BOSTON HOUSING MARKET PREDICTION

steve dubois

12/10/2018

Overview

In this report I detail the machine learning (ML) models I implemented to accurately predict the housing prices in Boston suburbs. The data set for this experiment is accessed from the UCI Machine Learning repository via https://archive.ics.uci.edu/ml/datasets/Housing. The report is organized in such a way as to demonstrate the entire process right from getting and cleaning the data, to exploratory analysis of the data set to understand the distribution and importance of various features in influencing the algorithm, to coming with a hypothesis, training ML models, evaluation of the models, etc.

Introduction

title: "Boston Housing Price Prediction" The data set (Boston Housing Price) was taken from the StatLib library which is maintained at Carnegie Mellon University and is freely available for download from the UCI Machine Learning Repository. The data set consists of 506 observations of 14 attributes. The median value of house price in $10000s, denoted by MEDV, is the outcome or the dependent variable in our model. Below is a brief description of each feature and the outcome in our data set: Variables:

1. CRIM – per capita crime rate by town
2. ZN – proportion of residential land zoned for lots over 25,000 sq.ft
3. CHAS – Charles River dummy variable (1 if tract bounds river; else 0)
4. NOX – nitric oxides concentration (parts per 10 million)
5. RM – average number of rooms per dwelling
6. AGE – proportion of owner-occupied units built prior to 1940
7. DIS – weighted distances to five Boston employment centers
8. RAD – index of accessibility to radial highways
9. INDUS – proportion of non-retail business acres per town
10. TAX – full-value property-tax rate per $10,000
11. PTRATIO – pupil-teacher ratio by town
12. B – 1000(Bk - 0.63)^2 where Bk is the proportion of blacks by town
13. LSTAT – % lower status of the population
14. MEDV – Median value of owner-occupied homes in $10000's (response variable)

Getting and Cleaning the Data

Getting the data into R as an R object, cleaning the data and transforming it as a neat and usable R data frame or equivalent. The df (Boston housing) file consists of the actual data, The R-function readLiness() reads data from fixed-width files. Below, I use subsetting and the strsplit R function to extract the columns/predictors and column names alone from this file.

```r
# DATA DOWNLOWNED
text <- readLines("boston.txt")
text <- text[c(-1, -2)]
```

PREPROCESSING/TRANSFORM DATA

```r
i=1
df2 <- NULL
while (i <= 1012) {
    if (i%%2 == 0) {
i=i+1 } else i
j <- i + 1
texti <- as.numeric(strsplit(text, " ")[[i]])
texti <- na.omit(texti)
textj <- as.numeric(strsplit(text, " ")[[j]])
textj <- na.omit(textj)
textC <- as.vector(c(texti, textj))
df <- NULL
df <- rbind(df2, textC)
colnames(df) <- c("CRIM", "ZN", "INDUS", "CHAS", "NOX", "RM", "AGE", "DIS",
"RAD", "TAX", "PTRATIO", "B", "LSTAT", "MEDV")
rownames(df) <- c()
df2 <- df
i <- j + 1
df <- as.data.frame(df) }
# first 6 observations
head(df)
```

```
##      CRIM ZN INDUS CHAS   NOX    RM  AGE    DIS RAD TAX PTRATIO      B
## 1 0.00632 18  2.31    0 0.538 6.575 65.2 4.0900   1 296    15.3 396.90
## 2 0.02731  0  7.07    0 0.469 6.421 78.9 4.9671   2 242    17.8 396.90
## 3 0.02729  0  7.07    0 0.469 7.185 61.1 4.9671   2 242    17.8 392.83
## 4 0.03237  0  2.18    0 0.458 6.998 45.8 6.0622   3 222    18.7 394.63
## 5 0.06905  0  2.18    0 0.458 7.147 54.2 6.0622   3 222    18.7 396.90
## 6 0.02985  0  2.18    0 0.458 6.430 58.7 6.0622   3 222    18.7 394.12
##   LSTAT MEDV
## 1  4.98 24.0
## 2  9.14 21.6
## 3  4.03 34.7
## 4  2.94 33.4
## 5  5.33 36.2
## 6  5.21 28.7
```

Now, let us check and explore the cleaned data frame containing the housing data. The df R object is of class ‘data.frame’, which is very easy to work with using R scripts. The str() function is powerful in displaying the structure of an R dataframe. Below, the output of str() compactly provides the relevant information of our dataframe, like the number of observations, number of variables, names of each column, the class of each column, and sample values from each column.

```r
# DiSPLAY CLASS OF R OBJECY the class of the R object "df"
class(df)
```

```
## [1] "data.frame"
```

```r
# DiSPLAY SUMMARY STATISTICS
summary(df)
```

```
##       CRIM                ZN             INDUS            CHAS
##  Min.   : 0.00632   Min.   :  0.00   Min.   : 0.46   Min.   :0.00000
##  1st Qu.: 0.08204   1st Qu.:  0.00   1st Qu.: 5.19   1st Qu.:0.00000
```

```
##  Median : 0.25651   Median :  0.00   Median : 9.69   Median :0.00000
##  Mean   : 3.61352   Mean   : 11.36   Mean   :11.14   Mean   :0.06917
##  3rd Qu.: 3.67708   3rd Qu.: 12.50   3rd Qu.:18.10   3rd Qu.:0.00000
##  Max.   :88.97620   Max.   :100.00   Max.   :27.74   Max.   :1.00000
##       NOX               RM             AGE              DIS
##  Min.   :0.3850   Min.   :3.561   Min.   :  2.90   Min.   : 1.130
##  1st Qu.:0.4490   1st Qu.:5.886   1st Qu.: 45.02   1st Qu.: 2.100
##  Median :0.5380   Median :6.208   Median : 77.50   Median : 3.207
##  Mean   :0.5547   Mean   :6.285   Mean   : 68.57   Mean   : 3.795
##  3rd Qu.:0.6240   3rd Qu.:6.623   3rd Qu.: 94.08   3rd Qu.: 5.188
##  Max.   :0.8710   Max.   :8.780   Max.   :100.00   Max.   :12.127
##       RAD              TAX           PTRATIO             B
##  Min.   : 1.000   Min.   :187.0   Min.   :12.60   Min.   :  0.32
##  1st Qu.: 4.000   1st Qu.:279.0   1st Qu.:17.40   1st Qu.:375.38
##  Median : 5.000   Median :330.0   Median :19.05   Median :391.44
##  Mean   : 9.549   Mean   :408.2   Mean   :18.46   Mean   :356.67
##  3rd Qu.:24.000   3rd Qu.:666.0   3rd Qu.:20.20   3rd Qu.:396.23
##  Max.   :24.000   Max.   :711.0   Max.   :22.00   Max.   :396.90
##      LSTAT            MEDV
##  Min.   : 1.73   Min.   : 5.00
##  1st Qu.: 6.95   1st Qu.:17.02
##  Median :11.36   Median :21.20
##  Mean   :12.65   Mean   :22.53
##  3rd Qu.:16.95   3rd Qu.:25.00
##  Max.   :37.97   Max.   :50.00
```

```r
# DISPLAY TRUCTURE OF HOUSING .df data-frame-set
str(df)
```

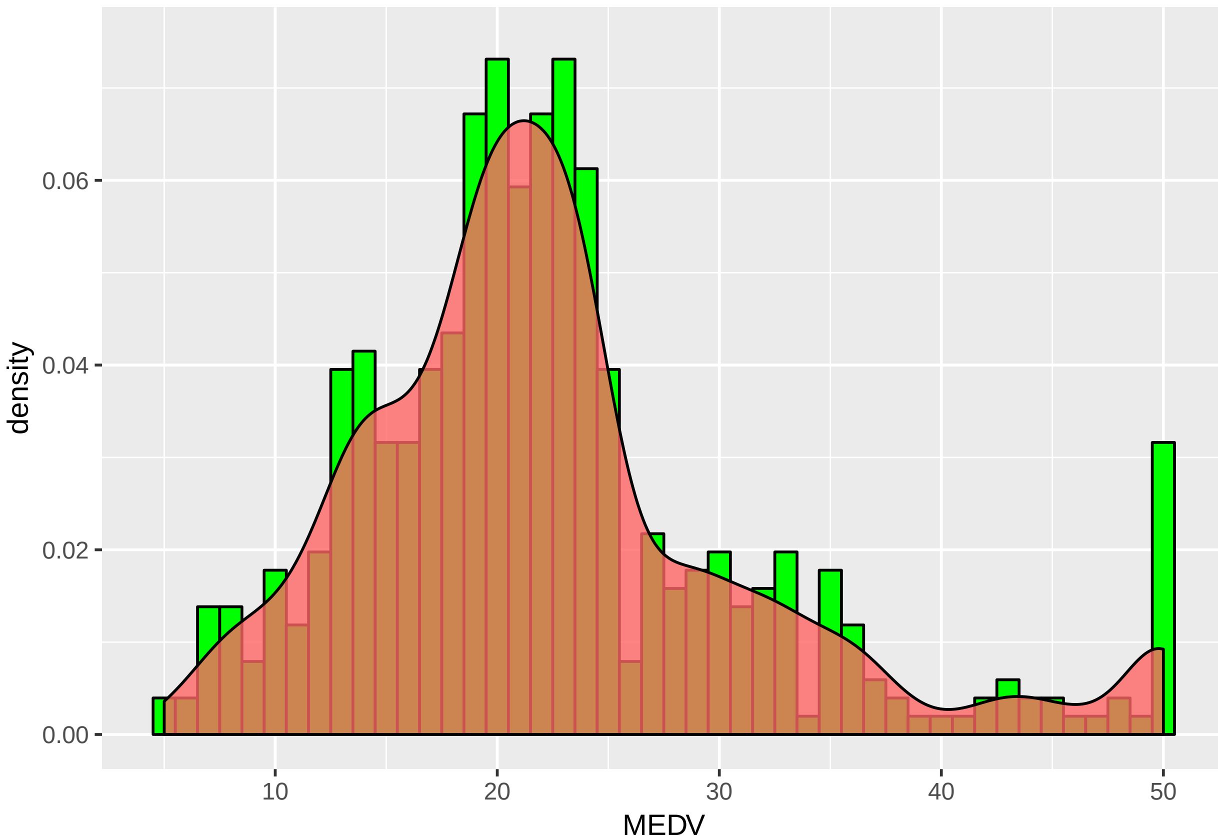
```
## 'data.frame':    506 obs. of  14 variables:
##  $ CRIM   : num  0.00632 0.02731 0.02729 0.03237 0.06905 ...
##  $ ZN     : num  18 0 0 0 0 0 12.5 12.5 12.5 12.5 ...
##  $ INDUS  : num  2.31 7.07 7.07 2.18 2.18 2.18 7.87 7.87 7.87 7.87 ...
##  $ CHAS   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ NOX    : 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    : num  1 2 2 3 3 3 5 5 5 5 ...
##  $ TAX    : num  296 242 242 222 222 222 311 311 311 311 ...
##  $ PTRATIO: num  15.3 17.8 17.8 18.7 18.7 18.7 15.2 15.2 15.2 15.2 ...
##  $ B      : num  397 397 393 395 397 ...
##  $ LSTAT  : num  4.98 9.14 4.03 2.94 5.33 ...
##  $ MEDV   : num  24 21.6 34.7 33.4 36.2 28.7 22.9 27.1 16.5 18.9 ...
```

Data Exploration

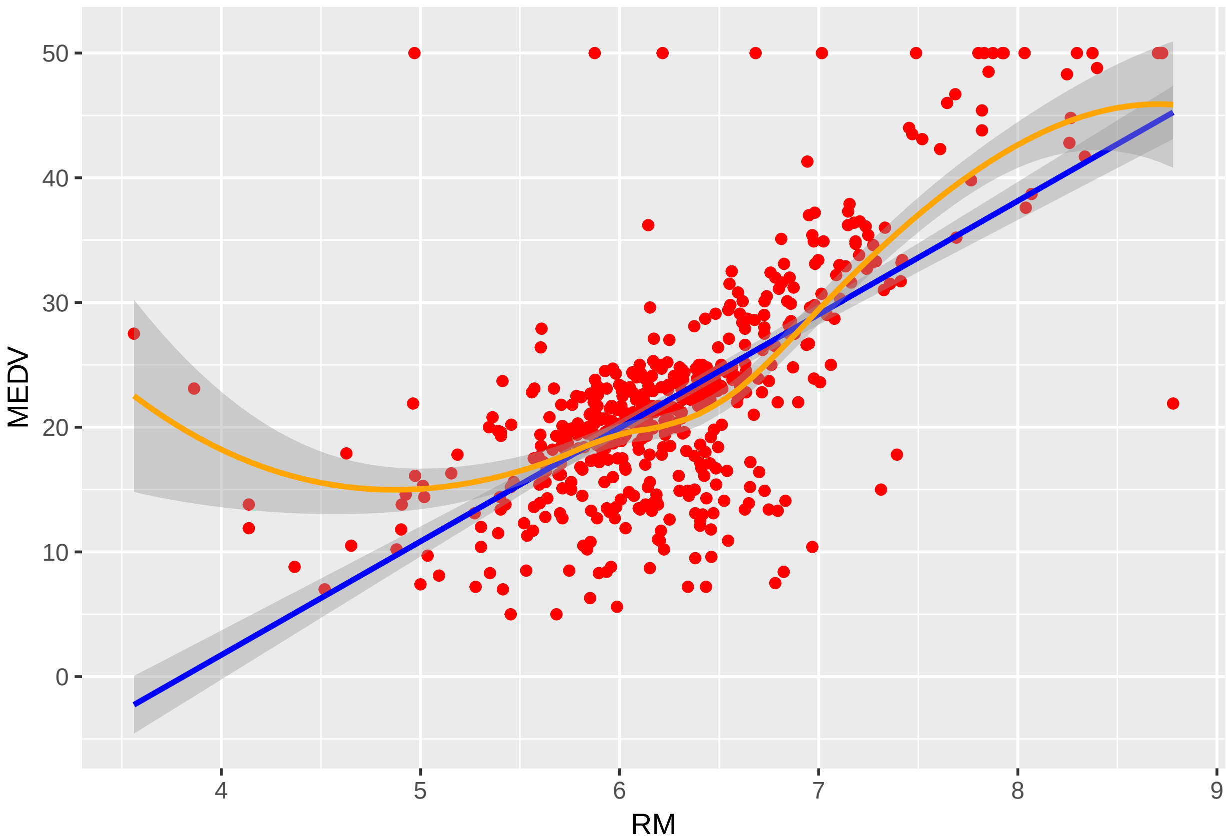
Let us visualize the distribution and density of the outcome, MEDV. The black curve represents the density. In addition, the boxplot is also plotted to bring an additional perspective. We see that the median value of housing price is skewed to the right, with a number of outliers to the right. It may be useful to transform 'MEDV' column using functions like natural logarithm, while modeling the hypothesis for regression analysis.

```r
p01 <- ggplot(df, aes(x = MEDV)) + geom_histogram(aes(y = ..density..), binwidth = 1,
    colour = "black", fill = "green") + geom_density(alpha = 0.8, fill = "#FF6666") +
```

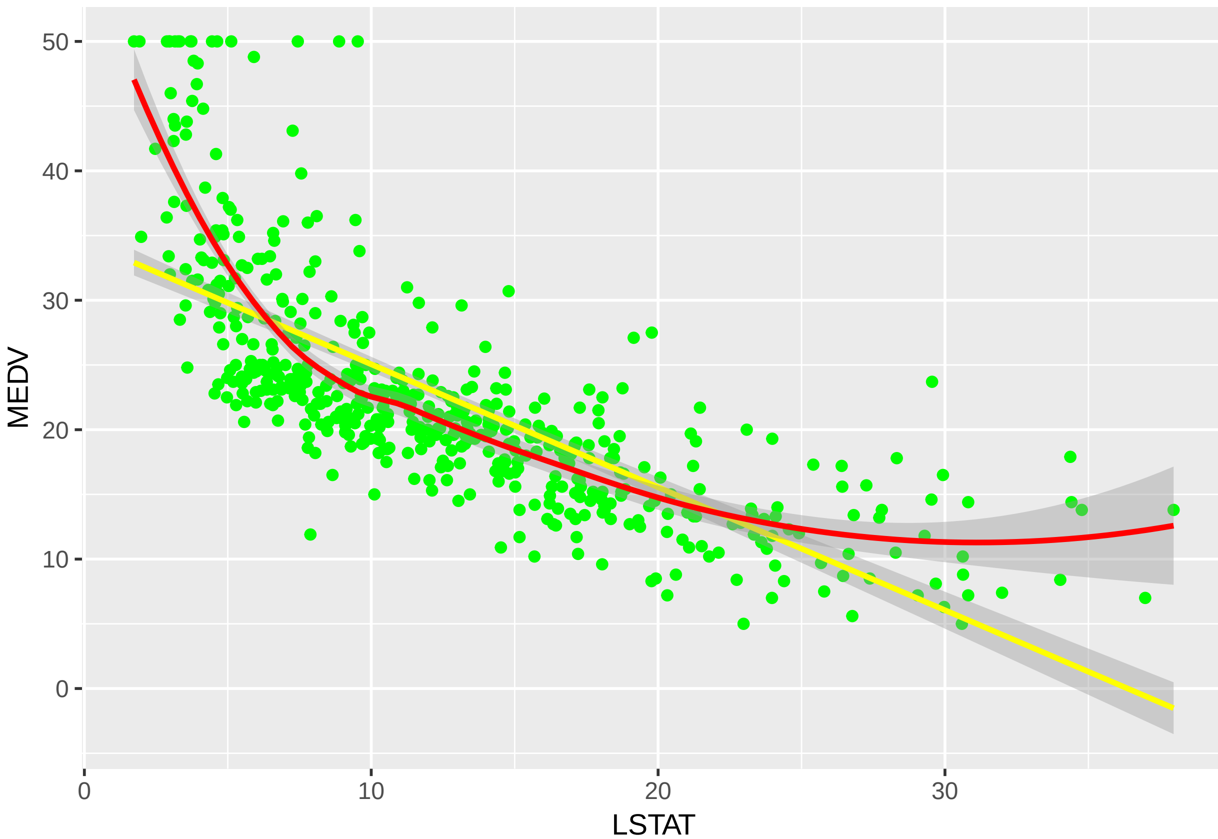
```
    ylim(0, 0.075)
p01
```



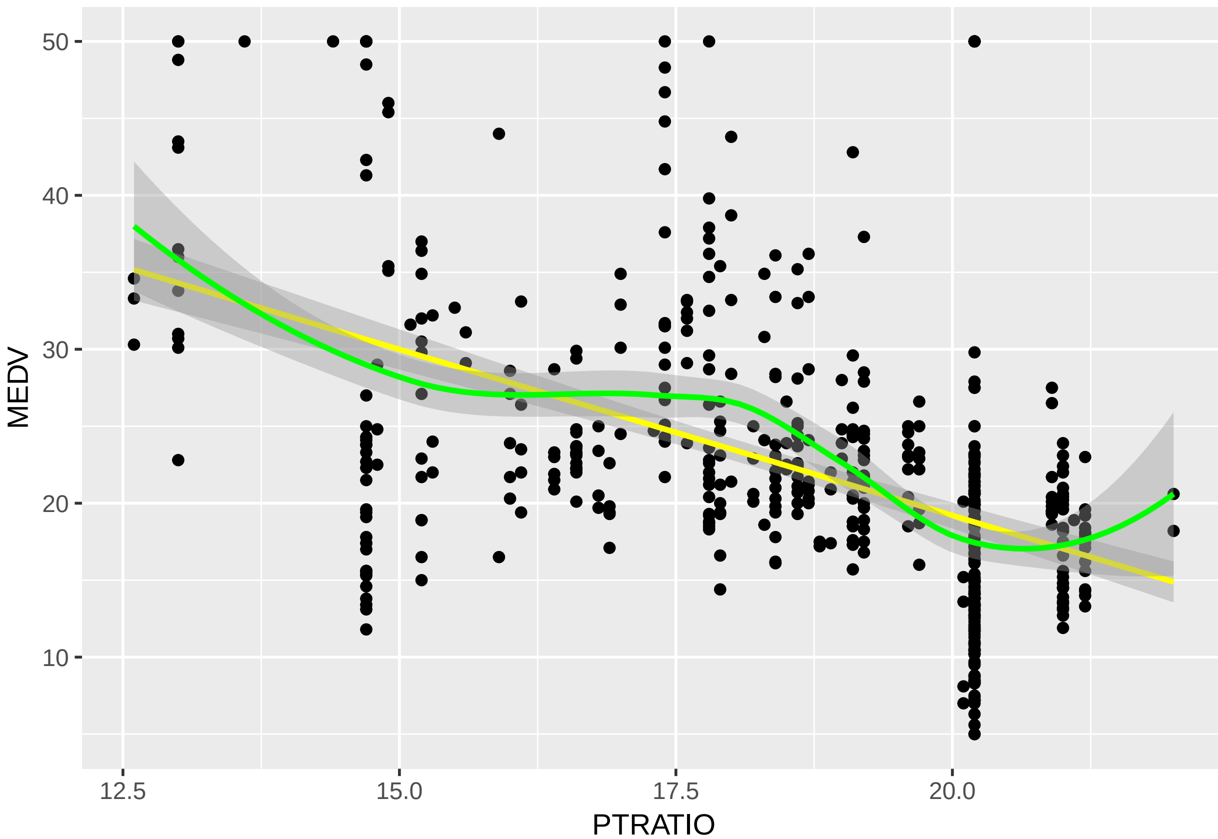

```
p02 <- ggplot(df, aes(y = MEDV, x = RM)) + geom_point(colour = "red") + geom_smooth(method = lm,
    colour = "blue") + geom_smooth(colour = "orange")
p02
```




```
p03 <- ggplot(df, aes(y = MEDV, x = LSTAT)) + geom_point(colour = "green") +
    geom_smooth(method = lm, colour = "yellow") + geom_smooth(colour = "red")
p03
```

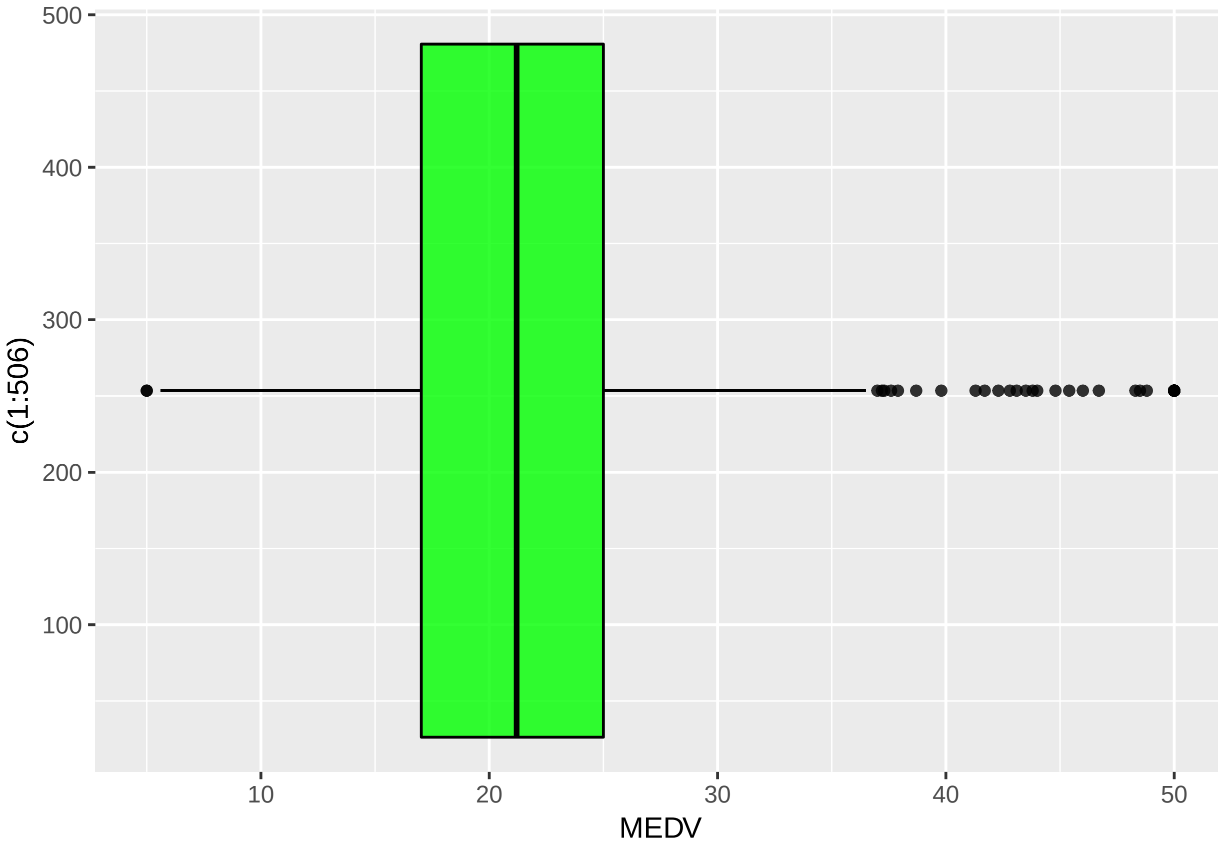



```
p04 <- ggplot(df, aes(y = MEDV, x = PTRATIO)) + geom_point(colour = "black") +
    geom_smooth(method = lm, colour = "yellow") + geom_smooth(colour = "green")
p04
```



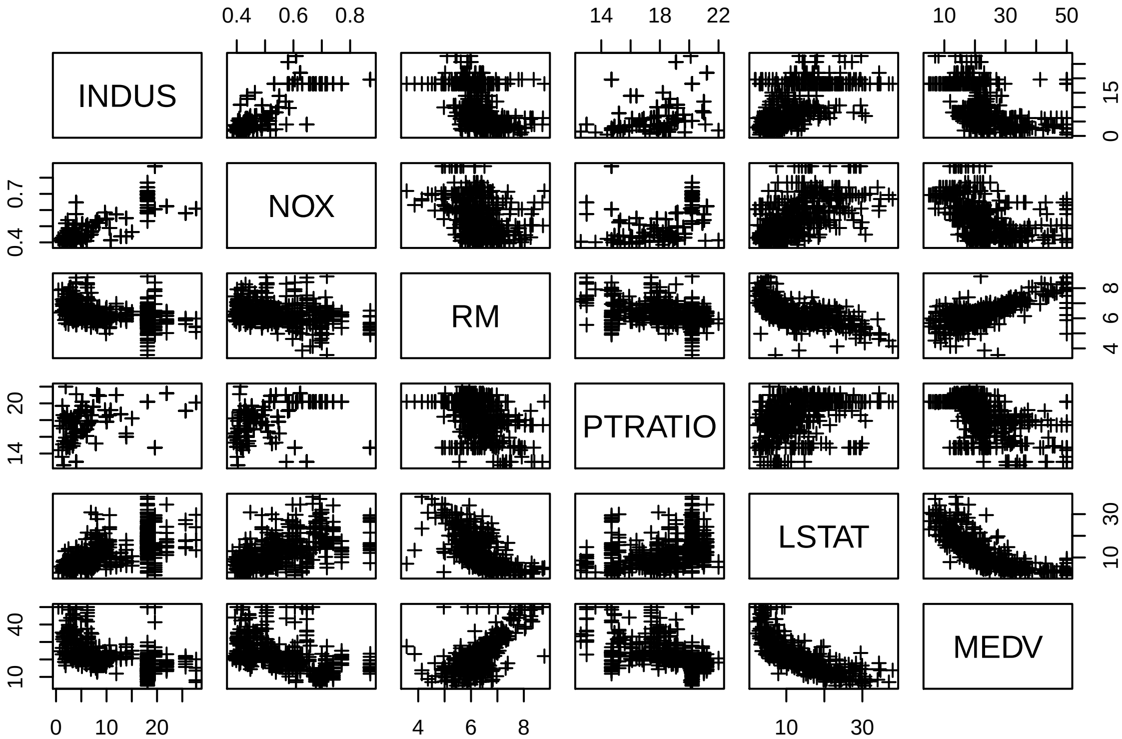

```
b0x <- ggplot(df, aes(y = MEDV, x = c(1:506))) + geom_boxplot(alpha = 0.8, colour = "black",
    fill = "green") + coord_flip()
b0x
```

```
## Warning: Continuous x aesthetic -- did you forget aes(group=...)?
```



Now, let us do scatter plot of some of the important variables (based on intuition) with the outcome variable MEDV. We see that there is strong positive or negative correlation between these variables and the outcome. It is also obviously evident that INDUS and NOX are strongly positively correlated with one another, as nitric oxide levels tend to go up with increase in industries.

```
plot(df[,c(3,5,6,11,13,14)],pch=3)
```



Correlation and near zero variance A few important properties to check now are the correlation of input features with the dependent variable, and to check if any feature has near zero variance (values not varying much within the column).

```r
suppressMessages(library(caret))
# Correlation of each independent variable with the dependent variable
cor(df, df$MEDV)
```

```
##                [,1]
## CRIM    -0.3883046
## ZN       0.3604453
## INDUS   -0.4837252
## CHAS     0.1752602
## NOX     -0.4273208
## RM       0.6953599
## AGE     -0.3769546
## DIS      0.2499287
## RAD     -0.3816262
## TAX     -0.4685359
## PTRATIO -0.5077867
## B        0.3334608
## LSTAT   -0.7376627
## MEDV     1.0000000
```

We see that the number of rooms RM has the strongest positive correlation with the median value of the housing price, while the percentage of lower status population, LSTAT and the pupil-teacher ratio, PTRATIO, have strong negative correlation. The feature with the least correlation to MEDV is the proximity to Charles River, CHAS.

```r
#### Calulate near zero variance
nzv <- nearZeroVar(df, saveMetrics = TRUE)
sum(nzv$nzv)
```

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

The output shows that there are no variable with zero or near zero variance.

Feature Engineering and Data Partitioning

Next we perform centering and scaling on the input features. Then we partition the data on a 7/3 ratio as training/test data sets.

```r
# Centering/scaling of input features

df <- cbind(scale(df[1:13]), df[14])

set.seed(12345)
#Do data partitioning
inTrain <- createDataPartition(y = df$MEDV, p = 0.70, list = FALSE)
training <- df[inTrain,]
testing <- df[-inTrain,]
```

Regression Models

Linear model 1

First, let us try generalized linear regression model with MEDV as the dependent variable and all the remaining variables as independent variables. We train the model with the training data set. For this linear model, below is the coefficients of all the features, and the intercept. Next, we use the trained model to predict the outcome (MEDV) for the testing data set. A good metric to test the accuracy of the model is to calculate the root-mean squared error, which is given by

$$\sqrt{\sum_{i=1}^{n} \frac{(y_{pred_i} - y_{act_i})^2}{n}}$$

```r
set.seed(12345)
#Try linear model using all features
fit.lm <- lm(MEDV~.,data = training)

#check cooeffs
data.frame(coef = round(fit.lm$coefficients,2))
```

```
##              coef
## (Intercept) 22.68
## CRIM        -0.83
## ZN           1.09
## INDUS       -0.01
## CHAS         0.64
## NOX         -2.32
## RM           2.26
## AGE          0.24
## DIS         -3.35
## RAD          2.89
## TAX         -2.08
## PTRATIO     -2.25
```

```
## B            0.87
## LSTAT       -3.74
```

```r
summary(fit.lm)
```

```
##
## Call:
## lm(formula = MEDV ~ ., data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.1105  -2.8013  -0.6267   1.7328  24.8091
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 22.677332   0.263546  86.047  < 2e-16 ***
## CRIM        -0.828888   0.391508  -2.117  0.03497 *
## ZN           1.089461   0.410333   2.655  0.00830 **
## INDUS       -0.005754   0.510514  -0.011  0.99101
## CHAS         0.635875   0.259995   2.446  0.01496 *
## NOX         -2.315683   0.542545  -4.268 2.56e-05 ***
## RM           2.256830   0.360072   6.268 1.10e-09 ***
## AGE          0.241668   0.456785   0.529  0.59711
## DIS         -3.348724   0.529149  -6.329 7.76e-10 ***
## RAD          2.890625   0.724827   3.988 8.15e-05 ***
## TAX         -2.078324   0.786972  -2.641  0.00865 **
## PTRATIO     -2.250143   0.348884  -6.450 3.83e-10 ***
## B            0.867883   0.305892   2.837  0.00482 **
## LSTAT       -3.743443   0.421175  -8.888  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.94 on 342 degrees of freedom
## Multiple R-squared:  0.7239, Adjusted R-squared:  0.7134
## F-statistic: 68.98 on 13 and 342 DF,  p-value: < 2.2e-16
```

```r
set.seed(12345)

#predict on test set
pred.lm <- predict(fit.lm, newdata = testing)

# Root-mean squared error
rmse.lm <- sqrt(sum((pred.lm - testing$MEDV)^2)/
                   length(testing$MEDV))

c(RMSE = rmse.lm, R2 = summary(fit.lm)$r.squared, P_value = summary(fit.lm)$coefficients[1,4])
```

```
##          RMSE             R2       P_value
##  4.381992e+00  7.239054e-01 8.505559e-234
```

```r
data.frame(RMSE = rmse.lm, R2 = summary(fit.lm)$r.squared, P_value = summary(fit.lm)$coefficients[1,4])
```

```
##       RMSE        R2       P_value
## 1 4.381992 0.7239054 8.505559e-234
```

We see that the RMSE is 4.381992 and the R2R2 value is 0.7239 for this model.

Linear model 2

We also saw that the output variable MEDV was skewed to the right. Performing a log transformation would normalize the distribution of MEDV. Let us perform glm with log(MEDV) as the outcome and all remaining features as input. We see that the RMSE value has reduced for this model.

```
set.seed(12345)
#Try linear model using all features
fit.lm1 <- lm(MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS + RAD + TAX + PTRATIO + B + LSTAT, data = trainin

summary(fit.lm1)
```

```
##
## Call:
## lm(formula = MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS + RAD +
##     TAX + PTRATIO + B + LSTAT, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2477  -2.7846  -0.6335   1.5938  25.0858
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.6822     0.2627  86.338  < 2e-16 ***
## CRIM         -0.8236     0.3901  -2.111  0.03547 *
## ZN            1.0641     0.4058   2.623  0.00912 **
## CHAS          0.6434     0.2569   2.505  0.01272 *
## NOX          -2.2422     0.4994  -4.490 9.75e-06 ***
## RM            2.2912     0.3512   6.525 2.44e-10 ***
## DIS          -3.4281     0.4912  -6.979 1.54e-11 ***
## RAD           2.8606     0.6909   4.141 4.36e-05 ***
## TAX          -2.0617     0.7137  -2.889  0.00411 **
## PTRATIO      -2.2363     0.3426  -6.527 2.40e-10 ***
## B             0.8754     0.3046   2.874  0.00431 **
## LSTAT        -3.6721     0.3966  -9.258  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.928 on 344 degrees of freedom
## Multiple R-squared:  0.7237, Adjusted R-squared:  0.7148
## F-statistic:  81.9 on 11 and 344 DF,  p-value: < 2.2e-16
```

```
set.seed(12345)
#predict on test set
pred.lm1 <- predict(fit.lm1, newdata = testing)
# Root-mean squared error
rmse.lm1 <- sqrt(sum((exp(pred.lm1) - testing$MEDV)^2)/
                   length(testing$MEDV))

c(RMSE = rmse.lm1, R2 = summary(fit.lm1)$r.squared, P_value = summary(fit.lm1)$coefficients[1,4])
```

```
##           RMSE            R2       P_value
##  3.063848e+16  7.236795e-01 3.224089e-235
```

```
c(RMSE = rmse.lm1, R2 = summary(fit.lm1)$r.squared)
```

```
##         RMSE           R2
## 3.063848e+16 7.236795e-01
```

We see that the RMSE is 4.381992 and the R2R2 value is 0.7239 for this model.

Let us examine the calculated p-value for each feature in the linear model. Any feature which is not significant ($p<0.05$) is not contributing significantly for the model, probably due to multicollinearity among other features. We see that the features, ZN, INDUS, and AGE are not significant.

```r
library(car)
summary(fit.lm1)
```

```
##
## Call:
## lm(formula = MEDV ~ CRIM + ZN + CHAS + NOX + RM + DIS + RAD +
##     TAX + PTRATIO + B + LSTAT, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -14.2477  -2.7846  -0.6335   1.5938  25.0858
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.6822     0.2627  86.338  < 2e-16 ***
## CRIM         -0.8236     0.3901  -2.111  0.03547 *
## ZN            1.0641     0.4058   2.623  0.00912 **
## CHAS          0.6434     0.2569   2.505  0.01272 *
## NOX          -2.2422     0.4994  -4.490 9.75e-06 ***
## RM            2.2912     0.3512   6.525 2.44e-10 ***
## DIS          -3.4281     0.4912  -6.979 1.54e-11 ***
## RAD           2.8606     0.6909   4.141 4.36e-05 ***
## TAX          -2.0617     0.7137  -2.889  0.00411 **
## PTRATIO      -2.2363     0.3426  -6.527 2.40e-10 ***
## B             0.8754     0.3046   2.874  0.00431 **
## LSTAT        -3.6721     0.3966  -9.258  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 4.928 on 344 degrees of freedom
## Multiple R-squared:  0.7237, Adjusted R-squared:  0.7148
## F-statistic:  81.9 on 11 and 344 DF,  p-value: < 2.2e-16
```

vif(fit.lm1) Variance inflation factors are computed using vif() for the standard errors of linear model coefficient estimates. It is imperative for the vif to be less than 5 for all the features. We see that the vif is greater than 5 for RAD and TAX.

Linear model 3

Based on all these observations, we now construct a new linear model as below: log(MEDV) ~ CRIM + CHAS + NOX + RM + DIS + PTRATIO + RAD + B + LSTAT

```r
set.seed(12345)
#Try simple linear model using selected features
fit.lm2 <- lm(formula = MEDV ~ CRIM + CHAS + NOX + RM + DIS + PTRATIO +
            RAD + B + LSTAT, data = training)
summary(fit.lm2)
```

```
##
## Call:
## lm(formula = MEDV ~ CRIM + CHAS + NOX + RM + DIS + PTRATIO +
##     RAD + B + LSTAT, data = training)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -15.1246  -3.0435  -0.5736   1.8585  25.5956
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  22.7179     0.2663  85.295  < 2e-16 ***
## CRIM         -0.7004     0.3939  -1.778  0.07628 .
## CHAS          0.6916     0.2600   2.660  0.00817 **
## NOX          -2.6617     0.4912  -5.419 1.12e-07 ***
## RM            2.5425     0.3489   7.287 2.17e-12 ***
## DIS          -2.7546     0.4290  -6.420 4.49e-10 ***
## PTRATIO      -2.6433     0.3199  -8.263 3.06e-15 ***
## RAD           1.4273     0.4357   3.276  0.00116 **
## B             0.9281     0.3088   3.006  0.00284 **
## LSTAT        -3.6778     0.4023  -9.142  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.002 on 346 degrees of freedom
## Multiple R-squared:  0.7137, Adjusted R-squared:  0.7062
## F-statistic: 95.82 on 9 and 346 DF,  p-value: < 2.2e-16
```

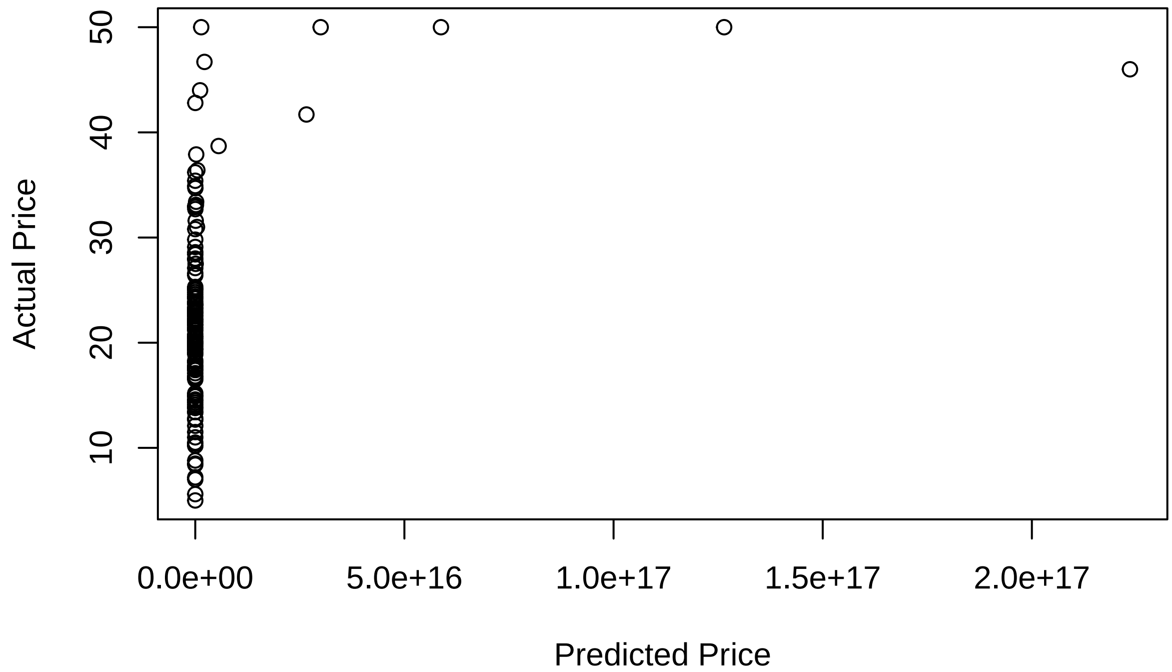
```r
set.seed(12345)
#predict on test set
pred.lm2 <- predict(fit.lm2, newdata = testing)

# Root-mean squared error
rmse.lm2 <- sqrt(sum((exp(pred.lm2) - testing$MEDV)^2)/length(testing$MEDV))

c(RMSE = rmse.lm2, R2 = summary(fit.lm2)$r.squared)
```

```
##         RMSE           R2
## 2.175771e+16 7.136732e-01
```

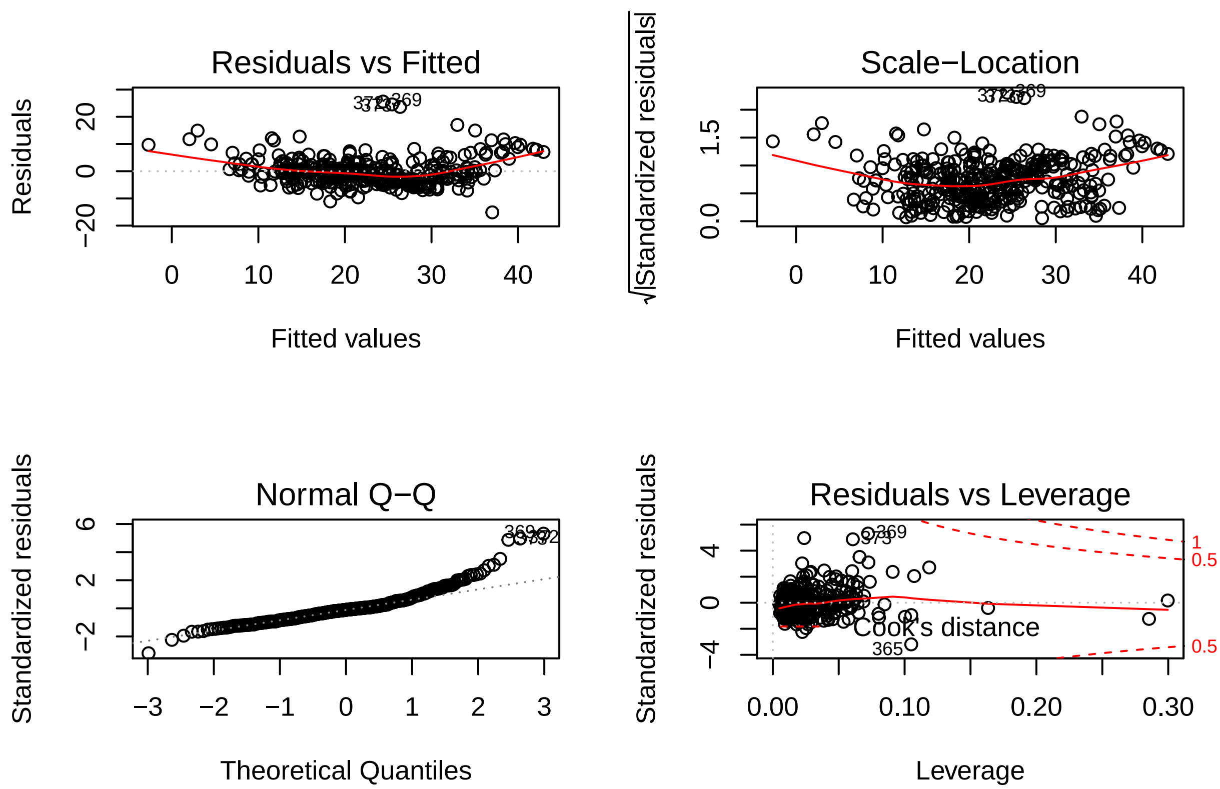
This model is marginally less accurate than linear model 2, based on slight increase in RMSE and slight decrease in R2R2 value. Let us plot the predicted vs actual values of the outcome MEDV.

Linear Model 3 Plot of Predicted Prices vs Actual Prices

```r
plot(exp(pred.lm2),testing$MEDV, xlab = "Predicted Price", ylab = "Actual Price")
```




```
# Diagnostics plots
layout(matrix(c(1,2,3,4),2,2))
plot(fit.lm2)
```



Linear Model 3 – TABLE ###Table Shoming 1st six observations – Actual vs Predicted Price

```
table <- data.frame(x = pred.lm2*10, y = testing$MEDV)
names(table) <- c("Predicted_Price", ylab = "Actual_Price")
```

```r
head(table)
```

```
##    Predicted_Price Actual_Price
## 3         308.8627         34.7
## 7         239.6531         22.9
## 11        200.8473         15.0
## 12        226.6119         18.9
## 15        190.5379         18.2
## 16        191.3903         19.9
```

Random Forest Model

For random forest implementation, we could use the linear model formula of MEDV ~ . (meaning MEDV is the outcome with all other features as input). Inspecting the results, we see that the random forest model has given the best accuracy so far.

```r
suppressMessages(library(randomForest))
set.seed(12345)
fit.rf <- randomForest(formula = MEDV ~ ., data = training)
fit.rf
```

```
##
## Call:
##  randomForest(formula = MEDV ~ ., data = training)
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 4
##
##           Mean of squared residuals: 12.00575
##                     % Var explained: 85.86
```

```r
summary(fit.rf)
```

```
##                 Length Class  Mode
## call              3    -none- call
## type              1    -none- character
## predicted       356    -none- numeric
## mse             500    -none- numeric
## rsq             500    -none- numeric
## oob.times       356    -none- numeric
## importance       13    -none- numeric
## importanceSD      0    -none- NULL
## localImportance   0    -none- NULL
## proximity         0    -none- NULL
## ntree             1    -none- numeric
## mtry              1    -none- numeric
## forest           11    -none- list
## coefs             0    -none- NULL
## y               356    -none- numeric
## test              0    -none- NULL
## inbag             0    -none- NULL
## terms             3    terms  call
```

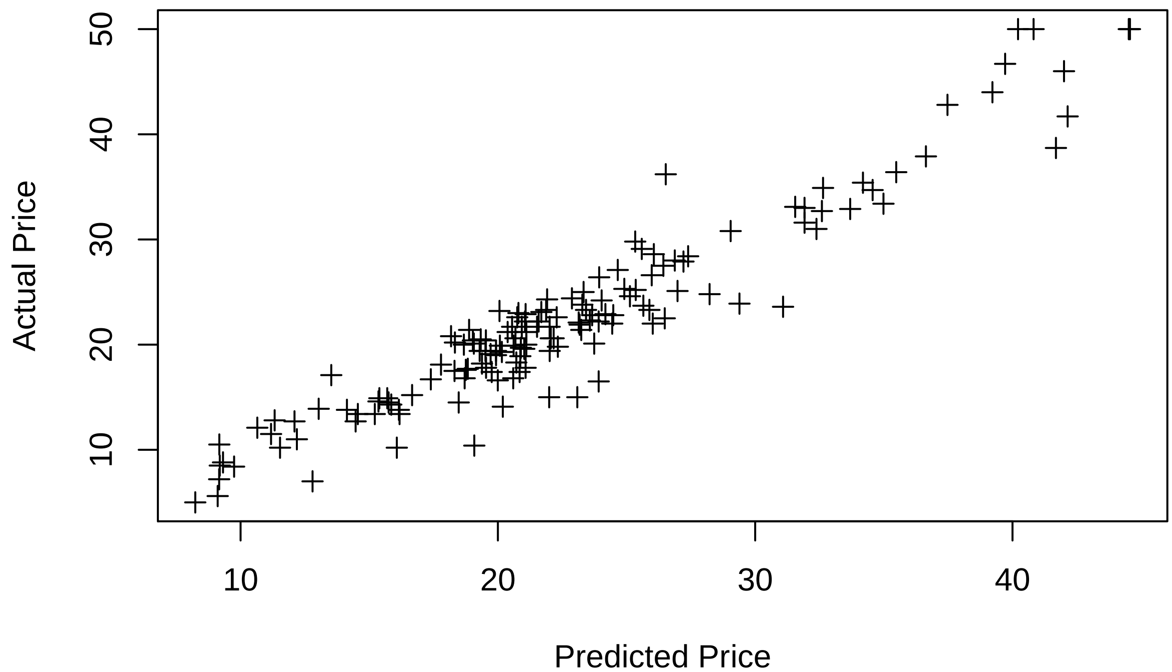
```r
set.seed(12345)
pred.rf <- predict(fit.rf, testing)
```

```
rmse.rf <- sqrt(sum(((pred.rf) - testing$MEDV)^2)/
                    length(testing$MEDV))
R2 = summary(fit.rf$r.squared)
c(RMSE = rmse.rf)
```

```
##     RMSE
## 2.973576
```

Random Forest Model Plot of Predicted Prices vs Actual Prices

```
plot(pred.rf, testing$MEDV, xlab = "Predicted Price", ylab = "Actual Price", pch = 3)
```



RANDOM FOREST MODEL TABLE

Table Shoming 1st six observations – Actual vs Predicted Price

```
table1 <- data.frame(x = pred.rf, y = testing$MEDV)
names(table1) <- c("Predicted_Price", ylab = "Actual_Price")
head(table1)
```

```
##    Predicted_Price Actual_Price
## 3         34.56432         34.7
## 7         21.07742         22.9
## 11        21.99309         15.0
## 12        20.87224         18.9
## 15        19.38131         18.2
## 16        20.61029         19.9
```

MODEL COMPARISON

```r
Linear_Model_1 <- c(RMSE = rmse.lm, R2 = summary(fit.lm)$r.squared)
Linear_Model_2 <- c(RMSE = rmse.lm1, R2 = summary(fit.lm1)$r.squared)
Linear_Model_3 <- c(RMSE = rmse.lm2, R2 = summary(fit.lm2)$r.squared)
Random_Forest_Model <- c(RMSE = rmse.rf, R2 = mean(fit.rf$rsq))
model_comparison <- rbind(Linear_Model_1, Linear_Model_2, Linear_Model_3, Random_Forest_Model)
model_comparison
```

```
##                             RMSE        R2
## Linear_Model_1      4.381992e+00 0.7239054
## Linear_Model_2      3.063848e+16 0.7236795
## Linear_Model_3      2.175771e+16 0.7136732
## Random_Forest_Model 2.973576e+00 0.8516357
```

First 6 predictions

```r
head(table1)
```

```
##    Predicted_Price Actual_Price
## 3         34.56432         34.7
## 7         21.07742         22.9
## 11        21.99309         15.0
## 12        20.87224         18.9
## 15        19.38131         18.2
## 16        20.61029         19.9
```

Conclusion

We experimented with several linear regression models and a random forest model to predict the housing prices in Boston suburbs. Among these models, the Random forest model with a simple linear relationship between the outcome and all input features yielded the best model to predict outcomes, as determined from having the smallest RMSE (i.e., root mean squared error) and the highest R2 (i.e., accuracy | R-squared statistic, and the smallest p-value (greatest significance).