. Though they have a significant impact on the statistical analysis of the dataset because they can distort the mean, standard deviation, and other measures of central tendency, in other cases, outliers may be legitimate data points that reflect the true nature of the data, and therefore should not be removed.

Feature scaling of the Dataset

```
data_x_vars <- Boston[ , -14]
data_x_vars_scaled <- as.data.frame(scale(data_x_vars, scale=TRUE, center=TRUE))
data_scaled <- cbind(data_x_vars_scaled , medv = Boston[ , 14])
```

Splitting the Dataset by 70-30 ratio

```
set.seed(42)
train_index <- createDataPartition(data_scaled$medv, p = 0.7, list = FALSE)
scaled_train_data <- data_scaled[train_index, ]
scaled_test_data <- data_scaled[-train_index, ]
```

In order to identify the best regression model, i would be splitting the data set 70:30 which means that 70% of the data is used for training the model, while the remaining 30% will be used for testing the model's

performance.The reason for choosing this ratio is to ensure that I have enough data to train my model while still having a sufficient amount of data to test it and evaluate its performance.

Fitting Regression Models to Dataset:

Linear Regression Model for variable selection:

```
lm_model_All <- lm(medv ~ ., data = scaled_train_data)
summary(lm_model_All)
```

```
##
## Call:
## lm(formula = medv ~ ., data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7193  -3.0900  -0.3773   3.4995  13.3538
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.844677   0.263309  75.366   <2e-16 ***
## crim        -0.041916   0.276306  -0.152   0.8795
## zn           0.180316   0.260638   0.692   0.4895
## indus       -0.187781   0.255351  -0.735   0.4626
## chas        -0.373746   0.265632  -1.407   0.1603
## nox          0.166583   0.268994   0.619   0.5361
## rm          -0.006571   0.274233  -0.024   0.9809
## age         -0.239592   0.266792  -0.898   0.3698
## dis         -0.133430   0.266884  -0.500   0.6174
## rad          0.099774   0.266565   0.374   0.7084
## tax          0.172780   0.263145   0.657   0.5119
## ptratio      0.175032   0.268995   0.651   0.5157
## black       -0.417876   0.269794  -1.549   0.1223
## lstat        0.498668   0.265985   1.875   0.0617 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.928 on 342 degrees of freedom
## Multiple R-squared:  0.03832,    Adjusted R-squared:  0.001767
## F-statistic: 1.048 on 13 and 342 DF,  p-value: 0.4043
```

Improving lm Model performance by Removing less significant independent variable (rm):

```
lm_model_rm <- lm(medv ~ crim+zn+indus+chas+nox+age+dis+rad+tax+ptratio+black+lstat,
                  data = scaled_train_data)
summary(lm_model_rm)
```

```
##
## Call:
## lm(formula = medv ~ crim + zn + indus + chas + nox + age + dis +
##     rad + tax + ptratio + black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7242  -3.1039  -0.3819   3.4936  13.3482
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.84471    0.26292  75.478   <2e-16 ***
## crim        -0.04157    0.27552  -0.151   0.8802
## zn           0.18007    0.26005   0.692   0.4891
## indus       -0.18765    0.25492  -0.736   0.4622
## chas        -0.37354    0.26511  -1.409   0.1597
## nox          0.16666    0.26858   0.621   0.5353
## age         -0.23982    0.26624  -0.901   0.3683
## dis         -0.13278    0.26510  -0.501   0.6168
## rad          0.09944    0.26581   0.374   0.7086
## tax          0.17285    0.26274   0.658   0.5111
## ptratio      0.17520    0.26851   0.652   0.5145
## black       -0.41750    0.26895  -1.552   0.1215
## lstat        0.49845    0.26545   1.878   0.0613 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.921 on 343 degrees of freedom
## Multiple R-squared:  0.03832,    Adjusted R-squared:  0.004676
## F-statistic: 1.139 on 12 and 343 DF,  p-value: 0.3272
```

Improving lm Model performance by Removing less significant independent variable (crim):

```
lm_model_crim <- lm(medv ~zn+indus+chas+nox+age+dis+rad+tax+ptratio+black+lstat,
                    data = scaled_train_data)
summary(lm_model_crim)
```

```
##
## Call:
## lm(formula = medv ~ zn + indus + chas + nox + age + dis + rad +
##     tax + ptratio + black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.7041  -3.0795  -0.4035   3.5050  13.2901
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.84557    0.26249  75.606   <2e-16 ***
```

```
## zn           0.17915    0.25961   0.690   0.4906
## indus       -0.19084    0.25367  -0.752   0.4524
## chas        -0.36986    0.26360  -1.403   0.1615
## nox          0.16772    0.26811   0.626   0.5320
## age         -0.23895    0.26580  -0.899   0.3693
## dis         -0.13541    0.26415  -0.513   0.6085
## rad          0.09837    0.26534   0.371   0.7111
## tax          0.17132    0.26217   0.653   0.5139
## ptratio      0.17468    0.26811   0.652   0.5151
## black       -0.41923    0.26832  -1.562   0.1191
## lstat        0.49754    0.26500   1.878   0.0613 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.914 on 344 degrees of freedom
## Multiple R-squared:  0.03826,    Adjusted R-squared:  0.007503
## F-statistic: 1.244 on 11 and 344 DF,  p-value: 0.2563
```

Improving lm Model performance by Removing less significant independent variable (dis):

```
lm_model_dis <- lm(medv ~zn+indus+chas+nox+age+rad+tax+ptratio+black+lstat,
                   data = scaled_train_data)
summary(lm_model_dis)
```

```
##
## Call:
## lm(formula = medv ~ zn + indus + chas + nox + age + rad + tax +
##     ptratio + black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.4913  -3.1145  -0.4864   3.4207  13.1614
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 19.83821    0.26181  75.773   <2e-16 ***
## zn           0.18093    0.25931   0.698   0.4858
## indus       -0.18314    0.25296  -0.724   0.4696
## chas        -0.37626    0.26303  -1.431   0.1535
## nox          0.16731    0.26782   0.625   0.5326
## age         -0.24763    0.26497  -0.935   0.3507
## rad          0.09621    0.26502   0.363   0.7168
## tax          0.16382    0.26149   0.626   0.5314
## ptratio      0.18070    0.26756   0.675   0.4999
## black       -0.42588    0.26772  -1.591   0.1126
## lstat        0.50345    0.26447   1.904   0.0578 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 4.908 on 345 degrees of freedom
## Multiple R-squared:  0.03752,    Adjusted R-squared:  0.009624
## F-statistic: 1.345 on 10 and 345 DF,  p-value: 0.2049
```

Improving lm Model performance by Removing less significant independent variable (rad):

```
lm_model_rad <- lm(medv ~zn+indus+chas+nox+age+tax+ptratio+black+lstat,
                   data = scaled_train_data)
summary(lm_model_rad)
```

```
##
## Call:
## lm(formula = medv ~ zn + indus + chas + nox + age + tax + ptratio +
##     black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.5921  -3.0808  -0.3629   3.3970  13.2480
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8428     0.2612  75.972   <2e-16 ***
## zn            0.1733     0.2581   0.671   0.5025
## indus        -0.1844     0.2526  -0.730   0.4658
## chas         -0.3731     0.2626  -1.421   0.1562
## nox           0.1639     0.2673   0.613   0.5401
## age          -0.2479     0.2646  -0.937   0.3495
## tax           0.1634     0.2612   0.626   0.5318
## ptratio       0.1682     0.2650   0.635   0.5261
## black        -0.4319     0.2669  -1.618   0.1065
## lstat         0.5053     0.2641   1.913   0.0565 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.902 on 346 degrees of freedom
## Multiple R-squared:  0.03715,    Adjusted R-squared:  0.01211
## F-statistic: 1.483 on 9 and 346 DF,  p-value: 0.1524
```

Improving lm Model performance by Removing less significant independent variable (ptratio):

```
lm_model_pt <- lm(medv ~zn+indus+chas+nox+age+tax+black+lstat,
                   data = scaled_train_data)
summary(lm_model_pt)
```

```
##
```

```
## Call:
## lm(formula = medv ~ zn + indus + chas + nox + age + tax + black +
##     lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.2484  -3.0681  -0.3958   3.4003  13.1966
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8398     0.2609  76.038   <2e-16 ***
## zn            0.1803     0.2577   0.700    0.485
## indus        -0.1876     0.2524  -0.743    0.458
## chas         -0.3642     0.2620  -1.390    0.165
## nox           0.1600     0.2670   0.599    0.549
## age          -0.2397     0.2641  -0.908    0.365
## tax           0.1561     0.2607   0.599    0.550
## black        -0.4203     0.2660  -1.580    0.115
## lstat         0.5194     0.2629   1.975    0.049 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.898 on 347 degrees of freedom
## Multiple R-squared:  0.03603,    Adjusted R-squared:  0.01381
## F-statistic: 1.621 on 8 and 347 DF,  p-value: 0.1173
```

Improving lm Model performance by Removing less significant independent variable (tax):

```
lm_model_tax <- lm(medv ~zn+indus+chas+nox+age+black+lstat,
                   data = scaled_train_data)
summary(lm_model_tax)
```

```
##
## Call:
## lm(formula = medv ~ zn + indus + chas + nox + age + black + lstat,
##     data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1920  -3.1334  -0.2922   3.4774  13.1940
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8362     0.2606  76.115   <2e-16 ***
## zn            0.1851     0.2573   0.719   0.4724
## indus        -0.1852     0.2521  -0.734   0.4631
## chas         -0.3651     0.2617  -1.395   0.1639
## nox           0.1629     0.2667   0.611   0.5418
## age          -0.2439     0.2638  -0.925   0.3557
```

```
## black        -0.4256     0.2656  -1.602   0.1100
## lstat         0.5170     0.2627   1.968   0.0498 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.893 on 348 degrees of freedom
## Multiple R-squared:  0.03504,    Adjusted R-squared:  0.01563
## F-statistic: 1.805 on 7 and 348 DF,  p-value: 0.08525
```

Improving lm Model performance by Removing less significant independent variable (nox):

```
lm_model_nox <- lm(medv ~zn+indus+chas+age+black+lstat,
                 data = scaled_train_data)
summary(lm_model_nox)
```

```
##
## Call:
## lm(formula = medv ~ zn + indus + chas + age + black + lstat,
##     data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.1062  -3.1363  -0.2959   3.4651  13.4751
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8311     0.2602  76.202   <2e-16 ***
## zn            0.1786     0.2569   0.695   0.4873
## indus        -0.1778     0.2516  -0.707   0.4802
## chas         -0.3640     0.2615  -1.392   0.1647
## age          -0.2544     0.2630  -0.968   0.3339
## black        -0.4361     0.2648  -1.647   0.1005
## lstat         0.5221     0.2623   1.991   0.0473 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.889 on 349 degrees of freedom
## Multiple R-squared:  0.034,  Adjusted R-squared:  0.0174
## F-statistic: 2.048 on 6 and 349 DF,  p-value: 0.05886
```

Improving lm Model performance by Removing less significant independent variable (zn):

```
lm_model_zn <- lm(medv ~indus+chas+age+black+lstat,
                 data = scaled_train_data)
summary(lm_model_zn)
```

```
##
## Call:
## lm(formula = medv ~ indus + chas + age + black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0283  -3.1705  -0.3746   3.5203  13.6682
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8285     0.2600  76.257   <2e-16 ***
## indus        -0.1766     0.2514  -0.702   0.4829
## chas         -0.3738     0.2609  -1.433   0.1528
## age          -0.2314     0.2607  -0.888   0.3753
## black        -0.4375     0.2646  -1.653   0.0992 .
## lstat         0.5302     0.2618   2.025   0.0436 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.885 on 350 degrees of freedom
## Multiple R-squared:  0.03267,    Adjusted R-squared:  0.01885
## F-statistic: 2.364 on 5 and 350 DF,  p-value: 0.03955
```

Improving lm Model performance by Removing less significant independent variable (indus):

```
lm_model_in <- lm(medv ~ chas+age+black+lstat,
                  data = scaled_train_data)
summary(lm_model_in)
```

```
##
## Call:
## lm(formula = medv ~ chas + age + black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.0589  -3.2309  -0.4212   3.5292  14.0004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8220     0.2597  76.334   <2e-16 ***
## chas         -0.3824     0.2604  -1.468   0.1429
## age          -0.2349     0.2604  -0.902   0.3677
## black        -0.4455     0.2642  -1.686   0.0927 .
## lstat         0.5370     0.2615   2.054   0.0408 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.882 on 351 degrees of freedom
## Multiple R-squared:  0.0313, Adjusted R-squared:  0.02026
## F-statistic: 2.836 on 4 and 351 DF,  p-value: 0.02446
```

Improving lm Model performance by Removing less significant independent variable (age):

```
lm_model_ag <- lm(medv ~ chas+black+lstat,
                  data = scaled_train_data)
summary(lm_model_ag)
```

```
##
## Call:
## lm(formula = medv ~ chas + black + lstat, data = scaled_train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.955  -3.135  -0.328   3.348  14.138
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8195     0.2596  76.349   <2e-16 ***
## chas         -0.3829     0.2604  -1.471   0.1423
## black        -0.4695     0.2628  -1.787   0.0748 .
## lstat         0.5209     0.2608   1.997   0.0465 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.881 on 352 degrees of freedom
## Multiple R-squared:  0.02906,    Adjusted R-squared:  0.02078
## F-statistic: 3.511 on 3 and 352 DF,  p-value: 0.01549
```

Improving lm Model performance by Removing less significant independent variable (chas):

```
lm_model_chas <- lm(medv ~ black+lstat,
                  data = scaled_train_data)
summary(lm_model_chas)
```

```
##
## Call:
## lm(formula = medv ~ black + lstat, data = scaled_train_data)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -13.3957  -3.0712  -0.3097   3.3733  14.5122
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8365     0.2598  76.366   <2e-16 ***
## black        -0.5012     0.2623  -1.911   0.0569 .
## lstat         0.5282     0.2612   2.022   0.0439 *
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.889 on 353 degrees of freedom
## Multiple R-squared:  0.02309,    Adjusted R-squared:  0.01756
## F-statistic: 4.172 on 2 and 353 DF,  p-value: 0.01619
```

In this point we observe after removing independent variable "chas" we found our Adjusted R- squeared value go down than keeping this "chas" also lstat is the best infulancing variable. So we are selecting independent variables for lm model "chas", "black", and "lstat". Furthermore, We will check other models with those three independent variables.

Final Linear Regression Model with selected IV:

```
lm_model <- lm(medv ~ chas+black+lstat,
                data = scaled_train_data)
summary(lm_model)
```

```
##
## Call:
## lm(formula = medv ~ chas + black + lstat, data = scaled_train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -12.955  -3.135  -0.328   3.348  14.138
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  19.8195     0.2596  76.349   <2e-16 ***
## chas         -0.3829     0.2604  -1.471   0.1423
## black        -0.4695     0.2628  -1.787   0.0748 .
## lstat         0.5209     0.2608   1.997   0.0465 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.881 on 352 degrees of freedom
## Multiple R-squared:  0.02906,    Adjusted R-squared:  0.02078
## F-statistic: 3.511 on 3 and 352 DF,  p-value: 0.01549
```

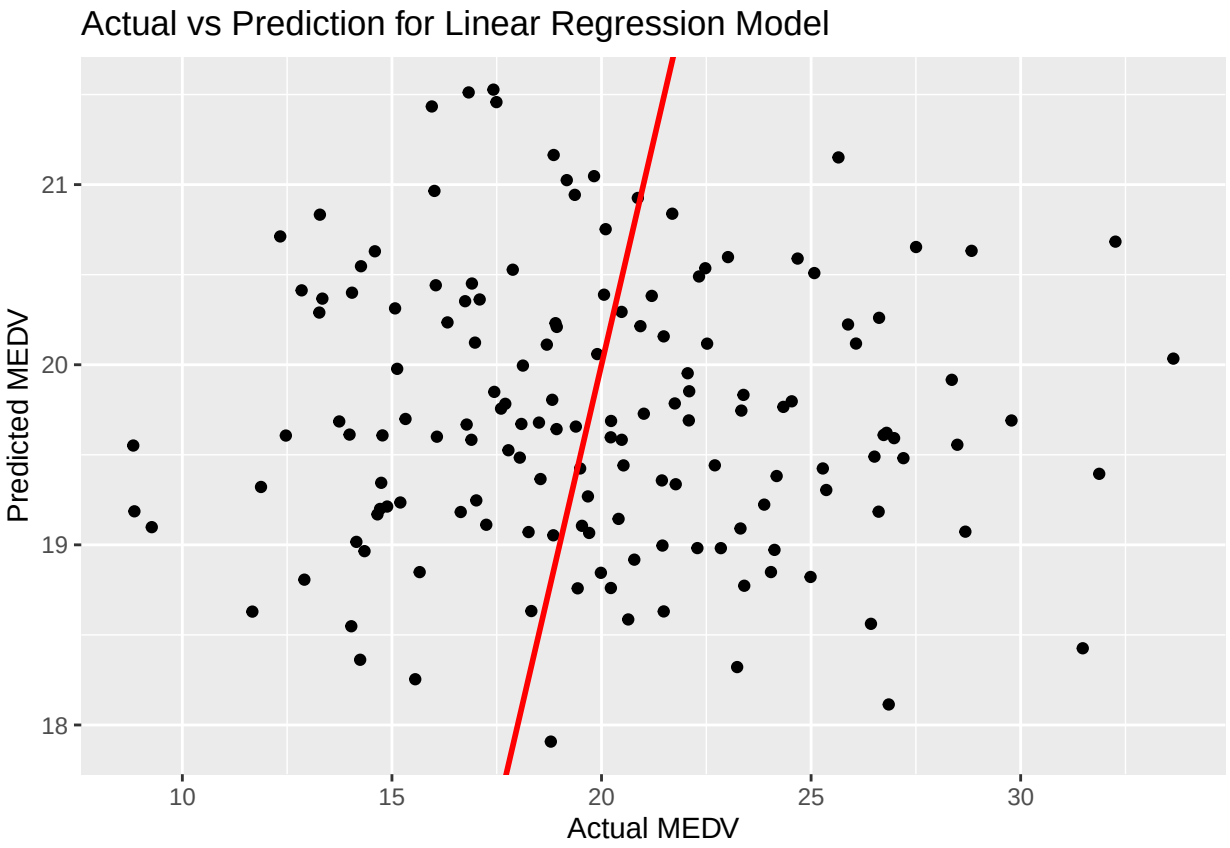
```
lm_pred <- predict(lm_model, newdata = scaled_test_data)
lm_mse <- mean((lm_pred - scaled_test_data$medv)^2)
lm_rsquared <- R2(lm_pred, scaled_test_data$medv)
```

Visualization of LM model

```
df<-data.frame(Actual=scaled_test_data$medv,
               Predicted = lm_pred,
               LSTAT=scaled_test_data$lstat)
```

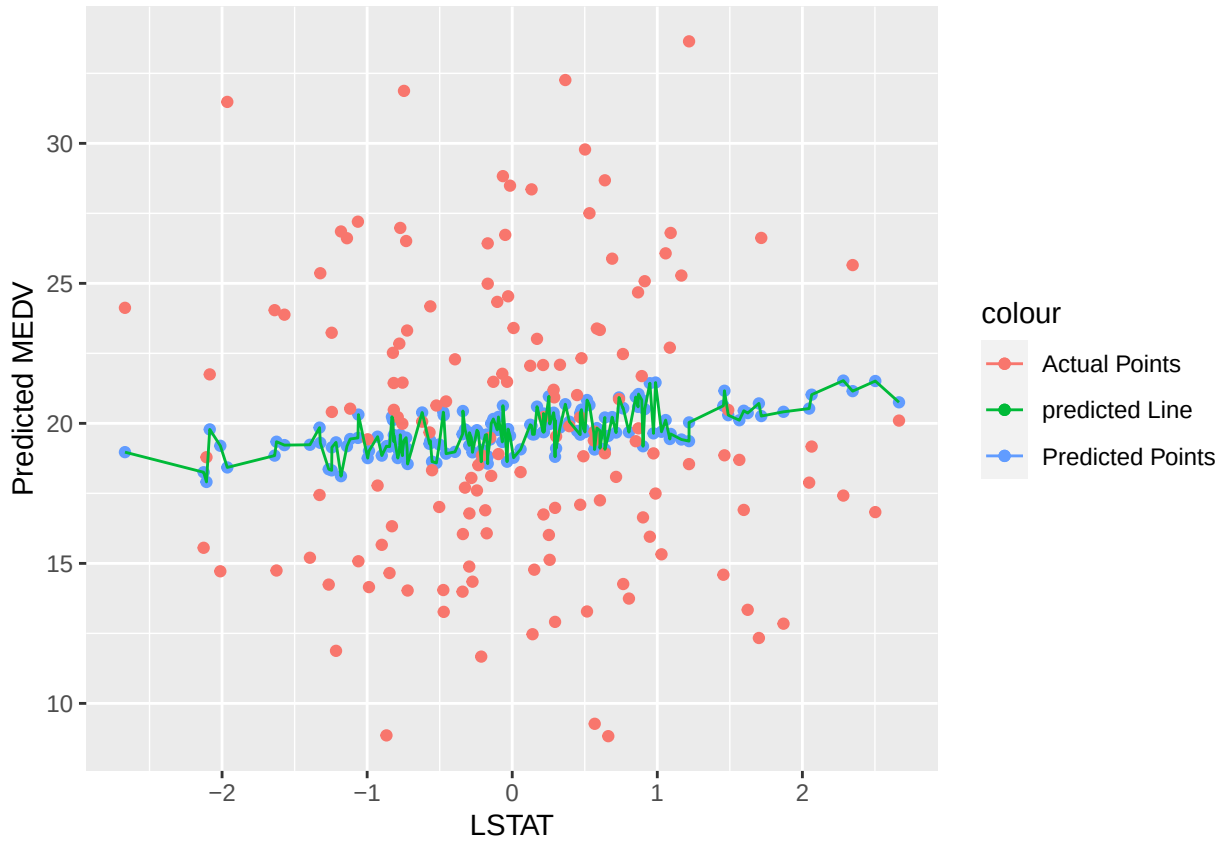
```
ggplot(data=df,aes(x=Actual,y=Predicted))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1,linetype=1,color="red",size=1)+
  labs(x="Actual MEDV", y= "Predicted MEDV")+
  ggtitle("Actual vs Prediction for Linear Regression Model")
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
```



Visualization with most correlated independent variable

```
ggplot()+
  geom_point(data=df,
             aes(x=LSTAT,y=Predicted, color="Predicted Points"))+
  geom_point(data=df,
             aes(x=LSTAT,y=Actual,color="Actual Points"))+
  geom_line(data=df,
            aes(x=LSTAT,y=Predicted,color="predicted Line"))+
  labs(y="Predicted MEDV", x= "LSTAT")
```



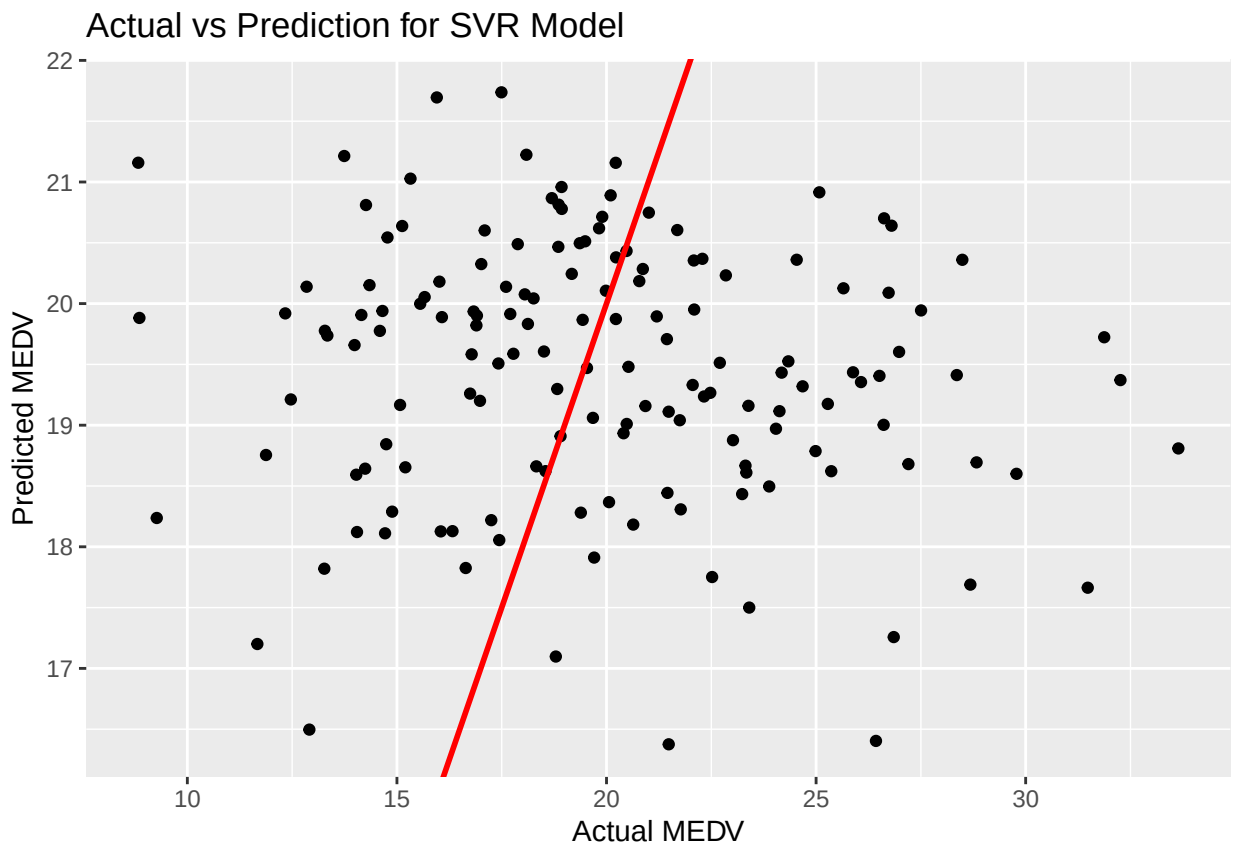
Support Vector Regression Model with selected IV:

```
svr_model <- svm(medv ~ chas+black+lstat, data = scaled_train_data)
svr_pred <- predict(svr_model, newdata = scaled_test_data)
svr_mse <- mean((svr_pred - scaled_test_data$medv)^2)
svr_rsquared <- R2(svr_pred, scaled_test_data$medv)
```

Visualizing the SVR Model

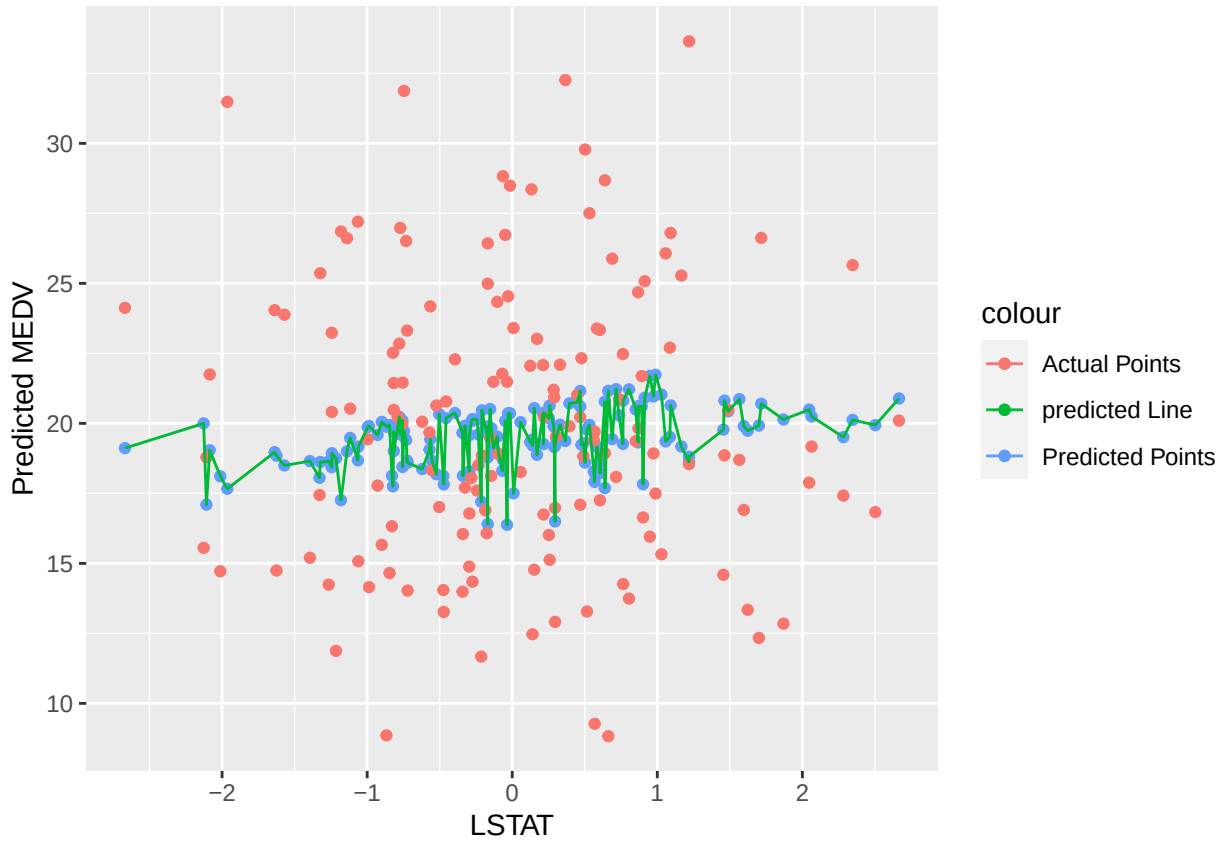
```
dfsvr<-data.frame(Actual=scaled_test_data$medv,
                  Predicted = svr_pred,
                  LSTAT=scaled_test_data$lstat)

ggplot(data=dfsvr,aes(x=Actual,y=Predicted))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1,linetype=1,color="red",size=1)+
  labs(x="Actual MEDV", y= "Predicted MEDV")+
  ggtitle("Actual vs Prediction for SVR Model")
```



Visualization with most correlated independent variable:

```
ggplot()+
  geom_point(data=dfsvr,
             aes(x=LSTAT,y=Predicted, color="Predicted Points"))+
  geom_point(data=dfsvr,
             aes(x=LSTAT,y=Actual,color="Actual Points"))+
  geom_line(data=dfsvr,
            aes(x=LSTAT,y=Predicted,color="predicted Line"))+
  labs(y="Predicted MEDV", x= "LSTAT")
```



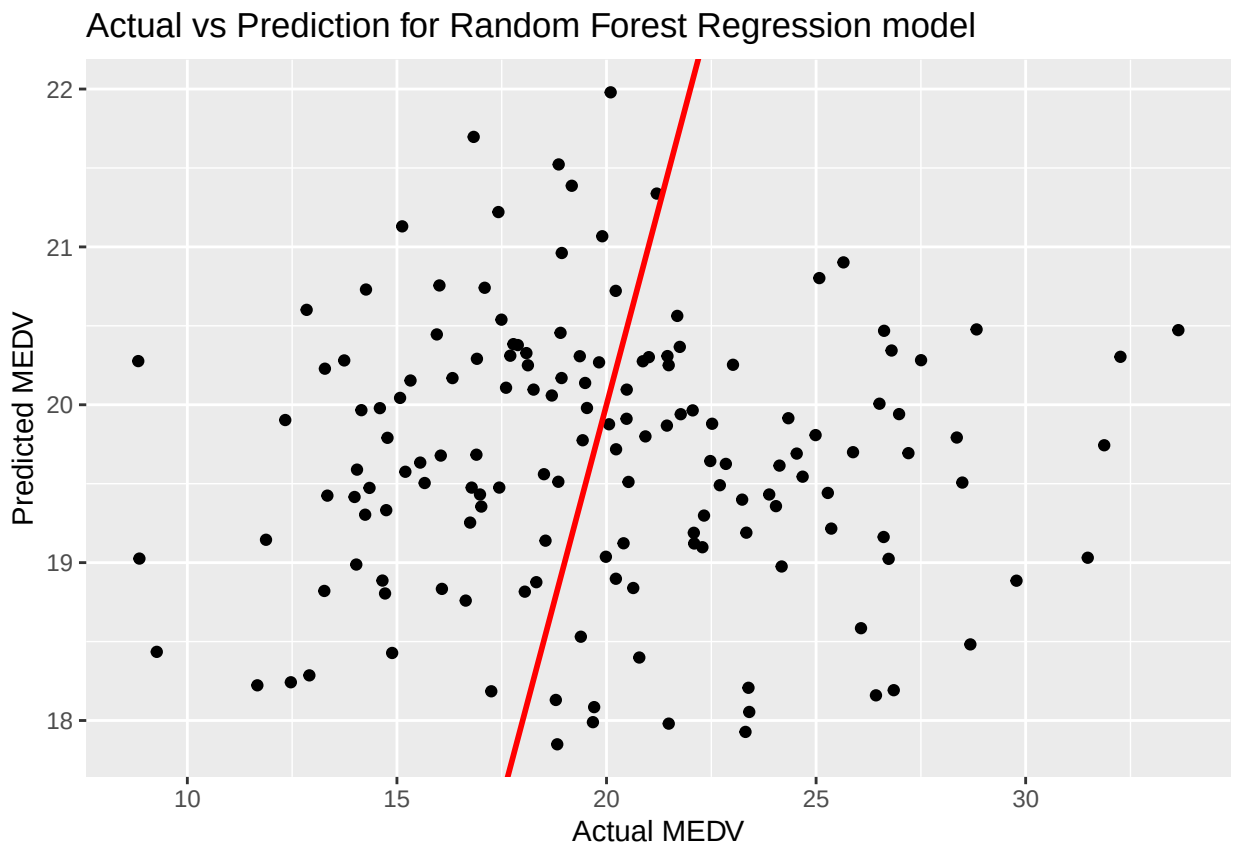
Random forest Regression Regression Model with selected IV:

```
rf_model <- randomForest(medv ~ chas+black+lstat, data = scaled_train_data,ntree=500)
rf_pred <- predict(rf_model, newdata = scaled_test_data)
rf_mse <- mean((rf_pred - scaled_test_data$medv)^2)
rf_rsquared <- cor(rf_pred, scaled_test_data$medv)^2
```

Visualization of Random Forest regression model

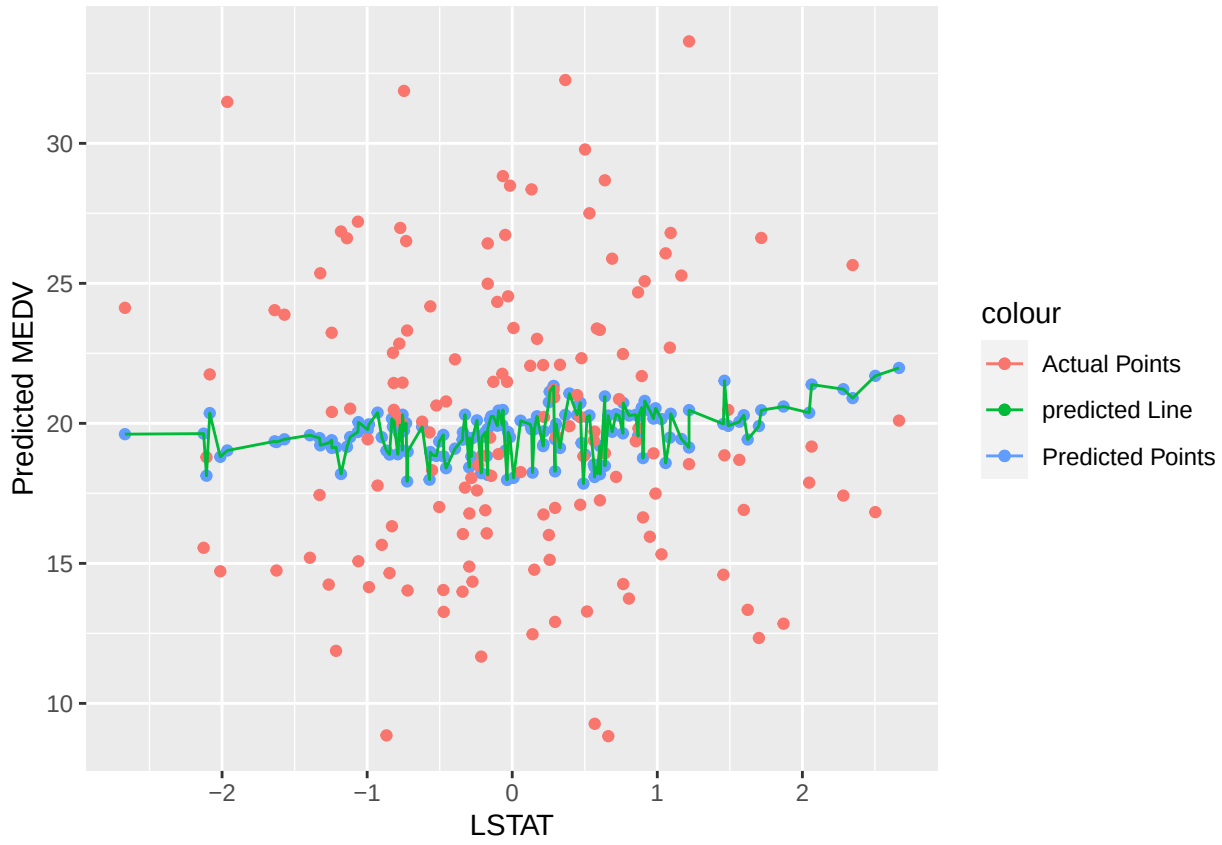
```
dfrf<-data.frame(Actual=scaled_test_data$medv,
                 Predicted = rf_pred,
                 LSTAT=scaled_test_data$lstat)

ggplot(data=dfrf,aes(x=Actual,y=Predicted))+
  geom_point()+
  geom_abline(intercept = 0,slope = 1,linetype=1,color="red",size=1)+
  labs(x="Actual MEDV", y= "Predicted MEDV")+
  ggtitle("Actual vs Prediction for Random Forest Regression model")
```



Visualization with most correlated independent variable

```
ggplot()+
  geom_point(data=dfrf,
             aes(x=LSTAT,y=Predicted, color="Predicted Points"))+
  geom_point(data=dfrf,
             aes(x=LSTAT,y=Actual,color="Actual Points"))+
  geom_line(data=dfrf,
            aes(x=LSTAT,y=Predicted,color="predicted Line"))+
  labs(y="Predicted MEDV", x= "LSTAT")
```



Create data frame to compare R-squared values

```
rsquared_df <- data.frame(Model = c("Linear Regression", "Support Vector Regression",
                                    "Random Forest Regression"),
                          R_Squared = c(lm_rsquared, svr_rsquared, rf_rsquared))

rsquared_df %>% arrange(desc(R_Squared))
```

```
##                         Model    R_Squared
## 1 Support Vector Regression 1.375404e-02
## 2         Linear Regression 8.374426e-04
## 3  Random Forest Regression 3.296482e-05
```

```
print(rsquared_df)
```

```
##                         Model    R_Squared
## 1         Linear Regression 8.374426e-04
## 2 Support Vector Regression 1.375404e-02
## 3  Random Forest Regression 3.296482e-05
```

Create data frame to compare MSE values

```
mse_df <- data.frame(Model = c("Linear Regression","Support Vector Regression",
                               "Random Forest Regression"),
                     mse = c(lm_mse,svr_mse,rf_mse))

mse_df %>% arrange(desc(mse))
```

```
##                         Model      mse
## 1 Support Vector Regression 26.72845
## 2         Linear Regression 24.95647
## 3  Random Forest Regression 24.87048
```

```
print(mse_df)
```

```
##                         Model      mse
## 1         Linear Regression 24.95647
## 2 Support Vector Regression 26.72845
## 3  Random Forest Regression 24.87048
```

Summary MSE and R-Squared value

```
Comparison_df<- cbind(rsquared_df,MSE=mse_df$mse)
print(Comparison_df)
```

```
##                         Model    R_Squared      MSE
## 1         Linear Regression 8.374426e-04 24.95647
## 2 Support Vector Regression 1.375404e-02 26.72845
## 3  Random Forest Regression 3.296482e-05 24.87048
```

The findings consist of the evaluation metrics (R-squared and Mean squared error) for three different regression models. The best-performing models in terms of R-squared is Support Vector Regression with high R-Squared value. In terms of mean squared error, Random Forest Regression model is the best-performing model for lowest MSE value. But their suitability depends on the specific problem being addressed where R-Squared value is not even more than 50% for any model. So not any model can perfectly predict target variable (Housing prices of Boston).

Overall, the choice of model will depend on multiple factors, and should not be based solely on these performance metrics.