Boston Housing Price Prediction Using R

The dataset for this project originates from the UCI Machine Learning Repository. The Boston housing data was collected in 1978 and each of the 506 entries represent aggregated data about 14 features for homes from various suburbs in Boston, Massachusetts.

**Acknowledgements**

<https://github.com/udacity/machine-learning>

<https://archive.ics.uci.edu/ml/datasets/Housing>

#Clean the global environment space  
rm(list = ls())  
  
#Import dataset  
df\_house = read.csv("boston\_housing.csv", sep = ";")  
  
#Find the first five observations in the dataset  
head(df\_house)

## RM LSTAT PTRATIO MEDV  
## 1 6.575 4.98 15.3 504000  
## 2 6.421 9.14 17.8 453600  
## 3 7.185 4.03 17.8 728700  
## 4 6.998 2.94 18.7 701400  
## 5 7.147 5.33 18.7 760200  
## 6 6.430 5.21 18.7 602700

#Find the dimension of the dataset  
dim(df\_house)

## [1] 489 4

#Find the column names of the dataset  
colnames(df\_house)

## [1] "RM" "LSTAT" "PTRATIO" "MEDV"

#Get the structure of the dataset  
str(df\_house)

## 'data.frame': 489 obs. of 4 variables:  
## $ RM : num 6.58 6.42 7.18 7 7.15 ...  
## $ LSTAT : num 4.98 9.14 4.03 2.94 5.33 ...  
## $ PTRATIO: num 15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...  
## $ MEDV : num 504000 453600 728700 701400 760200 ...  
  
#Find the summary of the dataset  
summary(df\_house)

## RM LSTAT PTRATIO MEDV   
## Min. :3.561 Min. : 1.98 Min. :12.60 Min. : 105000   
## 1st Qu.:5.880 1st Qu.: 7.37 1st Qu.:17.40 1st Qu.: 350700   
## Median :6.185 Median :11.69 Median :19.10 Median : 438900   
## Mean :6.240 Mean :12.94 Mean :18.52 Mean : 454343   
## 3rd Qu.:6.575 3rd Qu.:17.12 3rd Qu.:20.20 3rd Qu.: 518700   
## Max. :8.398 Max. :37.97 Max. :22.00 Max. :1024800  
  
#Check for missing values in the dataset  
colSums(is.na(df\_house))

## RM LSTAT PTRATIO MEDV   
## 0 0 0 0  
  
#Data Analysis  
#install.packages("dplyr")  
library(dplyr)

##   
## Attaching package: 'dplyr'

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

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

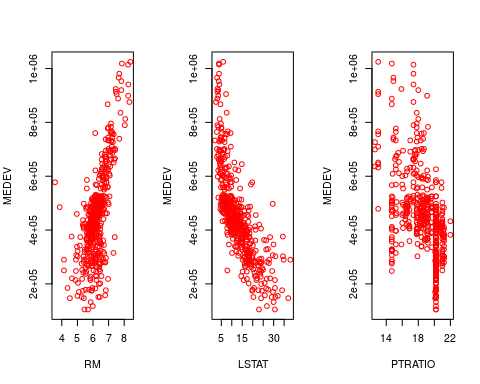
#Get the total sum for each RM  
rm\_total = summarise(group\_by(df\_house, RM), total\_sum = sum(MEDV))

## `summarise()` ungrouping output (override with `.groups` argument)

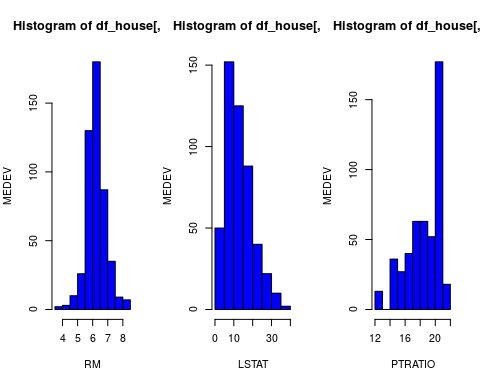
arrange(rm\_total)

## # A tibble: 430 x 2  
## RM total\_sum  
## <dbl> <dbl>  
## 1 3.56 577500  
## 2 3.86 485100  
## 3 4.14 539700  
## 4 4.37 184800  
## 5 4.52 147000  
## 6 4.63 375900  
## 7 4.65 220500  
## 8 4.88 214200  
## 9 4.90 247800  
## 10 4.91 289800  
## # … with 420 more rows

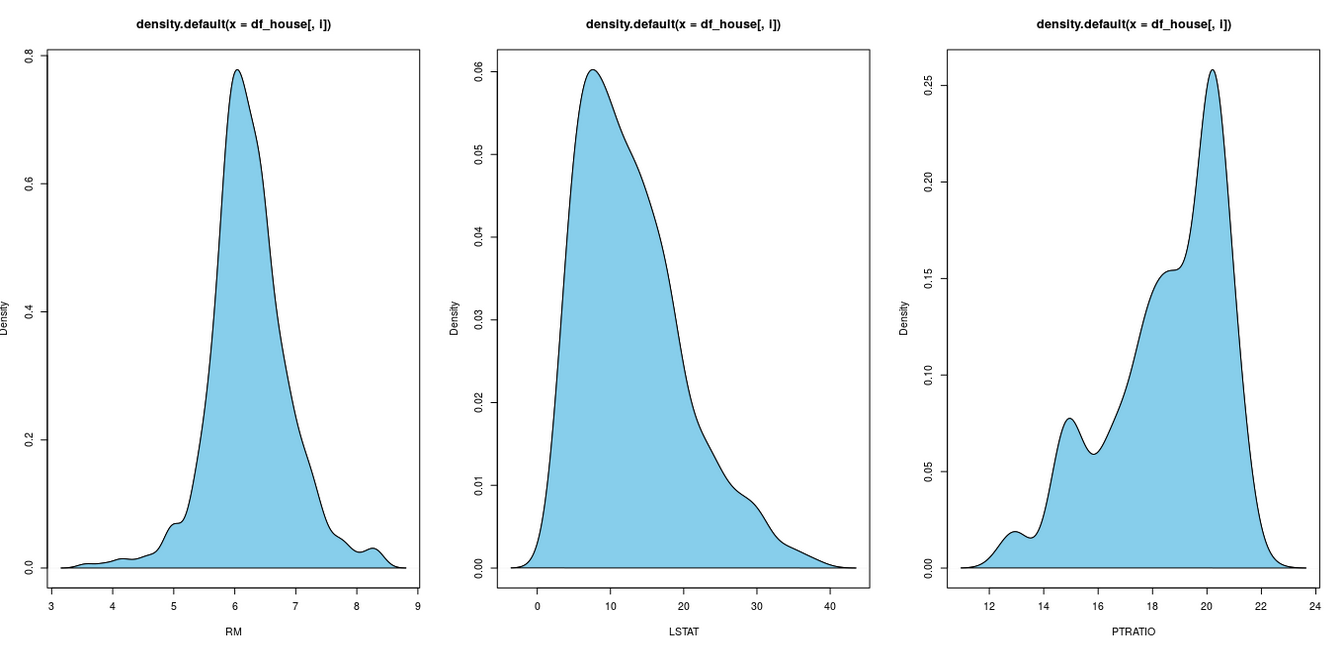
#Checking the relationship of the independent variables with the dependent using scatter plot  
par(mfrow = c(1,3))  
  
for (i in c("RM","LSTAT","PTRATIO")) {  
 plot(df\_house[,i], df\_house$MEDV,  
 xlab = i,  
 ylab = "MEDEV",  
 col = "red")  
}



#Plot a histogram of the dataset  
par(mfrow = c(1,3))  
  
for (i in c("RM","LSTAT","PTRATIO")) {  
 hist(df\_house[,i],  
 xlab = i,  
 ylab = "MEDEV",  
 col = "blue")  
}



#Plot a density plot  
par(mfrow = c(1,3))  
  
for (i in c("RM","LSTAT","PTRATIO")) {  
 density\_data = density(df\_house[,i])  
 plot(density\_data)  
 polygon(density\_data ,col=" skyblue ", border="black")  
}



#Correlation between the variables in the motor insurance  
cor(df\_house)

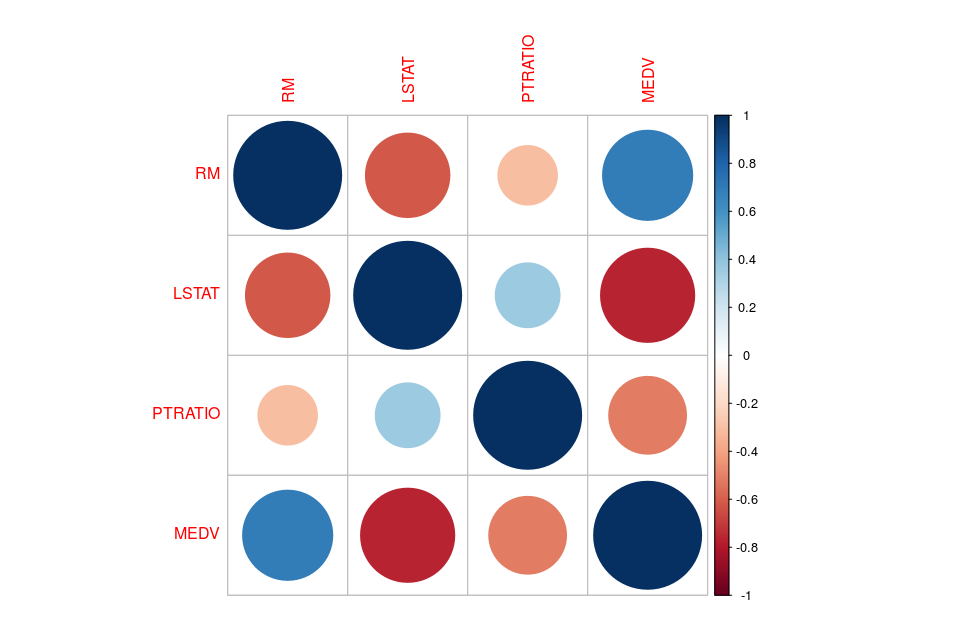
## RM LSTAT PTRATIO MEDV  
## RM 1.0000000 -0.6120332 -0.3045593 0.6972092  
## LSTAT -0.6120332 1.0000000 0.3604446 -0.7606701  
## PTRATIO -0.3045593 0.3604446 1.0000000 -0.5190335  
## MEDV 0.6972092 -0.7606701 -0.5190335 1.0000000  
  
#A for loop to check the independent variables against the dependent variable  
for (i in c("RM","LSTAT","PTRATIO")) {  
 df\_cor = cor(df\_house[,i], df\_house$MEDV)  
 print(df\_cor)  
}

## [1] 0.6972092  
## [1] -0.7606701  
## [1] -0.5190335

#Plotting the correlation between variables  
#install.packages('corrplot', dependencies = T)  
library(corrplot)

## corrplot 0.84 loaded

corrplot(cor(df\_house))

  
  
#Splitting the dataset into Training and Test set.  
#install.packages("caTools")  
library(caTools)  
set.seed(123)  
split\_dataset = sample.split(df\_house$MEDV, SplitRatio = 2/3)  
training\_set = subset(df\_house, split\_dataset == TRUE)  
test\_set = subset(df\_house, split\_dataset == FALSE)  
  
#Simple Linear Regression  
  
#Define Hypothesis  
  
# Ho: That RM has no an impact MEDV  
# H1: That RM has an impact on MEDV  
  
regressor = lm(MEDV ~ RM, data = training\_set) # Regression of payment ~ claims  
summary(regressor)

##   
## Call:  
## lm(formula = MEDV ~ RM, data = training\_set)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -406773 -47479 6614 67522 625778   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) -725489 60530 -11.99 <2e-16 \*\*\*  
## RM 190174 9600 19.81 <2e-16 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 121300 on 340 degrees of freedom  
## Multiple R-squared: 0.5358, Adjusted R-squared: 0.5344   
## F-statistic: 392.4 on 1 and 340 DF, p-value: < 2.2e-16

alpha = 0.05  
pvalue = 2e-16  
  
pvalue < alpha

## [1] TRUE

#Results: We reject the null hypotheses as RM is highly significant and has a relationship with MEDEV

#Predicting the Test set results  
y\_pred = predict(regressor, newdata = test\_set)  
y\_pred[0:10]

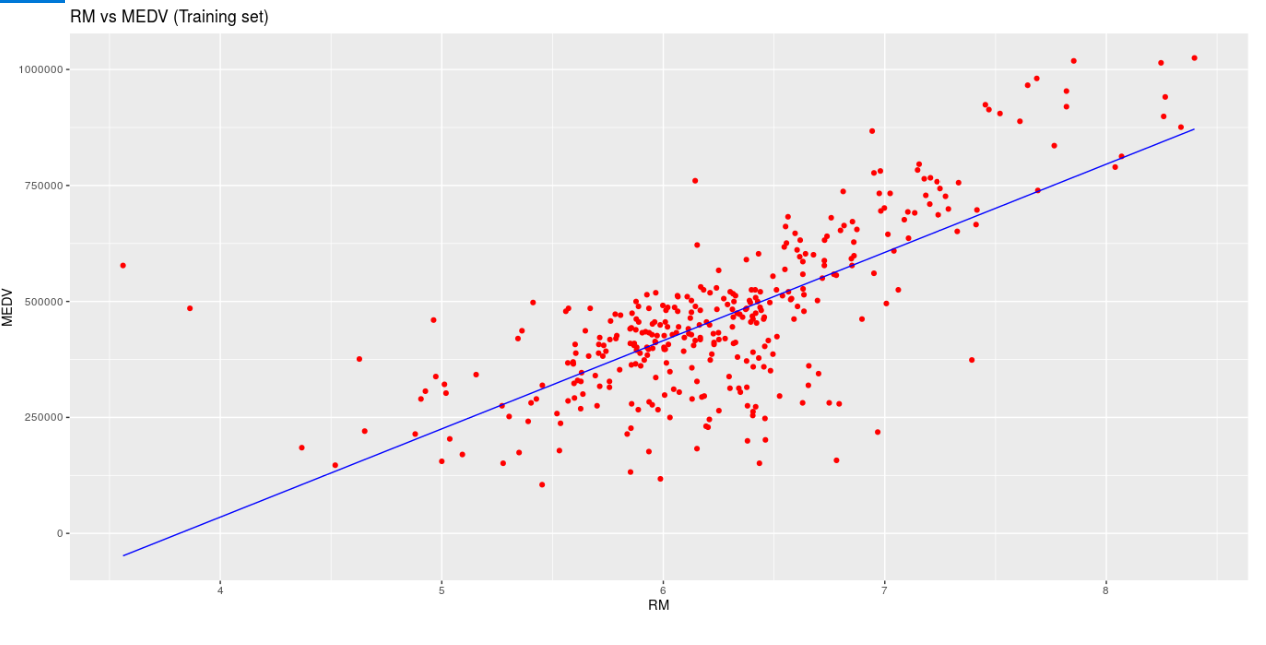
## 5 8 15 16 18 19 23 24   
## 633686.3 448266.5 433813.3 383987.6 413654.8 312101.8 442561.3 379993.9   
## 25 27   
## 401103.3 379993.9

#Evaluating the accuracy of our Simple Linear Regression Model  
Rsqd = summary(regressor)$r.squared  
Rsqd

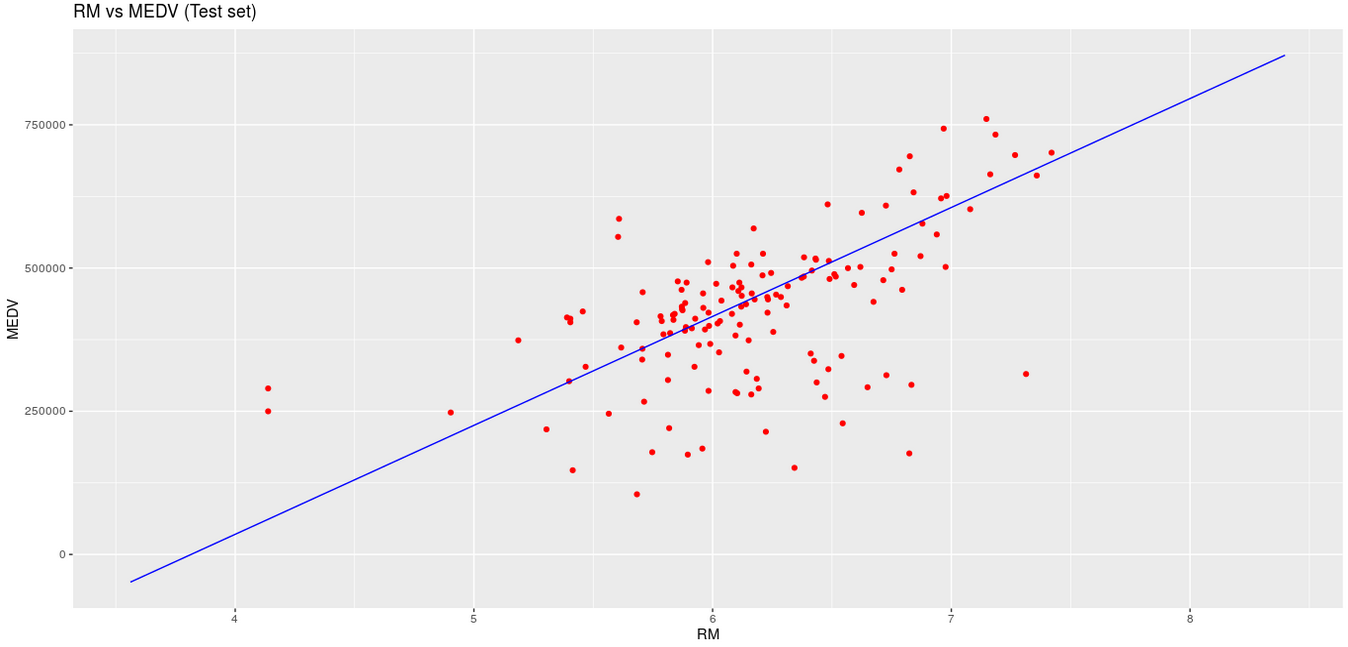
## [1] 0.5357829

#Visualising the Simple Linear Regression using Training set results  
#install.packages('ggplot2')

library(ggplot2)  
ggplot() +   
 geom\_point(aes(x = training\_set$RM, y = training\_set$MEDV), # Graph for training set  
 color = 'red') +   
 geom\_line(aes(x = training\_set$RM, y = predict(regressor, newdata = training\_set)),  
 color = 'blue') +  
 ggtitle('RM vs MEDV (Training set)') +  
 xlab('RM') +  
 ylab('MEDV')



#Visualizing the Simple Linear Regression using Test set results  
ggplot() +   
 geom\_point(aes(x = test\_set$RM, y = test\_set$MEDV), # Graph for test set  
 color = 'red') +   
 geom\_line(aes(x = training\_set$RM, y = predict(regressor, newdata = training\_set)),  
 color = 'blue') +  
 ggtitle('RM vs MEDV (Test set)') +  
 xlab('RM') +  
 ylab('MEDV')



#Multiple Linear Regression  
  
#Creating the multiple linear regression model  
regressor1 = lm(MEDV ~ ., data = training\_set)  
summary(regressor1)

##   
## Call:  
## lm(formula = MEDV ~ ., data = training\_set)  
##   
## Residuals:  
## Min 1Q Median 3Q Max   
## -234791 -57021 -11176 41261 344376   
##   
## Coefficients:  
## Estimate Std. Error t value Pr(>|t|)   
## (Intercept) 328192.3 82768.8 3.965 8.95e-05 \*\*\*  
## RM 97341.3 9181.5 10.602 < 2e-16 \*\*\*  
## LSTAT -11058.0 870.4 -12.704 < 2e-16 \*\*\*  
## PTRATIO -17968.7 2466.2 -7.286 2.27e-12 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## Residual standard error: 90590 on 338 degrees of freedom  
## Multiple R-squared: 0.7426, Adjusted R-squared: 0.7403   
## F-statistic: 325.1 on 3 and 338 DF, p-value: < 2.2e-16

#Predicting the Test set results  
y\_predict = predict(regressor1, newdata = test\_set)  
y\_predict[0:10]

## 5 8 15 16 18 19 23 24   
## 628936.8 444098.1 430787.3 425077.7 371703.4 352676.0 341714.4 296861.9   
## 25 27   
## 347254.3 352925.9

Rsq1 = summary(regressor1)$r.squared  
Rsq1 # 0.7426239 - 74%

## [1] 0.7426239