House Price Prediction

Md Sayeef Alam

31/03/2021

##HOUSE PRICE PREDICTION

Using the S&P Case-Schiller Home Price Index as a proxy for home prices building a data science model

Prerequisites:

Installing the required packages

```
library(randomForest)
library(tidyverse)
library(caret)
library(xgboost)
library(ggplot2)
library(neuralnet)
library(e1071)
```

Importing the data set

```
d = read.csv("/Users/mdsayeefalam/Downloads/hpi.csv")
```

#Exploratory Data Analysis:

Checking the data types of the variables, summary statistics and distribution of the variables

```
str(d)
```

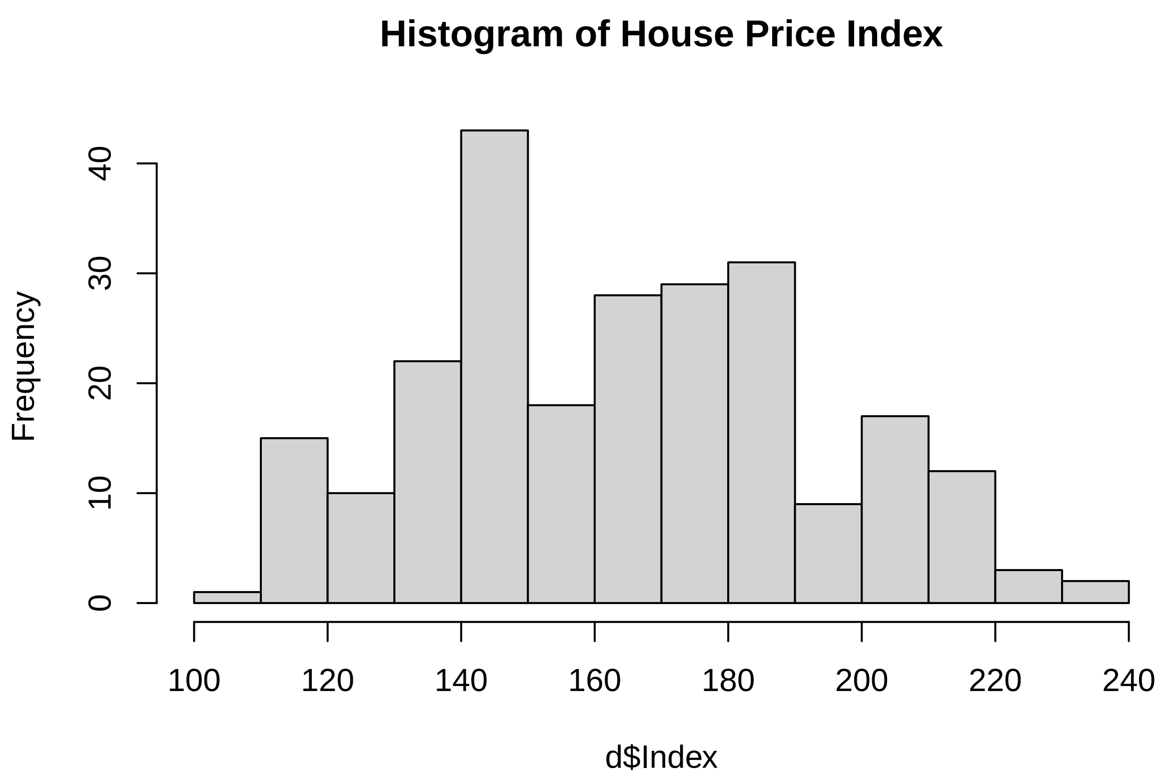
```
## 'data.frame':    240 obs. of  13 variables:
##  $ DATE                 : chr  "01/01/01" "01/02/01" "01/03/01" "01/04/01" ...
##  $ Index                : num  110 110 111 112 112 ...
##  $ TotalHouses          : int  1600 1625 1590 1649 1605 1636 1670 1567 1562 1540 ...
##  $ vacantland           : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ inflation            : num  NA NA NA NA NA NA NA NA NA NA ...
##  $ PPI                  : num  142 142 142 142 144 ...
##  $ RVR                  : num  8.2 8.2 8.2 8.3 8.3 8.3 8.4 8.4 8.4 8.8 ...
##  $ MSP                  : int  169800 169800 169800 179000 179000 179000 172500 172500 172500 171100
##  $ unemployment.rate    : num  4.2 4.2 4.3 4.4 4.3 4.5 4.6 4.9 5 5.3 ...
##  $ population           : int  283960 284166 284380 284602 284834 285076 285324 285584 285842 286086
##  $ existinghomesales    : int  117786 117786 117786 117786 118216 118216 118216 118635 118635 118635
##  $ monthlysupplyofhouses: num  3.8 3.7 3.8 3.9 4 4.2 4.2 4.4 4.4 4.3 ...
##  $ permit               : int  1699 1656 1659 1666 1665 1626 1598 1615 1565 1566 ...
```

```
summary(d)
```

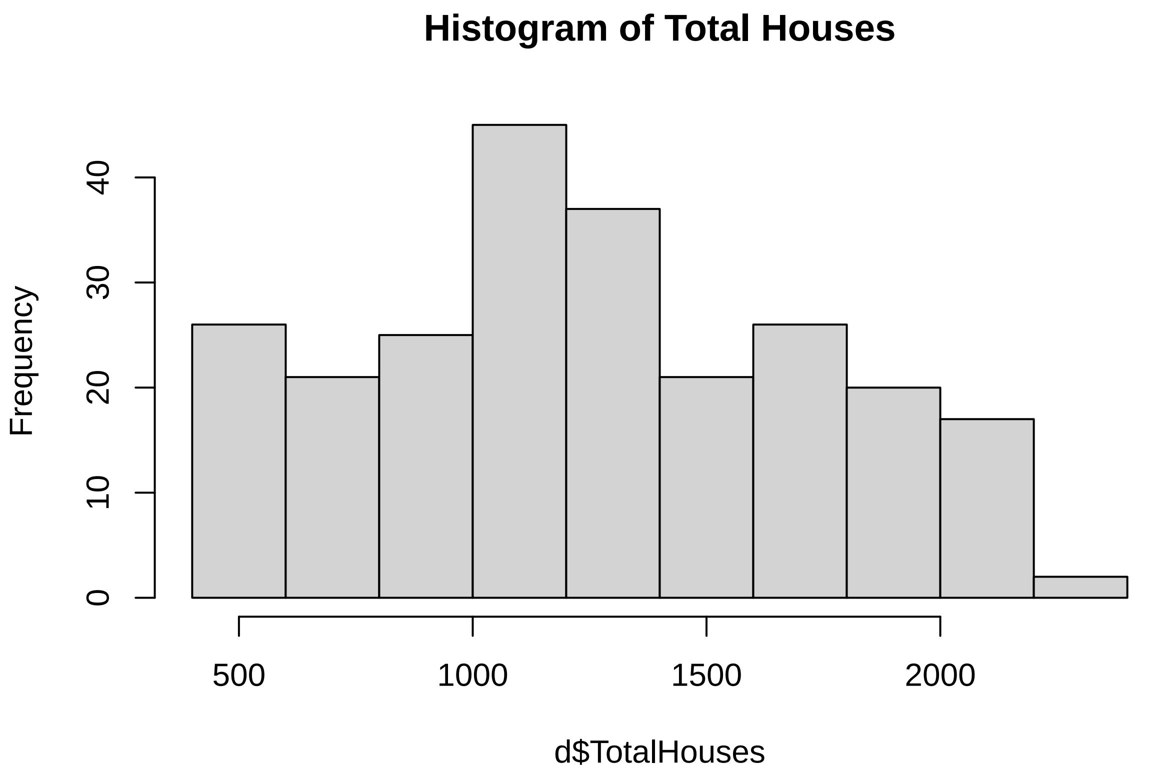
```
##      DATE               Index        TotalHouses       vacantland
##  Length:240         Min.   :109.8   Min.   : 478.0   Min.   : 5863
##  Class :character   1st Qu.:142.8   1st Qu.: 916.5   1st Qu.: 8593
##  Mode  :character   Median :163.6   Median :1204.5   Median :18107
```

```
##                      Mean   :164.5   Mean   :1256.7   Mean   :16441
##                      3rd Qu.:183.3   3rd Qu.:1630.0   3rd Qu.:22448
##                      Max.   :235.6   Max.   :2273.0   Max.   :32038
##                                                        NA's   :107
##    inflation          PPI              RVR               MSP
##  Min.   :1.290   Min.   :141.7   Min.   : 5.700   Min.   :169800
##  1st Qu.:1.845   1st Qu.:176.3   1st Qu.: 7.075   1st Qu.:221050
##  Median :2.100   Median :201.3   Median : 8.800   Median :242000
##  Mean   :2.080   Mean   :194.9   Mean   : 8.629   Mean   :254252
##  3rd Qu.:2.310   3rd Qu.:214.8   3rd Qu.: 9.900   3rd Qu.:300475
##  Max.   :2.710   Max.   :248.2   Max.   :11.100   Max.   :346800
##  NA's   :109
##  unemployment.rate   population     existinghomesales monthlysupplyofhouses
##  Min.   : 3.500    Min.   :283960   Min.   :117786    Min.   : 3.500
##  1st Qu.: 4.700    1st Qu.:297472   1st Qu.:126584    1st Qu.: 4.300
##  Median : 5.600    Median :310940   Median :132110    Median : 5.300
##  Mean   : 6.088    Mean   :309565   Mean   :131112    Mean   : 5.848
##  3rd Qu.: 7.350    3rd Qu.:322064   3rd Qu.:135868    3rd Qu.: 6.625
##  Max.   :14.800    Max.   :330924   Max.   :141241    Max.   :12.200
##
##      permit
##  Min.   : 513
##  1st Qu.: 979
##  Median :1274
##  Mean   :1306
##  3rd Qu.:1665
##  Max.   :2263
##
```

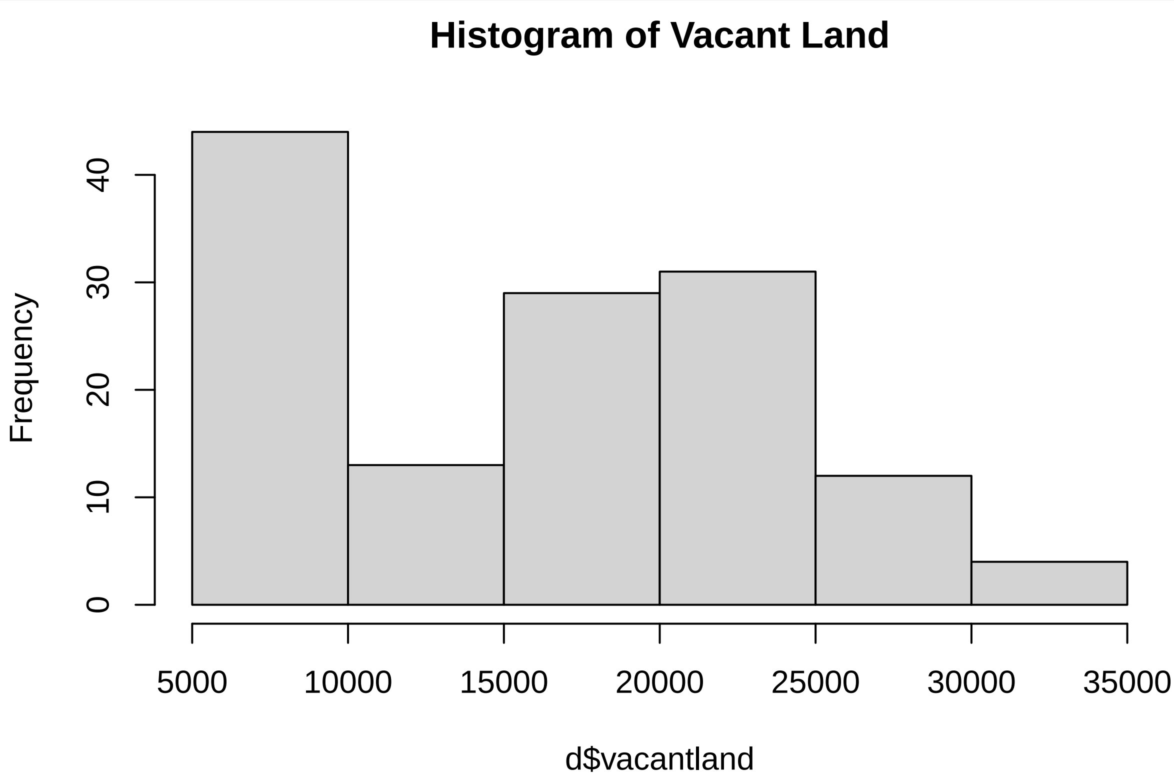
```
hist(d$Index, main = "Histogram of House Price Index")
```



```r
hist(d$TotalHouses, main = "Histogram of Total Houses")
```

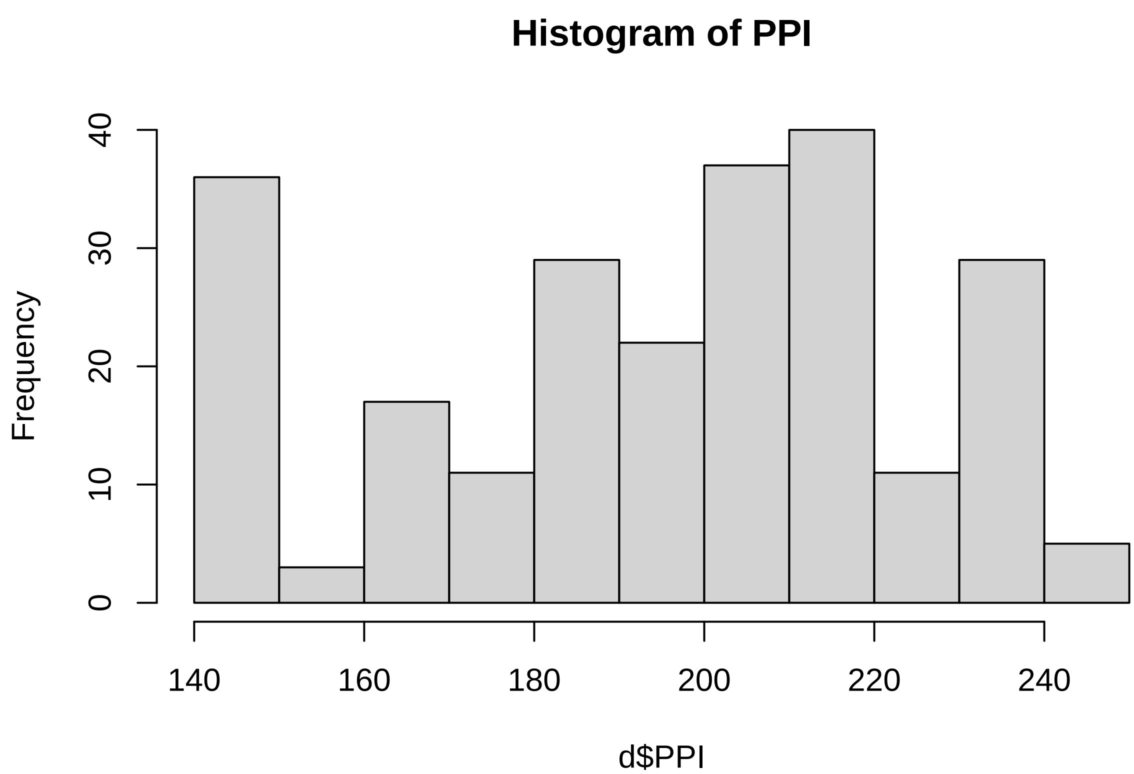



```r
hist(d$vacantland, main = "Histogram of Vacant Land")
```

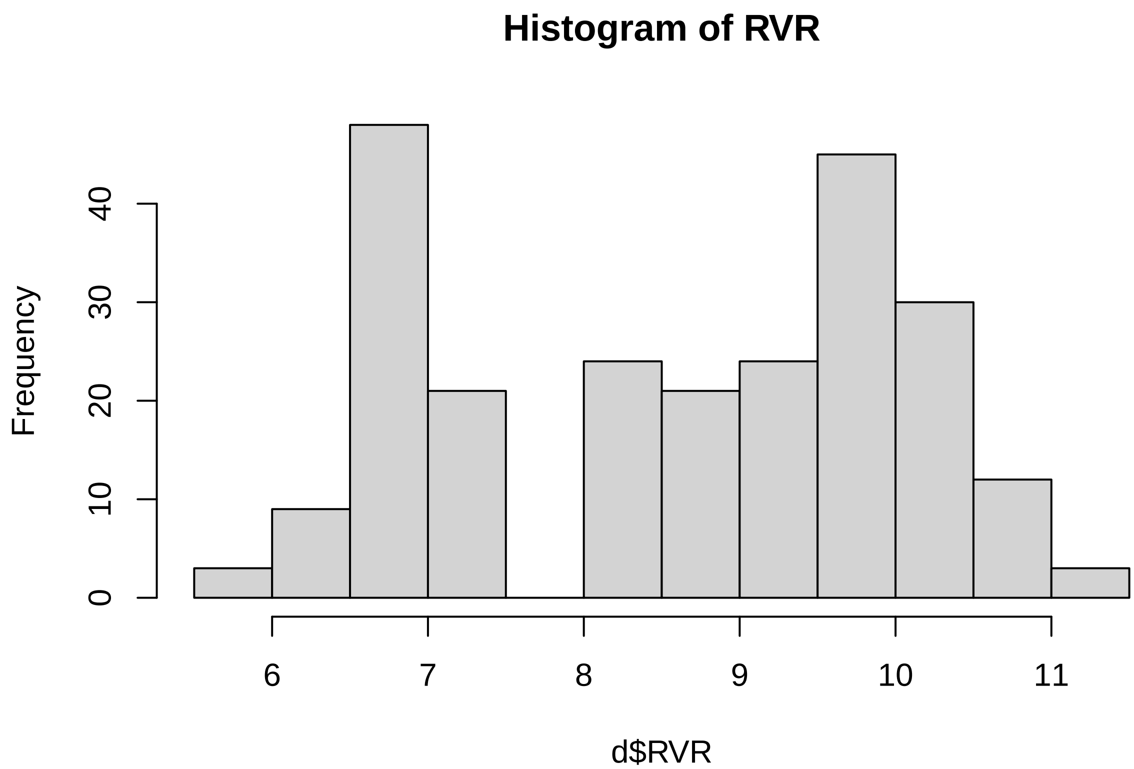


```r
hist(d$inflation, main = "Histogram of Inflation")
```

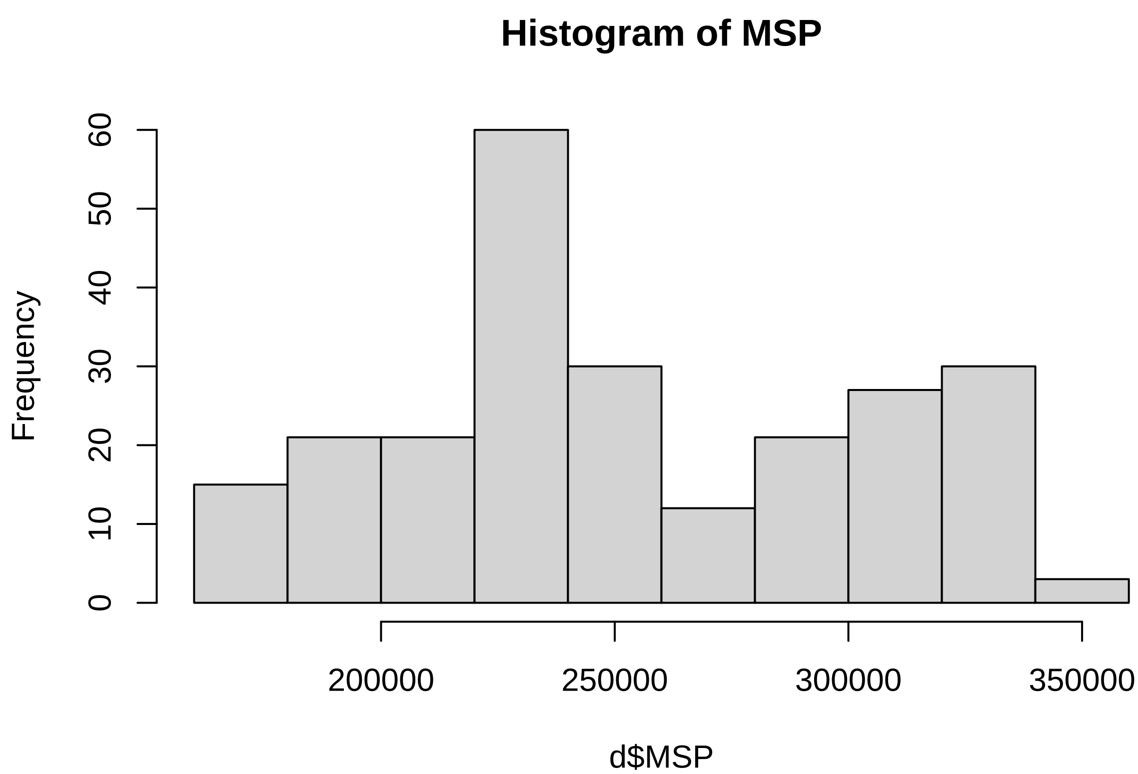

```r
hist(d$PPI, main = "Histogram of PPI")
```



```r
hist(d$RVR, main = "Histogram of RVR")
```




```r
hist(d$MSP, main = "Histogram of MSP")
```



Data cleaning:

Checking for missing values and then either droping or providing adjusted values in their place

```
sapply(d, function(x) sum(is.na(x)))
```

```
##                  DATE                 Index           TotalHouses 
##                     0                     0                     0 
##            vacantland             inflation                   PPI 
##                   107                   109                     0 
##                   RVR                   MSP     unemployment.rate 
##                     0                     0                     0 
##            population     existinghomesales monthlysupplyofhouses 
##                     0                     0                     0 
##                permit 
##                     0
```

So we observe that their are missing values in Vacant Land and Inflation variable.

Feature engineering is not required as all variables are in the required format. Moving onto the data wrangling process.

```
d$vacantland[is.na(d$vacantland)]<-mean(d$vacantland,na.rm=TRUE)
d$inflation[is.na(d$inflation)]<-mean(d$inflation,na.rm=TRUE)
d = select(d,-1)
```

Now that we have complete data we can proceed with model development but prior to that lets divide our dataset into training and testing to later check for robustness of the models used.

```
dt = sort(sample(nrow(d), nrow(d)*.8))
train<-d[dt,]
test<-d[-dt,]
```

Let us now fit our models to the training dataset:

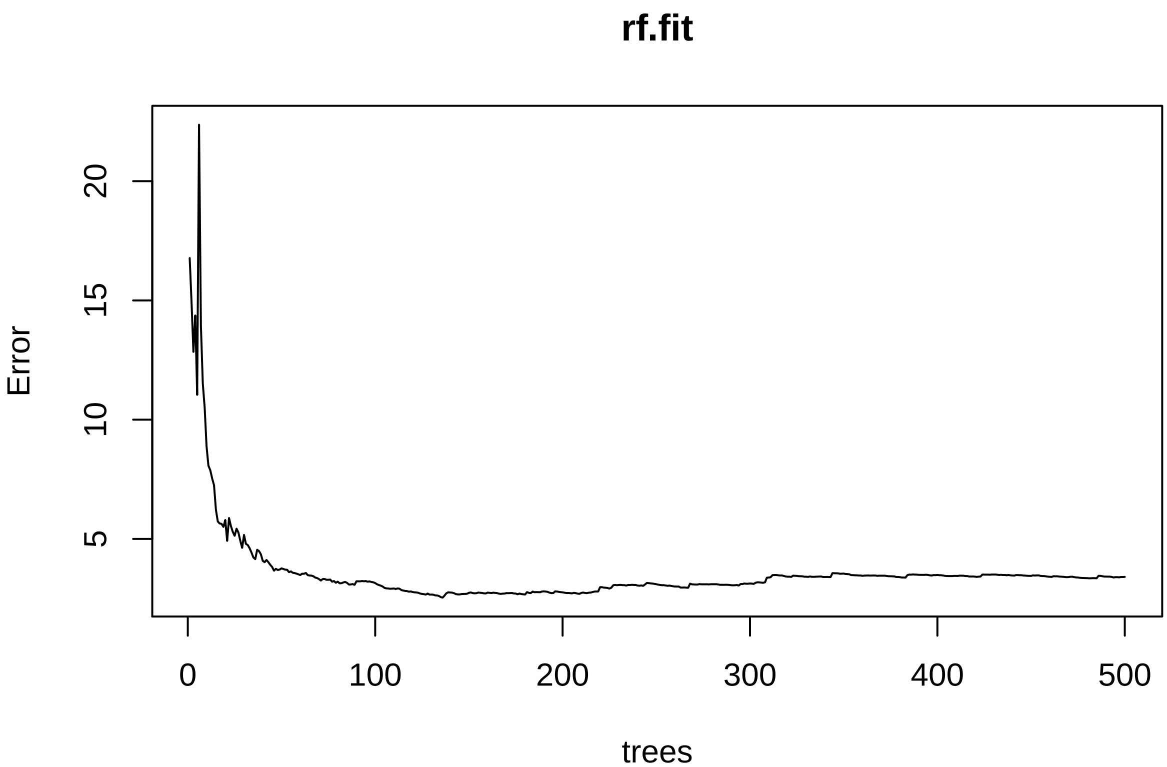
1. Random Forest Regression
2. Linear Regression
3. XGBoost Regression

For the random forest regression we have;

```
rf.fit <- randomForest(Index ~ ., data = train, mtry = 3, importance = TRUE, na.action = na.omit)
print(rf.fit)
```

```
## 
## Call:
##  randomForest(formula = Index ~ ., data = train, mtry = 3, importance = TRUE,      na.action = na.omi
##                Type of random forest: regression
##                      Number of trees: 500
## No. of variables tried at each split: 3
## 
##           Mean of squared residuals: 3.407833
##                     % Var explained: 99.57
```

```r
plot(rf.fit)
```



For linear regression we have;

```r
lr.fit = lm(Index ~ ., data = train)
summary(lr.fit)
```

```
##
## Call:
## lm(formula = Index ~ ., data = train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.0376 -2.2620 -0.2589  2.1251  7.5327
##
## Coefficients:
##                        Estimate Std. Error t value Pr(>|t|)
## (Intercept)          -7.202e+01  4.606e+01  -1.563   0.1197
## TotalHouses           2.305e-03  3.399e-03   0.678   0.4985
## vacantland            6.110e-04  1.139e-04   5.365 2.47e-07 ***
## inflation             1.672e+00  2.129e+00   0.786   0.4332
## PPI                   1.001e-01  6.627e-02   1.510   0.1327
## RVR                  -1.389e+00  6.049e-01  -2.296   0.0228 *
## MSP                   2.726e-04  3.336e-05   8.171 5.26e-14 ***
## unemployment.rate    -3.804e-02  2.740e-01  -0.139   0.8897
## population           -3.690e-03  3.101e-04 -11.897  < 2e-16 ***
## existinghomesales     9.484e-03  6.750e-04  14.049  < 2e-16 ***
## monthlysupplyofhouses 2.077e+00  3.674e-01   5.654 6.05e-08 ***
## permit                2.318e-02  3.562e-03   6.507 7.36e-10 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.205 on 180 degrees of freedom
## Multiple R-squared:  0.9877, Adjusted R-squared:  0.987
## F-statistic:  1318 on 11 and 180 DF,  p-value: < 2.2e-16
```

For XGBoost regression we have;

```
xgb.fit = xgboost(data = as.matrix(train[, 2:12]),
                  label = as.matrix(train[, 1]),
                  nrounds = 1000,
                  objective = "reg:squarederror",
                  early_stopping_rounds = 3,
                  max_depth = 6,
                  eta = .25)
```

Some more models to be compared, like Support Vector Regression and the infamous Neural Network

```
svr.fit = svm(Index ~ ., data = train)
```

Code for neural network model

```
nn.fit = neuralnet(Index ~ .,
                data = train, hidden = c(5,5),
                linear.output = TRUE)
```

The above models were tuned or used the default hyperparameters and would yield a much robust prediction if properly implemented.

Now let us check the validity of model on test dataset

```
pred_randomForest = predict(rf.fit, test[,-1])
pred_lr = predict(lr.fit, test[-1])
pred_xgb = predict(xgb.fit,as.matrix(test[,-1]))
pred_svr = predict(svr.fit, test[-1])
pred_nn = predict(nn.fit, test[,-1])
```

We are considering the root mean square error for model validity lower the RMSE scores better is the performance.

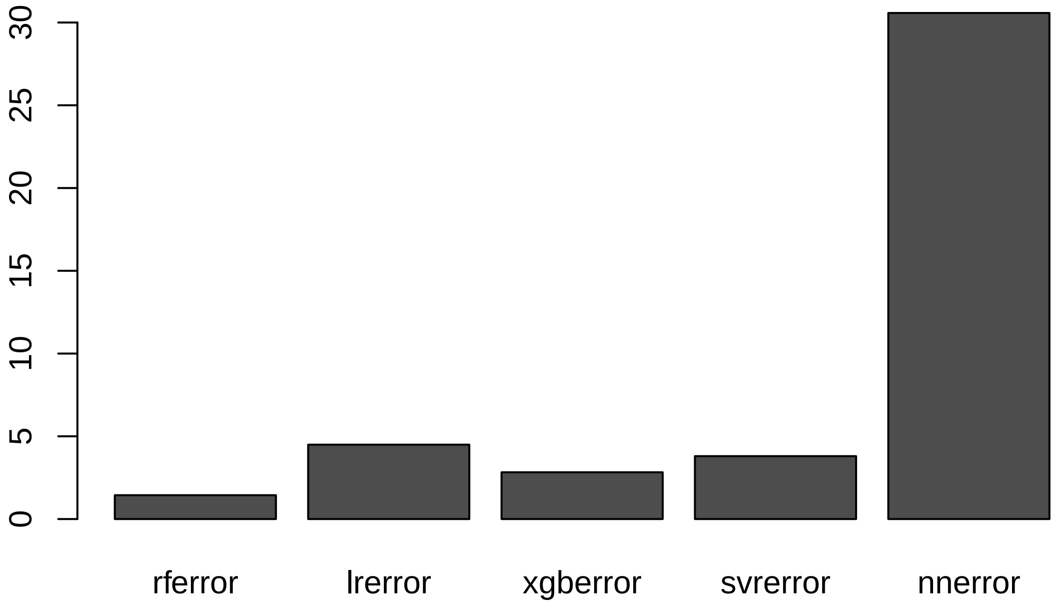
```
rferror = RMSE(pred_randomForest,test$Index)
lrerror = RMSE(pred_lr,test$Index)
xgberror = RMSE(pred_xgb,test$Index)
svrerror = RMSE(pred_svr,test$Index)
nnerror = RMSE(pred_nn,test$Index)
```

Visualizing the result for easier interpretation and choice of model.

```
modelchoice = cbind(rferror,lrerror,xgberror,svrerror,nnerror)
modelchoice
```

```
##       rferror  lrerror xgberror svrerror  nnerror
## [1,] 1.442159 4.493648 2.826983 3.799785 30.57536
```

```r
barplot(modelchoice)
```



Hence, Random Forest seems to be the best predictor model among the five with the lowest error rates.

————————————————Thank you.————————————————