Group 159: House Price Prediction in King County

First Name	Last Name	Monday or Tuesday class	Share project with ITMD 525? (Y or N)
Vineet	Sampat	Monday	N
Pranav	Budhkar	Monday	N

Table of Contents

1. Introduction

King County is one the biggest counties in the Seattle – Tacoma – Bellevue metropolitan statistical area. It is also the most populous county in Washington state. Being the most populous, it is a given that housing scenario in this region will have interesting statistics. We have selected the dataset for the period of May 2014 – May 2015 with various conditions and parameters pertaining to a house directly or indirectly influence its price.

Currently, our data is only for Row houses in King's county. There are various factors that impact the price of these row houses e.g. number of bedrooms, number of floors, number of bathrooms, square feet area of individual room, even the location of the house based on the latitude, longitude and zip code.

We have performed various analytical techniques on the data viz. Hypothesis testing, Multiple linear regressions, Random Forest Regression, Support Vector Machine Linear Regression, ANOVA, and Time series. The activities were performed to test few of our hypothesis, build models and predict the price of house with time as a factor. The upcoming document elaborates these activities in details and our findings and analysis of the selected dataset.

2. Data

We have selected this dataset from Kaggle.com the URL is as "https://www.kaggle.com/chaitanya94/house-sales-in-king-county/data", a platform for predictive data modelling and analytics.

The various variables involved in our dataset are as follows:

```
> str(data_proj)
'data.frame':   21613 obs. of  20 variables:
 $ date         : Factor w/ 372 levels "20140502T000000",..: 75 148 144 34 91 131 7 4 51 161 ...
 $ price        : num  510000 569000 470000 685000 480000 495000 595000 400000 930000 455000 ...
 $ bedrooms     : int  4 4 4 4 3 4 3 2 3 5 ...
 $ bathrooms    : num  2.75 1.75 2.5 2.5 2.5 2.5 2.5 1 2.5 3.5 ...
 $ sqft_living  : int  3180 1230 2470 2770 1590 2020 1750 840 3290 3080 ...
 $ sqft_lot     : int  13348 7890 8536 45514 1431 7200 3354 5510 6830 7759 ...
 $ floors       : num  2 1 2 2 2 1 2 1 2 2 ...
 $ waterfront   : int  0 0 0 0 0 0 0 0 0 0 ...
 $ view         : int  0 1 0 0 0 0 0 0 0 0 ...
 $ condition    : int  3 4 3 4 3 5 4 3 3 3 ...
 $ grade        : int  8 7 8 9 8 7 7 7 10 8 ...
 $ sqft_above   : int  3020 1090 2470 2770 1060 1010 1750 840 3290 2310 ...
 $ sqft_basement: int  160 140 0 0 530 1010 0 0 0 770 ...
 $ yr_built     : int  2004 1950 2002 1989 2010 1968 1991 1955 2000 2003 ...
 $ yr_renovated : int  0 0 0 0 0 0 0 0 0 0 ...
 $ zipcode      : int  98019 98004 98155 98077 98144 98034 98033 98136 98052 98019 ...
 $ lat          : num  47.7 47.6 47.8 47.8 47.6 ...
 $ long         : num  -122 -122 -122 -122 -122 ...
 $ sqft_living15: int  3020 2380 1690 2940 1620 1620 1750 1630 3200 2980 ...
 $ sqft_lot15   : int  10029 13176 8840 49495 1548 7275 4286 5510 6227 8223 ...
```

In the original dataset, we an additional variable named "Id". This variable had no purpose in our analysis and hence is removed from the dataset.

Moreover, our dependent y – variable is price on which we will perform predictions and rest are independent variables directly or indirectly influencing the y – variable.

3. Problems to be Solved

Our aim is to analyze the given historical dataset and predict the price of houses in King county, Seattle.

Moreover, we have to test hypothesis that the average price of house is not greater than 50000, with the alternate hypothesis as the average price of house is greater than 50000. And, the average price of houses having 1 and 1.5 floors is the same.

To successfully test the hypothesis, we have performed one - tail and two - tail hypothesis testing. We have also performed ANOVA testing on group mean price with bedroom as our significant influencing variable.

For predicting the house prices, we have built several models using Multiple Regression, Random Forest, Support Vector Machine and Time Series Analysis.

4. Data Processing

Our data is clean and requires no pre-processing in that aspect. However, we have made a few adjustments to the dataset based on our analysis criteria:

a. The variable “Id” is removed as it is not required by us.
b. For time series the “date” variable is processed using the substring() to remove the excess part in the date variable. i.e. “<date>T00000”, the excess “T00000” part is removed.

5. Methods and Process

The date post minute processing had 21613 rows. Since our dataset has less than 30000 records we have applied N – folds cross validation for building models using linear regression techniques. For our dataset, we have taken N = 10, hence, on applying 10 – fold cross validation we got the following model. However, the model we got had low accuracy and also the residual analysis showed <to be added by vineet>. Hence, we applied “log transformation” to the model and got the below model. To use N – fold cross validation we loaded the “caret” library.

```
> housemodel3<-train(log(price)~bedrooms+bathrooms+sqft_lot+waterfront+view+condition+grade+sqft_above+yr_built+yr_renovated+zipcod
e+lat+long+sqft_living15+sqft_lot15,
+                      data = data_proj,
+                      trControl = p.data_control,
+                      method = "lm",
+                      na.action = na.pass)
> vif(housemodel3$finalModel)
     bedrooms     bathrooms      sqft_lot    waterfront          view     condition         grade    sqft_above      yr_built
     1.513849      2.705941      2.099755      1.203750      1.401662      1.236913      3.295698      3.576237      2.230851
 yr_renovated       zipcode           lat          long sqft_living15    sqft_lot15
     1.147263      1.647215      1.171558      1.806971      2.774012      2.129372
> housemodel3$resample
        RMSE  Rsquared       MAE Resample
1  0.2613049 0.7482846 0.2060683   Fold01
2  0.2625765 0.7663233 0.2050364   Fold02
3  0.2639399 0.7459120 0.2042005   Fold03
4  0.2622987 0.7628759 0.2022765   Fold04
5  0.2517973 0.7686665 0.1965689   Fold05
6  0.2569471 0.7634161 0.1994015   Fold06
7  0.2568254 0.7560767 0.2013462   Fold07
8  0.2493972 0.7679580 0.1941936   Fold08
9  0.2476825 0.7661721 0.1935304   Fold09
10 0.2609717 0.7702692 0.2040277   Fold10
> housemodel3$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
  (Intercept)       bedrooms      bathrooms       sqft_lot     waterfront           view      condition          grade
   -1.967e+01      2.779e-03      1.208e-01      4.899e-07      3.732e-01      7.105e-02      6.563e-02      1.736e-01
   sqft_above       yr_built   yr_renovated        zipcode            lat           long  sqft_living15     sqft_lot15
    9.439e-05     -3.481e-03      3.986e-05     -5.653e-04      1.428e+00     -2.036e-01      1.148e-04     -2.249e-07
```

From the 10 – fold cross validation, we got an accurate model that predicts the price of houses in King county.

For time series analysis, we have intentionally withheld the last 100 records to test the trained model and perform evaluation on RMSE, MAE and AIC values.

6. Evaluations and Results

6.1. Evaluation Methods

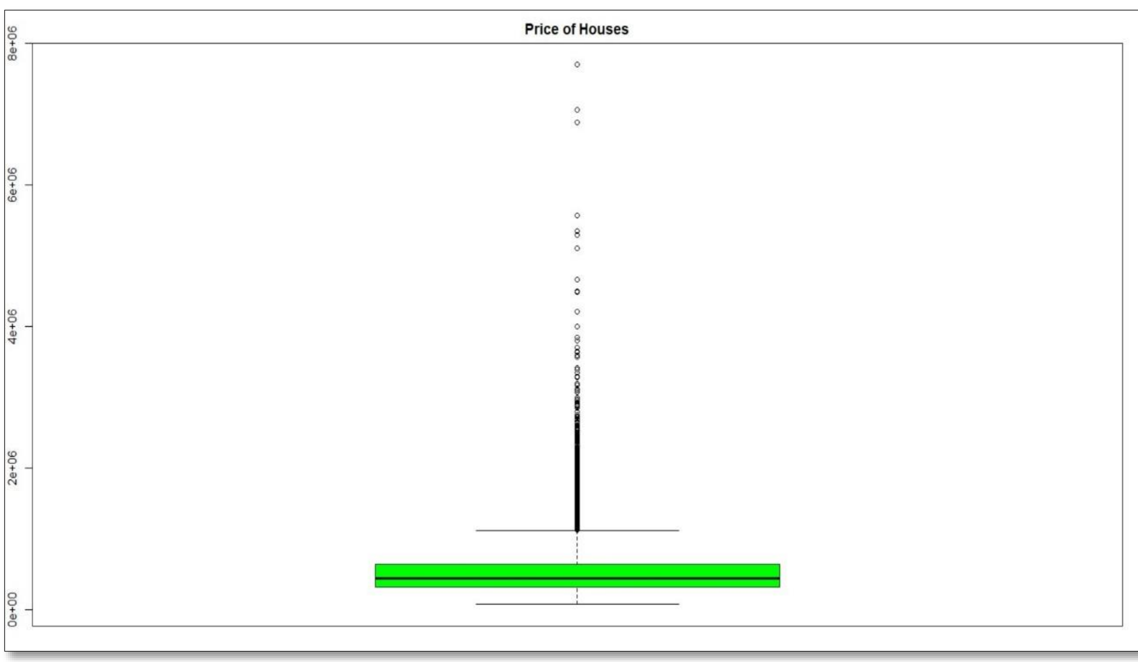
<u>Method 1 Hypothesis Testing:</u>

For Hypothesis testing, we have performed one – tail and Two – tail hypothesis testing.

a. One – tail Hypothesis testing:
    i. Null Hypothesis (H0) – The average price of house is not greater than 50000.
    ii. Alternative Hypothesis (Ha) – The average price of house is greater than 50000.

```
> hs_price = data_proj$price
> mean(hs_price)
[1] 540182.2
> describe(hs_price)
   vars     n     mean       sd median trimmed    mad   min     max   range skew kurtosis      se
X1    1 21613 540182.2 367362.2 450000  481704 222390 75000 7700000 7625000 4.02    34.51 2498.83
> par(mfrow=c(1,1))
> boxplot(hs_price, col = 'green', main = "Price of Houses")
> onet_price = mean(hs_price)
```

We loaded the price data into a variable named “hs_price” and described the same. On plotting the box plot we the following plot:



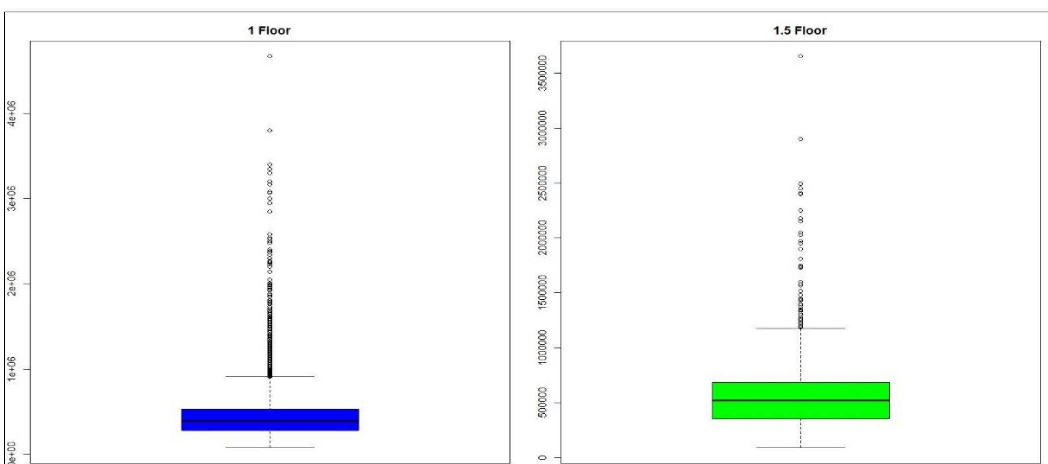
From the above box plot, we see that the variance of the house prices is very large hence we went ahead with performing one-tail hypothesis testing. Below we have calculated the various parameters required in one – tail hypothesis testing.

```
> onet_length = length(hs_price)
> onet_price = mean(hs_price)
> sdev = sd(hs_price)
> onet_length = length(hs_price)
> onet_err = (qnorm(0.975)*sdev)/sqrt(onet_length)
> onet_left = onet_price - onet_err
> onet_right = onet_price + onet_err
> onet_left
[1] 535284.5
> onet_right
[1] 545079.8
```

b. Two – Tail Hypothesis:
   i. Null Hypothesis: Average price of house having 1 floor and 1.5 floors is the same.
   ii. Alternative Hypothesis: Average price of house having 1 floor and 1.5 floors is not same

```
> hs_tt = data_proj$floors
> hs_1floor = data_proj %>% filter(hs_tt==1)
Warning message:
package 'bindrcpp' was built under R version 3.4.4
> hs_1.5floor = data_proj %>% filter(hs_tt==1.5)
> nrow(hs_1floor)
[1] 10680
> nrow(hs_1.5floor)
[1] 1910
> price_1floor = hs_1floor$price
> price_1.5floor = hs_1.5floor$price
> par(mfrow=c(1,2))
> boxplot(price_1floor, col="blue", main="1 Floor")
> boxplot(price_1.5floor, col="green", main="1.5 Floor")
> |
```

The box plot we get is as follows:



From the above plot, we see that there is difference in variance of the price with house having a floor and 1.5 floor. Hence, we perform the following steps for two – tailed hypothesis:

```
> sd_1floor = sd(price_1floor)
> err2_1floor = (qnorm(0.975)*sd_1floor)/sqrt(len_1floor)
> left_1floor = m_1floor - err2_1floor
> right_1floor = m_1floor + err2_1floor
> m_1.5floor = mean(price_1.5floor)
> sd_1.5floor = sd(price_1.5floor)
> len_1.5floor = length(price_1.5floor)
> err2_1.5floor = (qnorm(0.975)*sd_1.5floor)/sqrt(len_1.5floor)
> left_1.5floor = m_1.5floor - err2_1.5floor
> right_1.5floor = m_1.5floor + err2_1.5floor
```

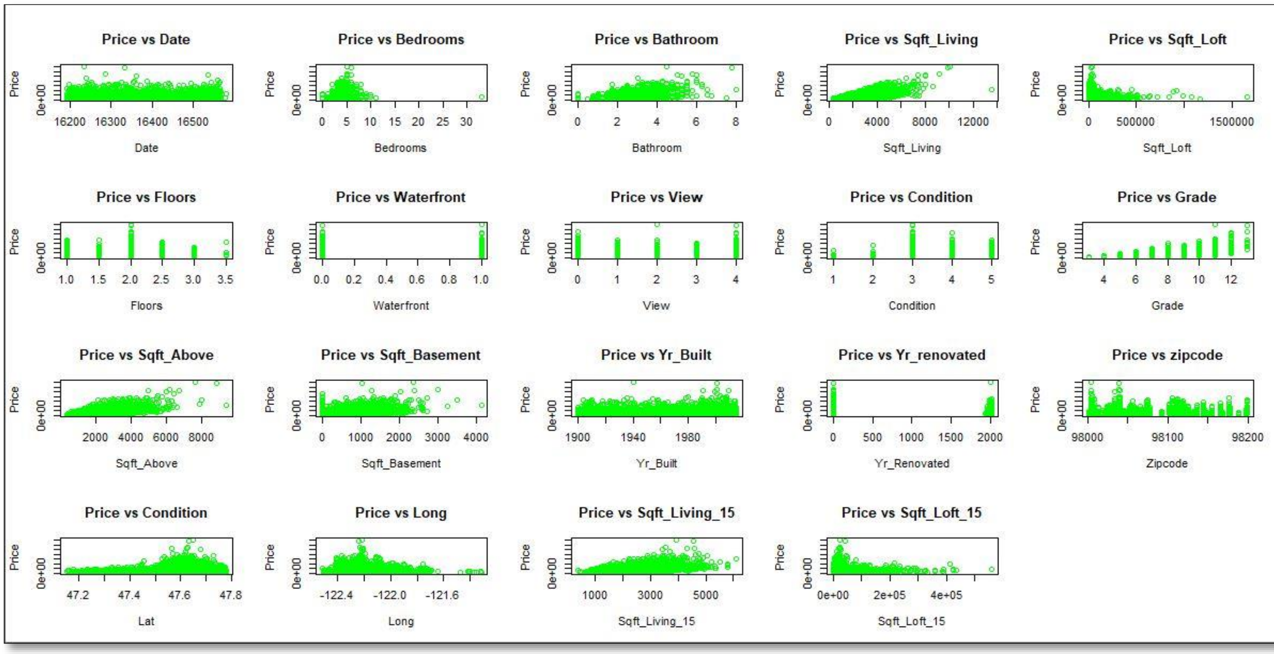
Method 2: Multiple Linear Regression

To check for multicollinearity issue, we have plotted the correlation matrix from which we see that a few variables have high collinearity with each other

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	zipcode	lat	long	sqft_living15	sqft_lot15
date	1	0	-0.02	-0.03	-0.03	0	-0.03	0	0	-0.05	-0.04	-0.03	-0.02	0	-0.02	-0.01	-0.04	-0.01	-0.03	0
price	0	1	0.3	0.52	0.7	0.09	0.25	0.28	0.4	0.04	0.67	0.6	0.31	0.05	0.12	-0.06	0.31	0.02	0.59	0.08
bedrooms	-0.02	0.3	1	0.51	0.57	0.03	0.17	-0.01	0.08	0.03	0.35	0.47	0.3	0.15	0.01	-0.15	-0.01	0.13	0.39	0.03
bathrooms	-0.03	0.52	0.51	1	0.75	0.08	0.5	0.06	0.19	-0.12	0.66	0.68	0.28	0.5	0.05	-0.2	0.02	0.22	0.57	0.08
sqft_living	-0.03	0.7	0.57	0.75	1	0.17	0.35	0.11	0.28	-0.06	0.76	0.88	0.43	0.32	0.05	-0.2	0.05	0.24	0.76	0.18
sqft_lot	0	0.09	0.03	0.08	0.17	1	-0.01	0.03	0.08	-0.01	0.11	0.18	0.01	0.05	0.01	-0.13	-0.09	0.23	0.14	0.72
floors	-0.03	0.25	0.17	0.5	0.35	-0.01	1	0.03	0.03	-0.26	0.46	0.52	-0.25	0.49	0.01	-0.05	0.05	0.12	0.28	-0.01
waterfront	0	0.28	-0.01	0.06	0.11	0.03	0.03	1	0.41	0.02	0.08	0.08	0.08	-0.03	0.1	0.02	-0.01	-0.04	0.09	0.04
view	0	0.4	0.08	0.19	0.28	0.08	0.03	0.41	1	0.05	0.25	0.17	0.27	-0.06	0.1	0.08	0.01	-0.08	0.28	0.08
condition	-0.05	0.04	0.03	-0.12	-0.06	-0.01	-0.26	0.02	0.05	1	-0.14	-0.16	0.17	-0.36	-0.06	0	-0.01	-0.11	-0.09	0
grade	-0.04	0.67	0.35	0.66	0.76	0.11	0.46	0.08	0.25	-0.14	1	0.76	0.16	0.45	0.01	-0.19	0.11	0.2	0.71	0.12
sqft_above	-0.03	0.6	0.47	0.68	0.88	0.18	0.52	0.08	0.17	-0.16	0.76	1	-0.06	0.42	0.03	-0.26	0	0.34	0.73	0.19
sqft_basement	-0.02	0.31	0.3	0.28	0.43	0.01	-0.25	0.08	0.27	0.17	0.16	-0.06	1	-0.13	0.06	0.07	0.11	-0.15	0.2	0.01
yr_built	0	0.05	0.15	0.5	0.32	0.05	0.49	-0.03	-0.06	-0.36	0.45	0.42	-0.13	1	-0.22	-0.35	-0.15	0.41	0.32	0.07
yr_renovated	-0.02	0.12	0.01	0.05	0.05	0.01	0.01	0.1	0.1	-0.06	0.01	0.03	0.06	-0.22	1	0.06	0.03	-0.07	0	0.01
zipcode	-0.01	-0.06	-0.15	-0.2	-0.2	-0.13	-0.05	0.02	0.08	0	-0.19	-0.26	0.07	-0.35	0.06	1	0.27	-0.56	-0.28	-0.15
lat	-0.04	0.31	-0.01	0.02	0.05	-0.09	0.05	-0.01	0.01	-0.01	0.11	0	0.11	-0.15	0.03	0.27	1	-0.14	0.05	-0.1
long	-0.01	0.02	0.13	0.22	0.24	0.23	0.12	-0.04	-0.08	-0.11	0.2	0.34	-0.15	0.41	-0.07	-0.56	-0.14	1	0.33	0.27
sqft_living15	-0.03	0.59	0.39	0.57	0.76	0.14	0.28	0.09	0.28	-0.09	0.71	0.73	0.2	0.32	0	-0.28	0.05	0.33	1	0.19
sqft_lot15	0	0.08	0.03	0.08	0.18	0.72	-0.01	0.04	0.08	0	0.12	0.19	0.01	0.07	0.01	-0.15	-0.1	0.27	0.19	1

Some variables display collinearity issue and hence we plot the individual correlation plots of each x variable with y variable.

The plot for the same is as follows:



From the above, figure we see that variables like sqft_basement, sqft_living, condition, yr_built show multicollinearity issue which are tackled using vif and transformations.

Using Backward elimination by p-value we build our linear model as follows:

```
> p.data_control<-trainControl(method = "cv", number = 10)
> housemodel<-train(price~bedrooms+bathrooms+sqft_living+sqft_lot+waterfront+view+condition+grade+sqft_above+yr_built+yr_renovated+
zipcode+lat+long+sqft_living15+sqft_lot15,
+                   data = data_proj,
+                   trControl = p.data_control,
+                   method = "lm",
+                   na.action = na.pass)
>
> housemodel$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
  (Intercept)     bedrooms    bathrooms   sqft_living     sqft_lot   waterfront         view    condition
    5.727e+06   -3.589e+04    4.274e+04     1.477e+02    1.264e-01    5.831e+05    5.303e+04    2.617e+04
        grade   sqft_above     yr_built   yr_renovated      zipcode          lat         long  sqft_living15
    9.634e+04    3.473e+01   -2.593e+03     2.018e+01   -5.768e+02    6.045e+05   -2.171e+05    2.096e+01
   sqft_lot15
   -3.872e-01

> vif(housemodel$finalModel)
     bedrooms     bathrooms   sqft_living      sqft_lot    waterfront          view     condition         grade    sqft_above
     1.650836      3.124278      7.846355      2.101503      1.203752      1.434339      1.245479      3.390557      5.594685
     yr_built   yr_renovated      zipcode           lat          long  sqft_living15   sqft_lot15
     2.315424      1.147325      1.647680      1.172532      1.812200      2.942773      2.133044
>
```

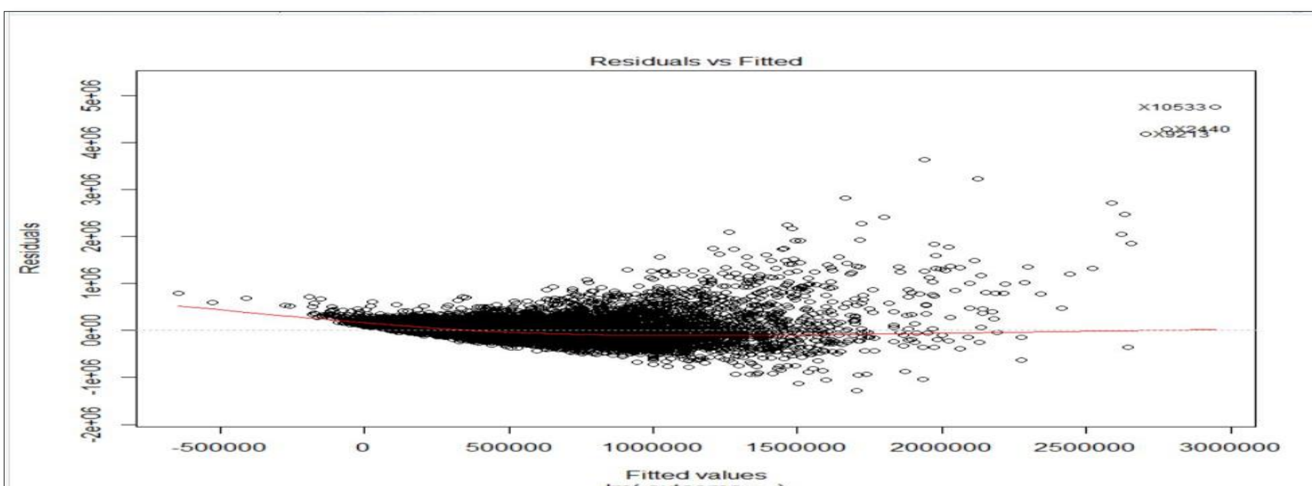
Due to multicollinearity issue and vif value of “sqft_living” variable being greater than 5 we removed that variable and rebuild the model as follows:

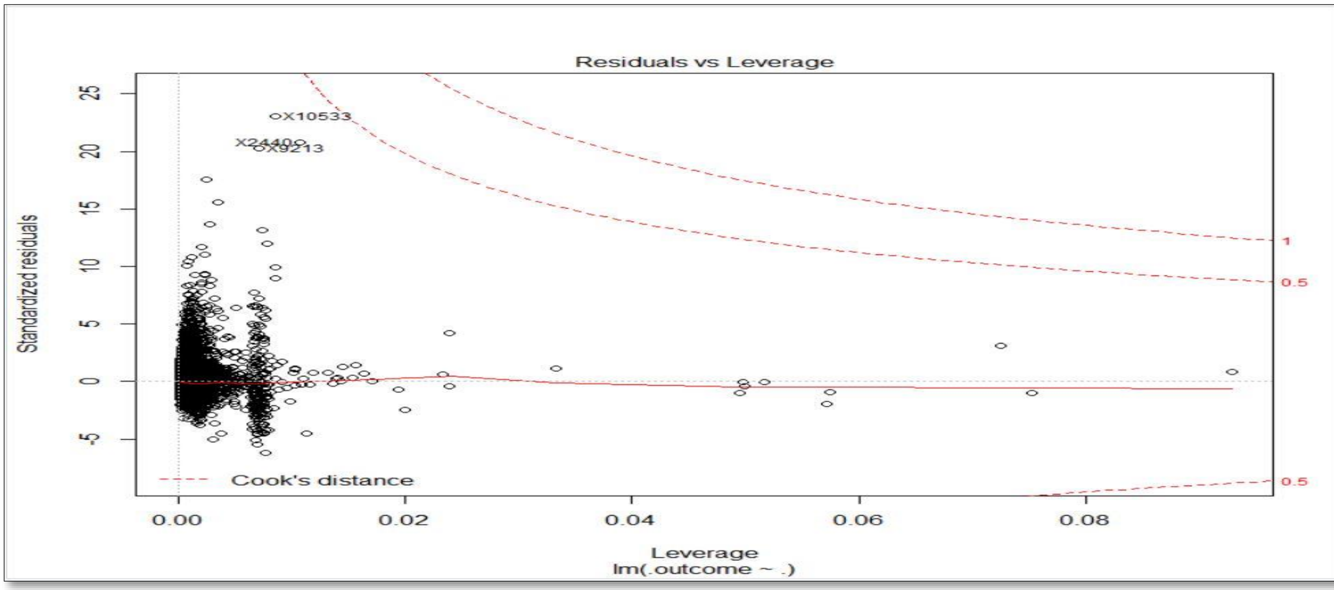
The new model we get is as follows:

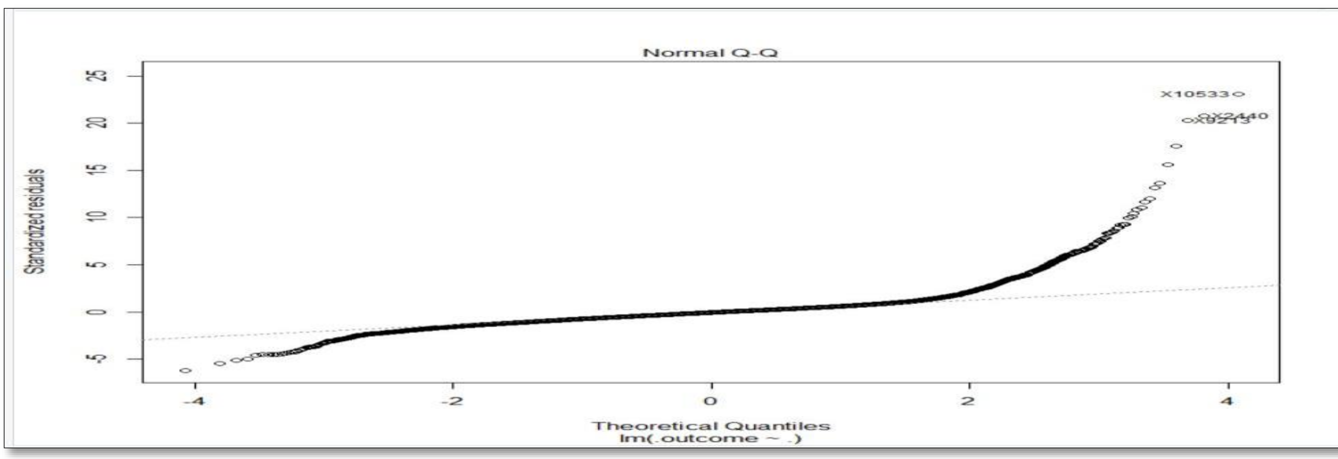
```
> housemodel2<-train(price~bedrooms+bathrooms+sqft_lot+waterfront+view+condition+grade+sqft_above+yr_built+yr_renovated+zipcode+lat
+long+sqft_living15+sqft_lot15,
+                    data = data_proj,
+                    trControl = p.data_control,
+                    method = "lm",
+                    na.action = na.pass)
> vif(housemodel2$finalModel)
     bedrooms     bathrooms      sqft_lot    waterfront          view     condition         grade    sqft_above      yr_built
     1.513849      2.705941      2.099755      1.203750      1.401662      1.236913      3.295698      3.576237      2.230851
 yr_renovated       zipcode           lat          long sqft_living15    sqft_lot15
     1.147263      1.647215      1.171558      1.806971      2.774012      2.129372
>
> housemodel2$resample
       RMSE  Rsquared      MAE Resample
1  229089.2 0.6790680 133617.8   Fold01
2  218840.7 0.6668754 124703.1   Fold02
3  201933.7 0.6824144 124416.1   Fold03
4  192688.0 0.7037446 125356.7   Fold04
5  216317.6 0.6739914 125982.3   Fold05
6  204767.2 0.6704763 131809.7   Fold06
7  199320.9 0.6724396 131481.3   Fold07
8  192803.2 0.6943603 128792.2   Fold08
9  206302.7 0.6828750 131192.3   Fold09
10 210354.6 0.6959618 134266.6   Fold10
> housemodel2$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
  (Intercept)       bedrooms      bathrooms       sqft_lot     waterfront           view      condition          grade
    1.024e+06     -1.662e+04      8.341e+04      1.753e-01      5.840e+05      6.445e+04      3.305e+04      1.090e+05
   sqft_above       yr_built   yr_renovated        zipcode            lat           long  sqft_living15     sqft_lot15
    1.178e+02     -3.072e+03      1.923e+01     -5.573e+02      6.154e+05     -2.419e+05      4.999e+01     -2.798e-01
```

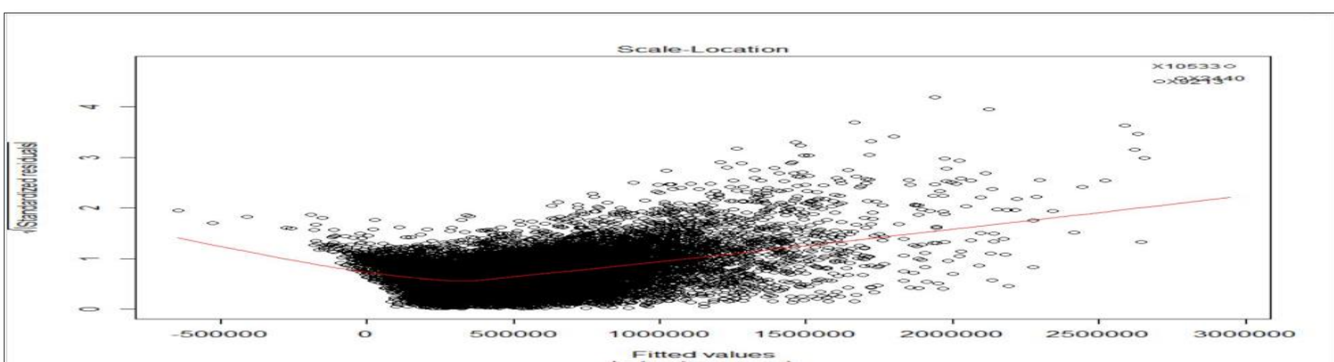
```
>
> housemodel2
Linear Regression

21613 samples
   15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 19451, 19452, 19453, 19452, 19453, 19450, ...
Resampling results:

  RMSE      Rsquared   MAE
  207241.8  0.6822207  129161.8

Tuning parameter 'intercept' was held constant at a value of TRUE
```

The plots, for the above model is as follows:









From the plots above, we see that the residual plots do not show constant variance and the qq plots display that not all points are on lie or near to it. Hence, we have applied transformation to our model.

Method 3: Random Forest Regression Technique

In Random Forest, we have used the “tuneRF” method which gave us the best mtry value as 7. Here is a screenshot depicting the same. Here, we have built the model without transformation getting the following output:

```
320  model_RF<-train(price~.,data = mydata,trControl = data_ctrl,method ="rf",tuneGrid = pGrid)
321
328:1  (Untitled) ÷

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/

Tuning parameter 'mtry' was held constant at a value of 7
> model_RF<-train(price~.,data = mydata,trControl = data_ctrl,method ="rf",tuneGrid = pGrid)
> summary(model_RF)
                Length Class      Mode
call                4  -none-     call
type                1  -none-     character
predicted       21613  -none-     numeric
mse               500  -none-     numeric
rsq               500  -none-     numeric
oob.times       21613  -none-     numeric
importance         18  -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               21613  -none-     numeric
test                0  -none-     NULL
inbag               0  -none-     NULL
xNames             18  -none-     character
problemType         1  -none-     character
tuneValue           1  data.frame list
obsLevels           1  -none-     logical
param               0  -none-     list
> model_RF$resample
      RMSE  Rsquared      MAE Resample
1 119649.4 0.8824887 68883.17   Fold01
2 116685.3 0.8931851 66339.66   Fold02
3 133800.3 0.8856080 69407.55   Fold03
4 128859.0 0.9111685 68320.81   Fold04
```

```
320  model_RF<-train(price~.,data = mydata,trControl = data_ctrl,method ="rf",tuneGrid = pGrid)
321
328:1  (Untitled) ÷

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
mse               500  -none-     numeric
rsq               500  -none-     numeric
oob.times       21613  -none-     numeric
importance         18  -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               21613  -none-     numeric
test                0  -none-     NULL
inbag               0  -none-     NULL
xNames             18  -none-     character
problemType         1  -none-     character
tuneValue           1  data.frame list
obsLevels           1  -none-     logical
param               0  -none-     list
> model_RF$resample
       RMSE  Rsquared      MAE Resample
1  119649.4 0.8824887 68883.17   Fold01
2  116685.3 0.8931851 66339.66   Fold02
3  133800.3 0.8856080 69407.55   Fold03
4  128859.0 0.9111685 68320.81   Fold04
5  122260.5 0.8880970 68472.35   Fold05
6  137420.8 0.8575139 68891.68   Fold06
7  120464.2 0.8865329 66299.93   Fold07
8  121638.2 0.8778632 68768.27   Fold08
9  135999.0 0.8752539 68827.34   Fold09
10 133537.5 0.8669336 67891.40   Fold10
```

```
> model_RF
Random Forest

21613 samples
   18 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 19452, 19451, 19452, 19451, 19453, 19451, ...
Resampling results:

  RMSE      Rsquared   MAE
  127031.4  0.8824645  68210.22

Tuning parameter 'mtry' was held constant at a value of 7
>
```

However, as we see that the RMSE value is coming high we have applied transformation to it. The output of transformation is explained in the next section.

<u>Method 4: Support Vector Machine Regression technique</u>

In SVM, we built the model using SVM linear function and for that the cost factor we got as 1.

The model we built is as follows:

```
##SVM Model without Transformation

model_SVM<-train(price~.,data = mydata,trControl = data_ctrl,method ='svmLinear',tuneGrid = psvm_grid)

summary(model_SVM)
model_SVM$resample
model_SVM$finalModel

model_SVM
```

```
369:1   (Untitled) ÷

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
> model_SVM<-train(price~.,data = mydata,trControl = data_ctrl,method ='svmLinear',tuneGrid = psvm_grid)
> summary(model_SVM)
Length  Class   Mode
     1   ksvm     S4
> model_SVM$resample
       RMSE  Rsquared      MAE Resample
1  215913.5 0.6962810 115114.7   Fold01
2  196746.4 0.7116750 113607.7   Fold02
3  185814.9 0.7092387 112769.3   Fold03
4  221366.5 0.6869260 117610.8   Fold04
5  212594.6 0.7281173 115938.3   Fold05
6  224361.3 0.6679056 118641.3   Fold06
7  192642.1 0.7163736 116827.1   Fold07
8  228038.6 0.6824760 121432.9   Fold08
9  219823.3 0.6556122 115225.7   Fold09
10 243757.1 0.6773490 121307.8   Fold10
> model_SVM$finalModel
Support Vector Machine object of class "ksvm"
```

```
##SVM Model without Transformation

model_SVM<-train(price~.,data = mydata,trControl = data_ctrl,method ='svmLinear',tuneGrid = psvm_grid)

summary(model_SVM)
model_SVM$resample
model_SVM$finalModel

model_SVM
```

369:1 (Untitled)

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/

```
> model_SVM$finalModel
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 15857

Objective Function Value : -4998.821
Training error : 0.341248
> model_SVM
Support Vector Machines with Linear Kernel

21613 samples
   18 predictor

No pre-processing
```

```
##SVM Model without Transformation

model_SVM<-train(price~.,data = mydata,trControl = data_ctrl,method ='svmLinear',tuneGrid = psvm_grid)

summary(model_SVM)
model_SVM$resample
model_SVM$finalModel

model_SVM


```

369:1 (Untitled)

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/

```
> model_SVM
Support Vector Machines with Linear Kernel

21613 samples
   18 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 19451, 19451, 19452, 19451, 19452, 19452, ...
Resampling results:

  RMSE      Rsquared   MAE
  214105.8  0.6931954  116847.6

Tuning parameter 'C' was held constant at a value of 1
>
```

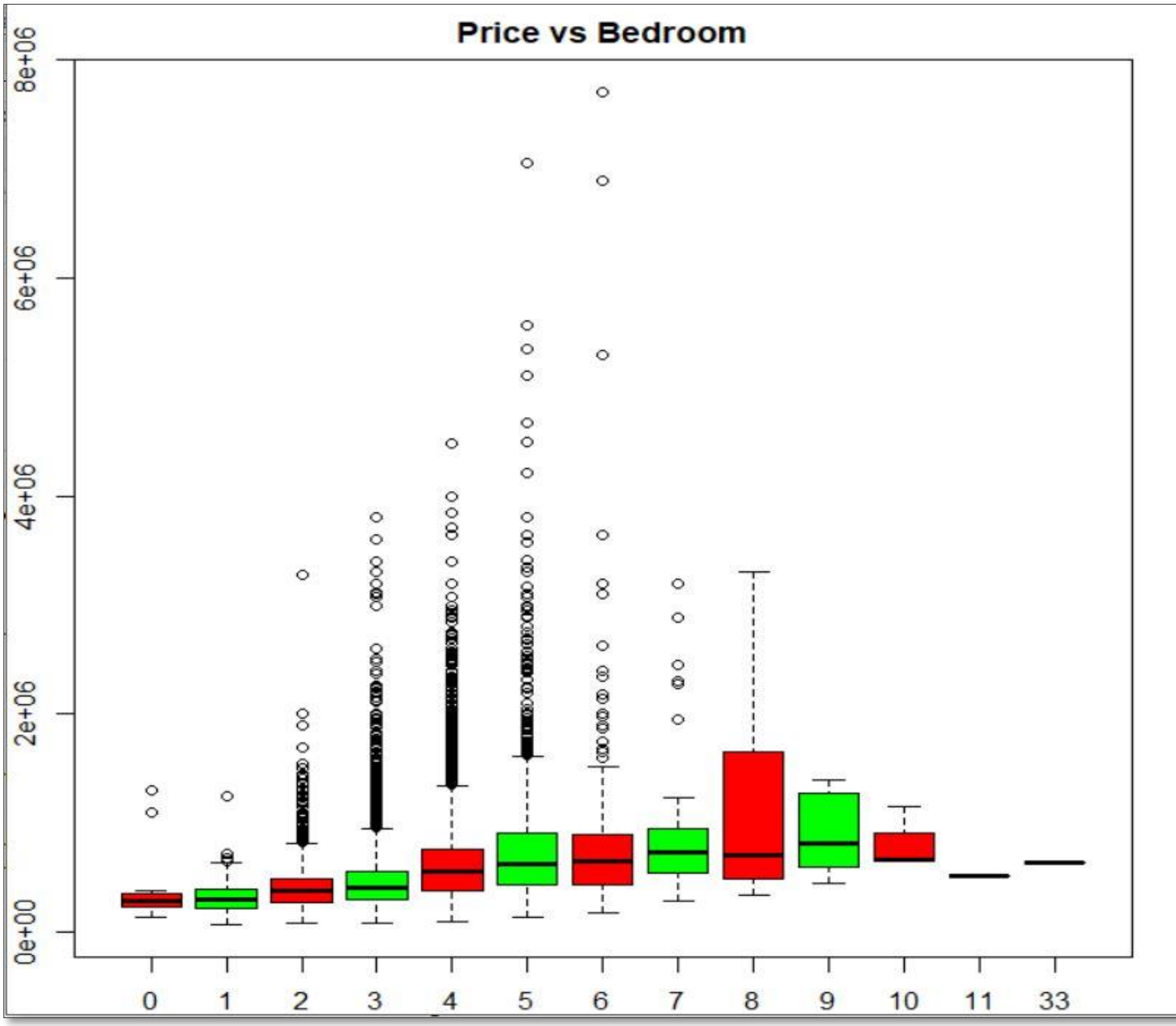
On looking at the RMSE value we see that the value is very high, hence we apply transformation to this model. This is displayed in the next section.

Method 5: ANNOVA

ANNOVA is an extension of linear regression where we use for comparing more than 2 sample means. We have set our Null Hypothesis (H0) as group mean price of all houses with different bedrooms are equal with Alternative Hypothesis (Ha) as group mean price of all houses with different bedrooms are not equal. It means that there is at least 1 group which has a significant difference at 95% confidence level.

```
> bedroom = data_proj$bedrooms
> bedroom <- as.factor(bedroom)
>
> boxplot(hs_price~bedroom, xlab = 'Bedrooms', ylab = 'Price', main = "Price vs Bedroom", col=c("red", "green"))
>
> an1 = lm(hs_price~bedroom)#baseline taken as bedrooms = 0
> summary(an1)

Call:
lm(formula = hs_price ~ bedroom)

Residuals:
    Min      1Q  Median      3Q     Max
-765077 -196388  -63777  103612 6874146

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   410223      96335   4.258 2.07e-05 ***
bedroom1      -92565      99432  -0.931  0.35189
bedroom2       -8835      96561  -0.091  0.92710
bedroom3       56054      96399   0.581  0.56093
bedroom4      225342      96426   2.337  0.01945 *
bedroom5      376651      96725   3.894 9.89e-05 ***
bedroom6      415630      98610   4.215 2.51e-05 ***
bedroom7      541225     111603   4.850 1.25e-06 ***
bedroom8      694854     136238   5.100 3.42e-07 ***
bedroom9      483777     171429   2.822  0.00478 **
bedroom10     409777     222476   1.842  0.06550 .
bedroom11     109777     360452   0.305  0.76071
bedroom33     229777     360452   0.637  0.52383
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 347300 on 21600 degrees of freedom
Multiple R-squared:  0.1065,	Adjusted R-squared:  0.106
F-statistic: 214.6 on 12 and 21600 DF,  p-value: < 2.2e-16
```

Here, for baseline we have taken bedrooms = 0. So, we took a box plot of price vs bedrooms and got the following output.



From the above plot we see that bedroom 8 has larger variance. Hence, in the next section we relevel annova with baseline ref as 8.

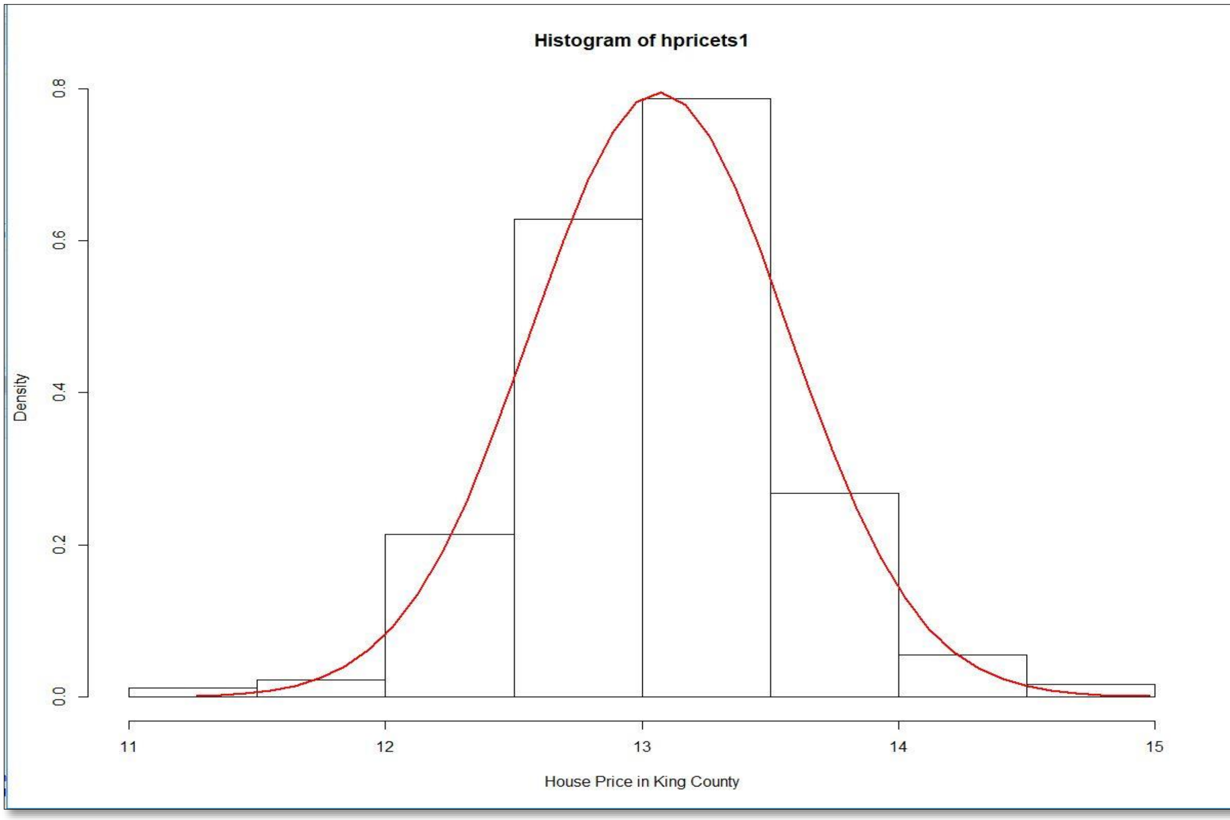
Method 6: Time Series Analysis – Predictions, Evaluations and Forecasting

We are predicting the prices for King county after May 2015. For time series we are considering only two variables viz. y variable is price and x variable is date. In this, we have the following points to be considered:

- The data needs to be stationary and serially correlated. Stationary, means it should a constant mean and variance over time which are the 2 moments of time series.
- In our analysis, we plotted histogram, QQ plots and Ljung – Box test to test if our data is stationary. The initial time series plot wasn't stationary hence, we applied differencing.
- After applying differencing, we found the time plot to be stationary and the data was serially correlated.

So, for our initial analysis we have to following plots:

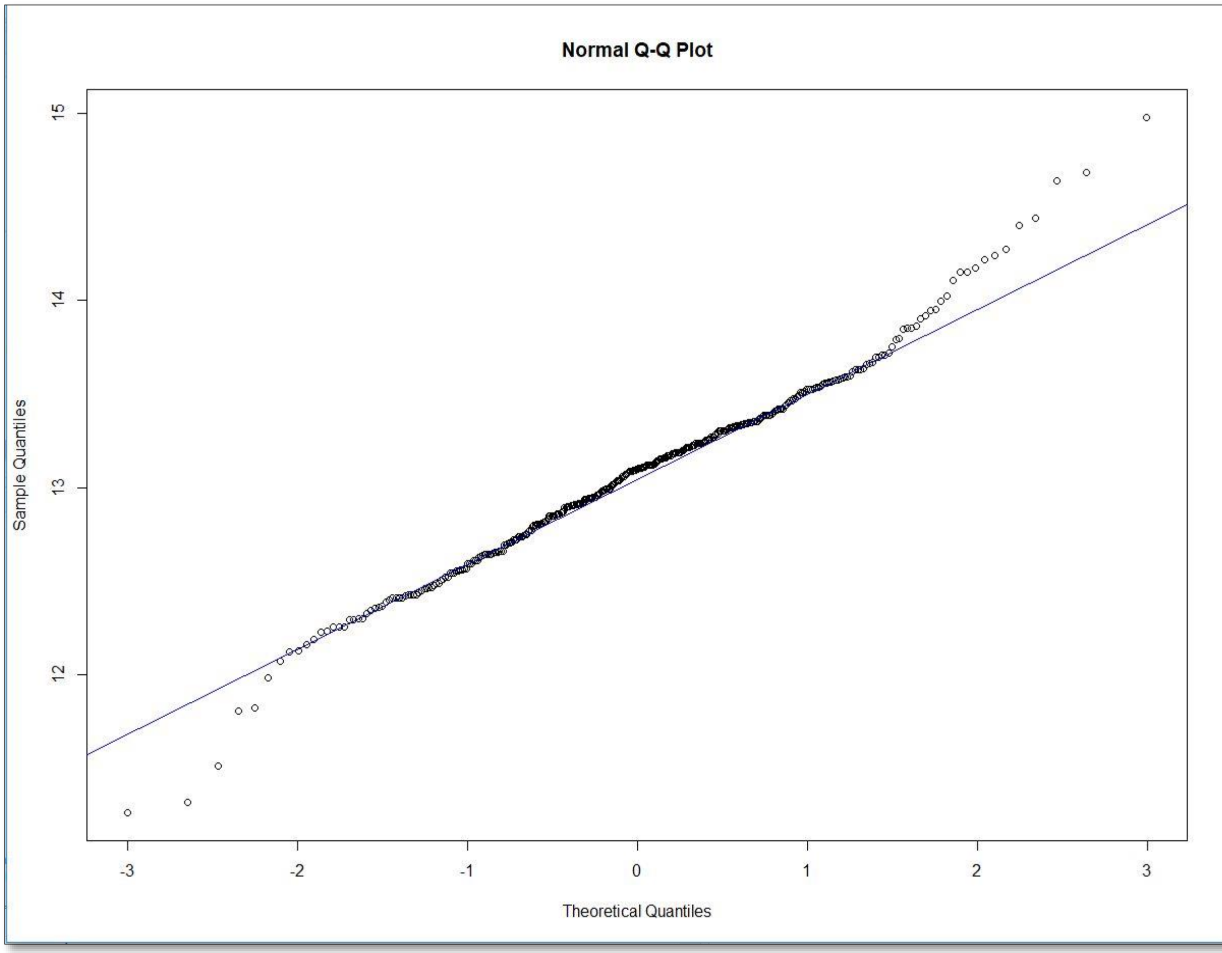
a. Histogram:



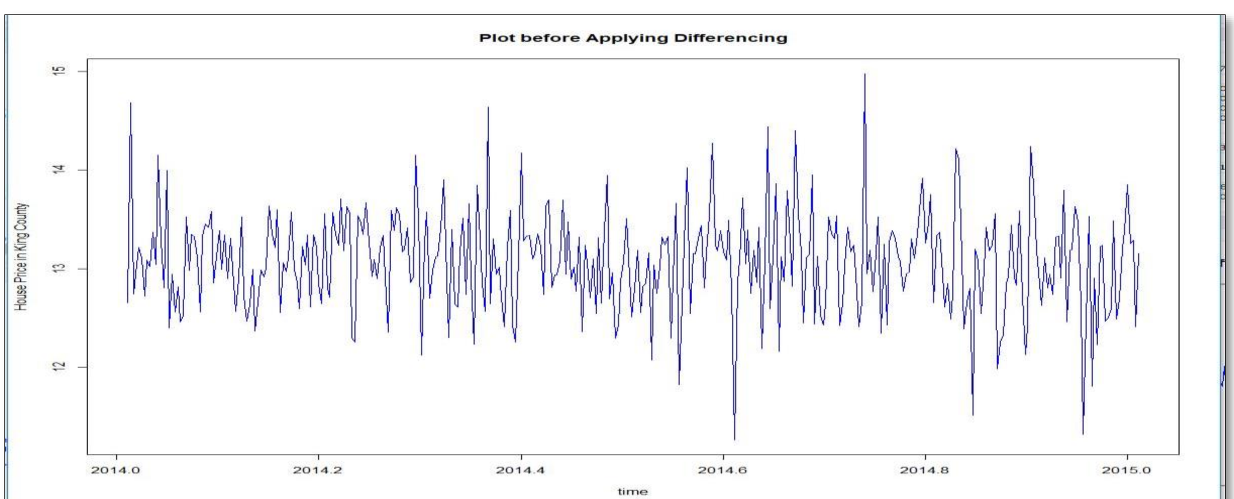
b. Ljung-Box test:

```
> #Using Ljung Box Test to check whether there is any white Gaussian Noise present in the data or not.
> Box.test(rt1,lag=6,type = 'Ljung')

        Box-Ljung test

data:  rt1
X-squared = 12.147, df = 6, p-value = 0.05878

> Box.test(rt1,lag=12,type = 'Ljung')

        Box-Ljung test

data:  rt1
X-squared = 45.356, df = 12, p-value = 8.954e-06

> Box.test(rt1,lag=18,type = 'Ljung')

        Box-Ljung test

data:  rt1
X-squared = 50.905, df = 18, p-value = 5.503e-05

>
```

c. QQ Plot:



d. Time Series Plot before Differencing:



6.2. Results and Findings

Method 1 Hypothesis Testing:

a. One – tail Hypothesis Testing:
   We performed the z-test for one tail hypothesis testing and got the following result

```
> z.test(hs_price, NULL, alternative = "greater", mu = 50000, sigma.x=sd(hs_price), conf.level = 0.95)

        One-sample z-Test

data:  hs_price
z = 196.16, p-value < 2.2e-16
alternative hypothesis: true mean is greater than 50000
95 percent confidence interval:
 536071.9       NA
sample estimates:
mean of x
 540182.2
```

   From the test we reject Null hypothesis and accept the Alternative Hypothesis.

b. Two – tailed Hypothesis testing:
   Based on the parameters calculated we performed the z test for two-tailed hypothesis with confidence interval as 95%, we got the following result:

```
> z.test(price_1floor, price_1.5floor,alternative = "two.sided", mu=0, sigma.x = sd(price_1floor), sigma.y = sd(price_1
.5floor), conf.level = 0.95)

        Two-sample z-Test

data:  price_1floor and price_1.5floor
z = -15.777, p-value < 2.2e-16
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -131338.4 -102312.2
sample estimates:
mean of x mean of y
 442219.6  559044.9
```

   Here too, we reject the Null Hypothesis and accept the Alternative Hypothesis.

Method 2: Multiple Linear Regression

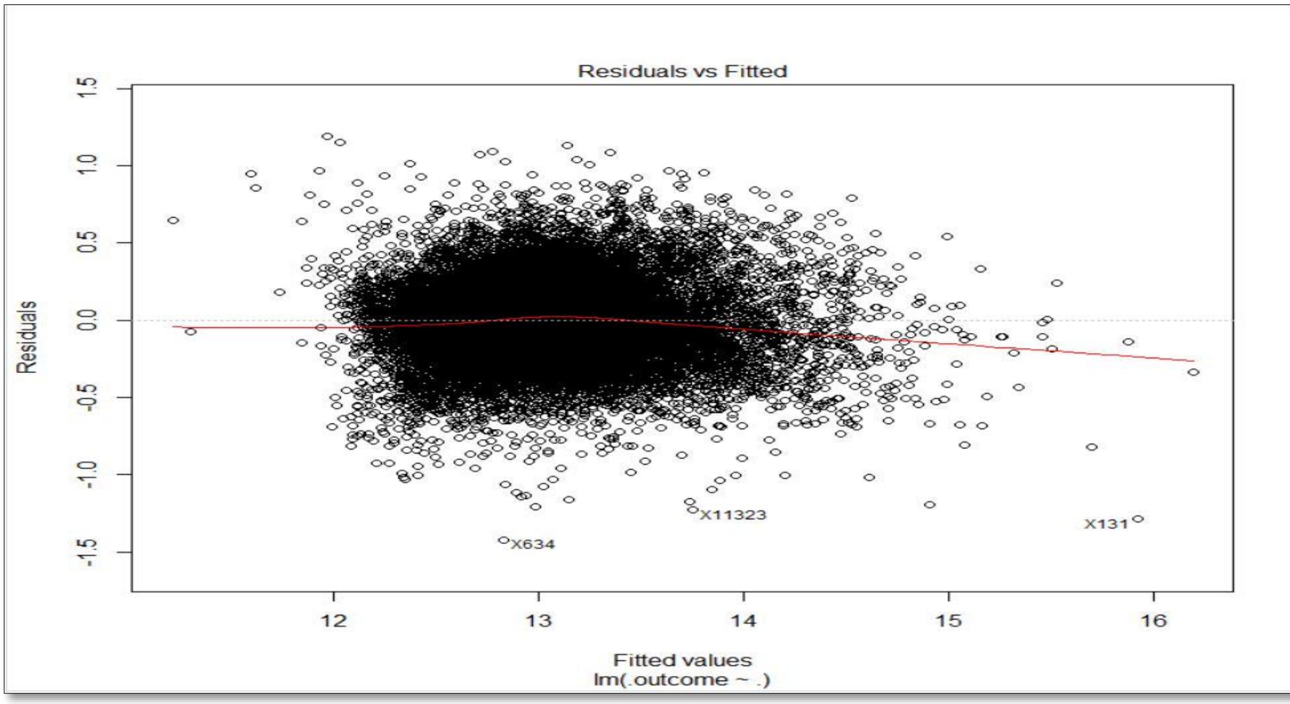
The final model we built post log transformation is as follows:

```
> housemodel3<-train(log(price)~bedrooms+bathrooms+sqft_lot+waterfront+view+condition+grade+sqft_above+yr_built+yr_renovated+zipcod
e+lat+long+sqft_living15+sqft_lot15,
+                    data = data_proj,
+                    trControl = p.data_control,
+                    method = "lm",
+                    na.action = na.pass)
> vif(housemodel3$finalModel)
     bedrooms     bathrooms      sqft_lot    waterfront          view     condition         grade    sqft_above      yr_built
     1.513849      2.705941      2.099755      1.203750      1.401662      1.236913      3.295698      3.576237      2.230851
 yr_renovated       zipcode           lat          long sqft_living15    sqft_lot15
     1.147263      1.647215      1.171558      1.806971      2.774012      2.129372
> housemodel3$resample
        RMSE  Rsquared       MAE Resample
1  0.2613049 0.7482846 0.2060683   Fold01
2  0.2625765 0.7663233 0.2050364   Fold02
3  0.2639399 0.7459120 0.2042005   Fold03
4  0.2622987 0.7628759 0.2022765   Fold04
5  0.2517973 0.7686665 0.1965689   Fold05
6  0.2569471 0.7634161 0.1994015   Fold06
7  0.2568254 0.7560767 0.2013462   Fold07
8  0.2493972 0.7679580 0.1941936   Fold08
9  0.2476825 0.7661721 0.1935304   Fold09
10 0.2609717 0.7702692 0.2040277   Fold10
> housemodel3$finalModel

Call:
lm(formula = .outcome ~ ., data = dat)

Coefficients:
  (Intercept)       bedrooms      bathrooms       sqft_lot     waterfront           view      condition          grade
   -1.967e+01      2.779e-03      1.208e-01      4.899e-07      3.732e-01      7.105e-02      6.563e-02      1.736e-01
   sqft_above       yr_built   yr_renovated        zipcode            lat           long  sqft_living15     sqft_lot15
    9.439e-05     -3.481e-03      3.986e-05     -5.653e-04      1.428e+00     -2.036e-01      1.148e-04     -2.249e-07
```

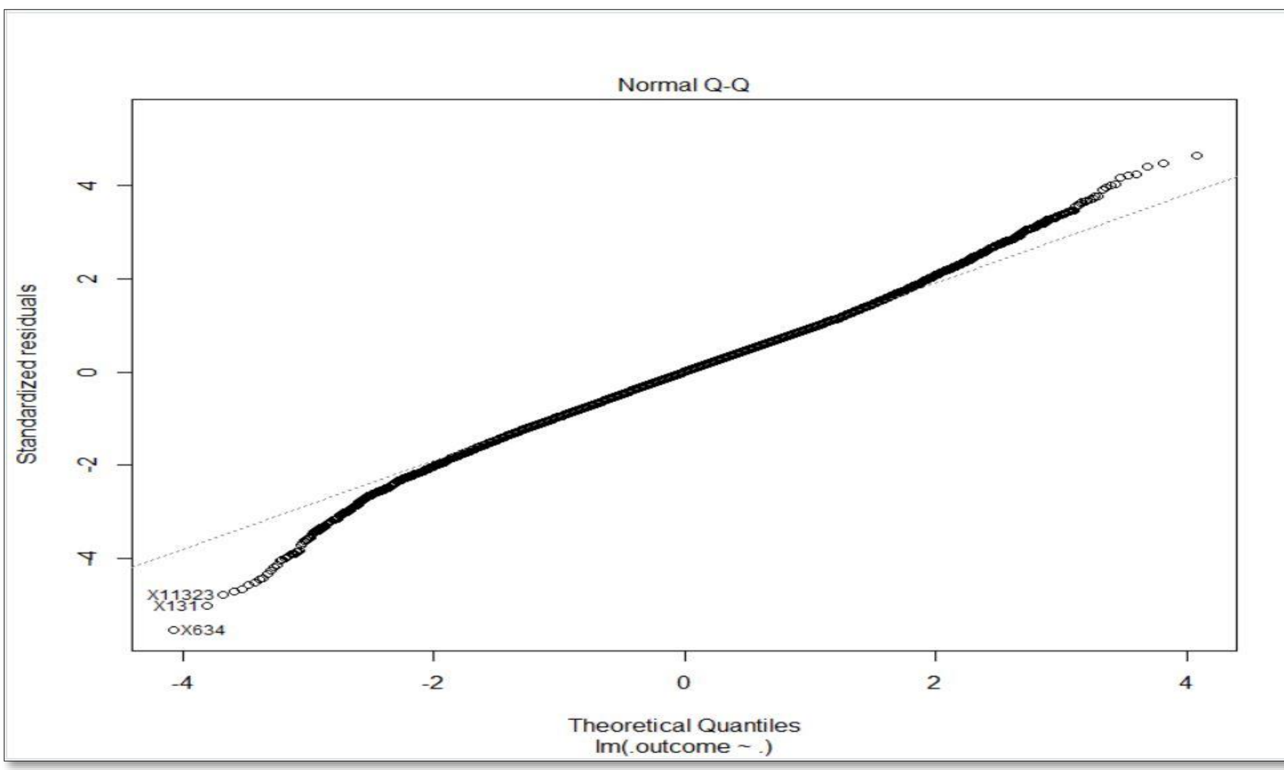
The RMSE and MAE values for the above model is as follows:

```
> housemodel3
Linear Regression

21613 samples
   15 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 19452, 19452, 19451, 19450, 19452, 19452, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2573741  0.7615954  0.200665

Tuning parameter 'intercept' was held constant at a value of TRUE
```

As, we see the RMSE value comes to 0.25 and the MAE values comes to 0.20. Further we will plot the charts for this model to check variance and normality.



From the above plot we see that the residual plot shows constant variance. Now we will check the normality through QQ plots.



Looking at the above plot, we can infer that majority of the points lie on or near the line and that the plot follows Normal distribution.

Hence the final model we get is as follows:

Log(price) = -1.967e+01 + (2.779e-03) * bedrooms + (1.208e-01) * bathrooms + (4.899e-07) * sqft_lot + (3.732e-01) * waterfront + (7.105e-02) * view + (6.563e-02) * condition + (1.736e-01) * grade + (9.439e-05) * sqft_above – (3.481e-03) * yr_built + (3.986e-05) * yr_renovated - (5.653e-04) * zipcode + (1.428) * lat – (-2.036e-01) * long + (1.148e-04) * sqft_living15 - (2.249e-07) * sqft_lot15 + e

Method 3: Random Forest Regression technique

We applied log transformation to the model as follows:

```
s<-createDataPartition(y = mydata$price,p=0.8,list = FALSE)
training <-mydata[s,]
test <-mydata[-s,]
stopifnot(nrow(training) + nrow(test)==nrow(mydata))

x = training[,2:18]
x
y = training[,1]
y

bestmtry<- tuneRF(x,y,stepFactor=1.5,improve=1e-5,ntree=500,doBest=TRUE)

pGrid<-expand.grid(mtry=c(7))

model_RF<-train(price~., data = mydata,trControl = data_ctrl,method ="rf",tuneGrid = pGrid)
```

```
335:1  (Untitled)                                                  R Script

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
> model_RF1<-train(log(price)~.,data = mydata,trControl = data_ctrl,method ="rf",tuneGrid = pGrid)
> summary(model_RF1)
                Length Class      Mode
call                4  -none-     call
type                1  -none-     character
predicted       21613  -none-     numeric
mse               500  -none-     numeric
rsq               500  -none-     numeric
oob.times       21613  -none-     numeric
importance         18  -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               21613  -none-     numeric
test                0  -none-     NULL
inbag               0  -none-     NULL
xNames             18  -none-     character
problemType         1  -none-     character
tuneValue           1  data.frame list
obsLevels           1  -none-     logical
param               0  -none-     list
> model_RF1$resample
        RMSE  Rsquared       MAE Resample
1  0.1720115 0.8936052 0.1216991   Fold01
2  0.1699382 0.8984208 0.1201795   Fold02
3  0.1735220 0.8968393 0.1224534   Fold03
4  0.1782672 0.8852865 0.1241692   Fold04
5  0.1797070 0.8829947 0.1256274   Fold05
6  0.1760572 0.8907433 0.1224249   Fold06
7  0.1716659 0.8900386 0.1196982   Fold07
8  0.1676034 0.8978973 0.1200113   Fold08
9  0.1751527 0.8895405 0.1221988   Fold09
10 0.1708691 0.8959251 0.1194928   Fold10
> model_RF1$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
```

The RMSE and MAE obtained for this model are:

```
model_RF1<-train(log(price)~.,data = mydata,trControl = data_ctrl,method ="rf",tuneGrid = pGrid)

summary(model_RF1)
model_RF1$resample
model_RF1$finalModel

model_RF1

335:1  (Untitled) :
```

```
Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
tuneValue                   1   data.frame list
obsLevels                   1   -none-     logical
param                       0   -none-     list
> model_RF1$resample
        RMSE  Rsquared        MAE Resample
1   0.1720115 0.8936052 0.1216991   Fold01
2   0.1699382 0.8984208 0.1201795   Fold02
3   0.1735220 0.8968393 0.1224534   Fold03
4   0.1782672 0.8852865 0.1241692   Fold04
5   0.1797070 0.8829947 0.1256274   Fold05
6   0.1760572 0.8907433 0.1224249   Fold06
7   0.1716659 0.8900386 0.1196982   Fold07
8   0.1676034 0.8978973 0.1200113   Fold08
9   0.1751527 0.8895405 0.1221988   Fold09
10 0.1708691 0.8959251 0.1194928   Fold10
> model_RF1$finalModel

Call:
 randomForest(x = x, y = y, mtry = param$mtry)
               Type of random forest: regression
                     Number of trees: 500
No. of variables tried at each split: 7

          Mean of squared residuals: 0.03012683
                    % Var explained: 89.14
> model_RF1
Random Forest

21613 samples
   18 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 19452, 19452, 19452, 19452, 19451, 19451, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.1734794  0.8921291  0.1217955

Tuning parameter 'mtry' was held constant at a value of 7
```

We get the values as 0.173 for RMSE and 0.121 for MAE.

Method 4: Support Vector Machine Linear Regression

We applied log transformations to SVM model to get the RMSE value down. After applying log, we got the below results:

```
## Cost Factor for our Model c is 1.
psvm_grid<-expand.grid(C = c(1))

model_SVM1<-train(log(price)~.,data = mydata,trControl = data_ctrl,method ='svmLinear',tuneGrid = psvm_grid)

summary(model_SVM1)
model_SVM1$resample
model_SVM1$finalModel

model_SVM1

358:1  (Untitled) :
```

```
Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
> model_SVM1<-train(log(price)~.,data = mydata,trControl = data_ctrl,method ='svmLinear',tuneGrid = psvm_grid)
> summary(model_SVM1)
Length  Class   Mode
     1   ksvm     S4
> model_SVM1$resample
        RMSE  Rsquared       MAE Resample
1  0.2513320 0.7736028 0.1954004   Fold01
2  0.2551432 0.7726741 0.1935907   Fold02
3  0.2565254 0.7738117 0.1973950   Fold03
4  0.2587576 0.7480273 0.2014592   Fold04
5  0.2550295 0.7752744 0.1943625   Fold05
6  0.2538699 0.7642409 0.1956683   Fold06
7  0.2548996 0.7668182 0.1948026   Fold07
8  0.2467071 0.7683905 0.1909269   Fold08
9  0.2536333 0.7693728 0.1942724   Fold09
10 0.2500404 0.7709442 0.1914909   Fold10
> model_SVM1$finalModel
```

```
353  summary(model_SVM1)
354  model_SVM1$resample
355  model_SVM1$finalModel
356
357  model_SVM1
358
358:1  (Untitled) ÷

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
> model_SVM1$finalModel
Support Vector Machine object of class "ksvm"

SV type: eps-svr  (regression)
 parameter : epsilon = 0.1  cost C = 1

Linear (vanilla) kernel function.

Number of Support Vectors : 17645

Objective Function Value : -6028.588
Training error : 0.231386
> model_SVM1
Support Vector Machines with Linear Kernel

21613 samples
   18 predictor
```

```
357  model_SVM1
358
358:1  (Untitled) ÷

Console C:/Users/Vineet Sampat/Desktop/Data Analytics/
> model_SVM1
Support Vector Machines with Linear Kernel

21613 samples
   18 predictor

No pre-processing
Resampling: Cross-Validated (10 fold)
Summary of sample sizes: 19453, 19452, 19453, 19453, 19452, 19451, ...
Resampling results:

  RMSE       Rsquared   MAE
  0.2535938  0.7683157  0.1949369

Tuning parameter 'C' was held constant at a value of 1
>
```

The RMSE value is 0.25 with the MAE as 0.19.

Selecting Best Fit Model for Regression:

Sr.No	Model Name	RMSE	R-Squared	MAE
1	Multiple Linear Regression	0.2573741	0.7615954	0.200665
2	**Random Forest**	**0.1734794**	**0.8921291**	**0.1217955**
3	SVM Linear Regression	0.2535938	0.7683157	0.1949369

From the above table it is clear that the **Random Forest** Regression technique is the Best fit for our dataset to predict price of house. The RMSE value for Random Forest technique is 0.173 with the accuracy value as 89%.

Thus, with low RMSE and MAE and a high accuracy value, Random Forest is the best Regression technique in predicting price of house.

Method 5: ANNOVA

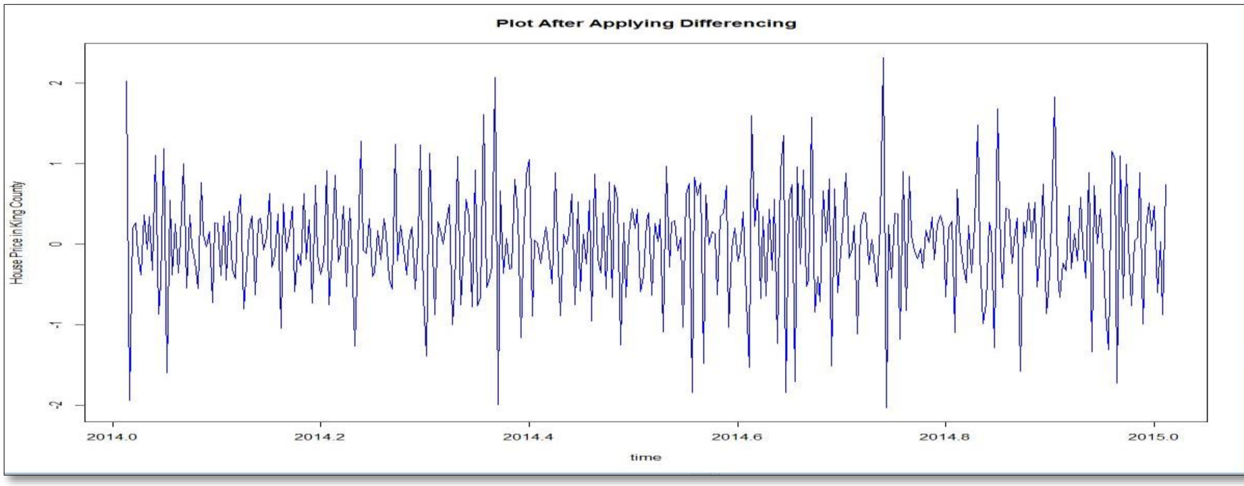
```
> #releveling the bedroom
> bedroom = relevel(bedroom, ref=8) #ref value taken as 8 as per box plot interpretation
> an2 = lm(hs_price~bedroom)
> summary(an2)

Call:
lm(formula = hs_price ~ bedroom)

Residuals:
    Min      1Q  Median      3Q     Max 
-765077 -196388  -63777  103612 6874146 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)   951448      56346  16.886  < 2e-16 ***
bedroom0     -541225     111603  -4.850 1.25e-06 ***
bedroom1     -633790      61491 -10.307  < 2e-16 ***
bedroom2     -550060      56733  -9.696  < 2e-16 ***
bedroom3     -485171      56455  -8.594  < 2e-16 ***
bedroom4     -315883      56501  -5.591 2.29e-08 ***
bedroom5     -164574      57011  -2.887   0.0039 ** 
bedroom6     -125594      60153  -2.088   0.0368 *  
bedroom8      153629     111603   1.377   0.1687    
bedroom9      -57448     152586  -0.376   0.7066    
bedroom10    -131448     208302  -0.631   0.5280    
bedroom11    -431448     351881  -1.226   0.2202    
bedroom33    -311448     351881  -0.885   0.3761    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 347300 on 21600 degrees of freedom
Multiple R-squared:  0.1065,	Adjusted R-squared:  0.106 
F-statistic: 214.6 on 12 and 21600 DF,  p-value: < 2.2e-16
```

From the above R output, we see that the bedrooms 0-6 have significant difference than rest of the bedroom prices with p value < 0.05.

Method 6: Time Series – After Differencing

- Post differencing, we see that the time series plot is stationary and is having constant mean and variance over time.
- Time series data is also correlated. This is confirmed using Ljung-Box test at 95% confidence interval with p-value < 0.05.
- The QQ plot, most of the points lie on or near the line that indicates Normality check for the time series data.
- Also, the histogram density shows a normal distribution.
- Finally, we have performed Jarque-Bera test of normality at 95% confidence interval.
- All the above conditions are satisfied, and we plot ACF and PACF plot to get q and p values respectively.

a. Time Series plot post differencing:

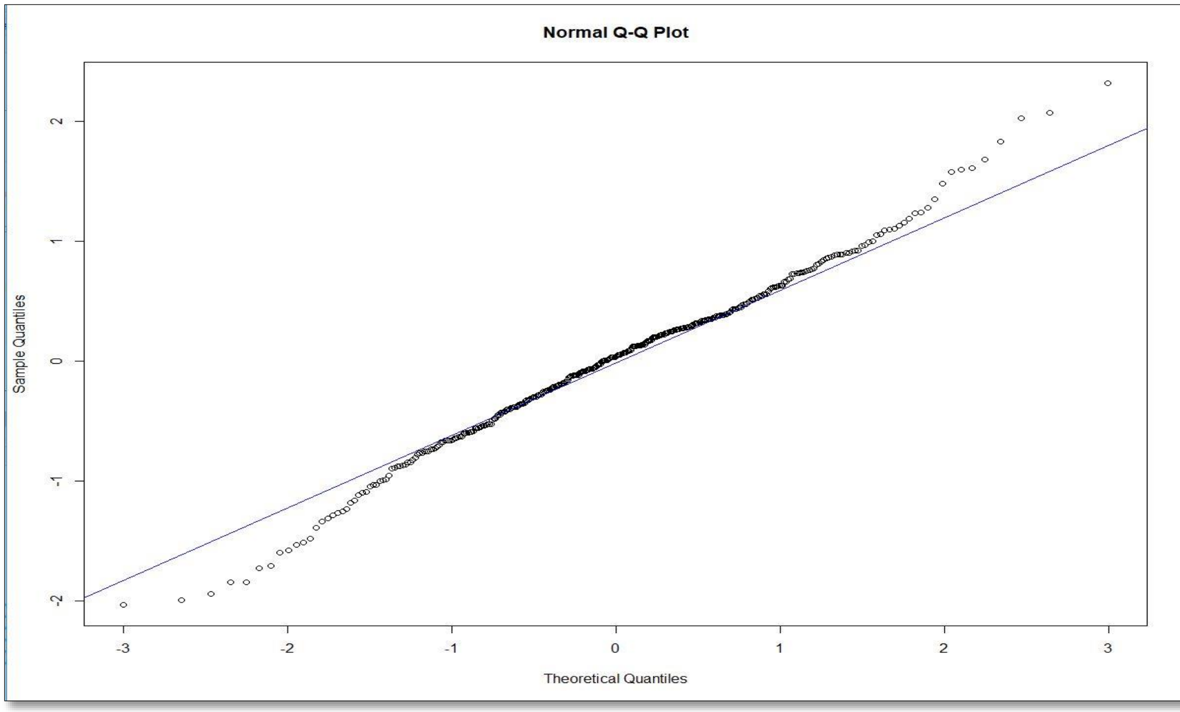


b. Ljung – Box Test post differencing:

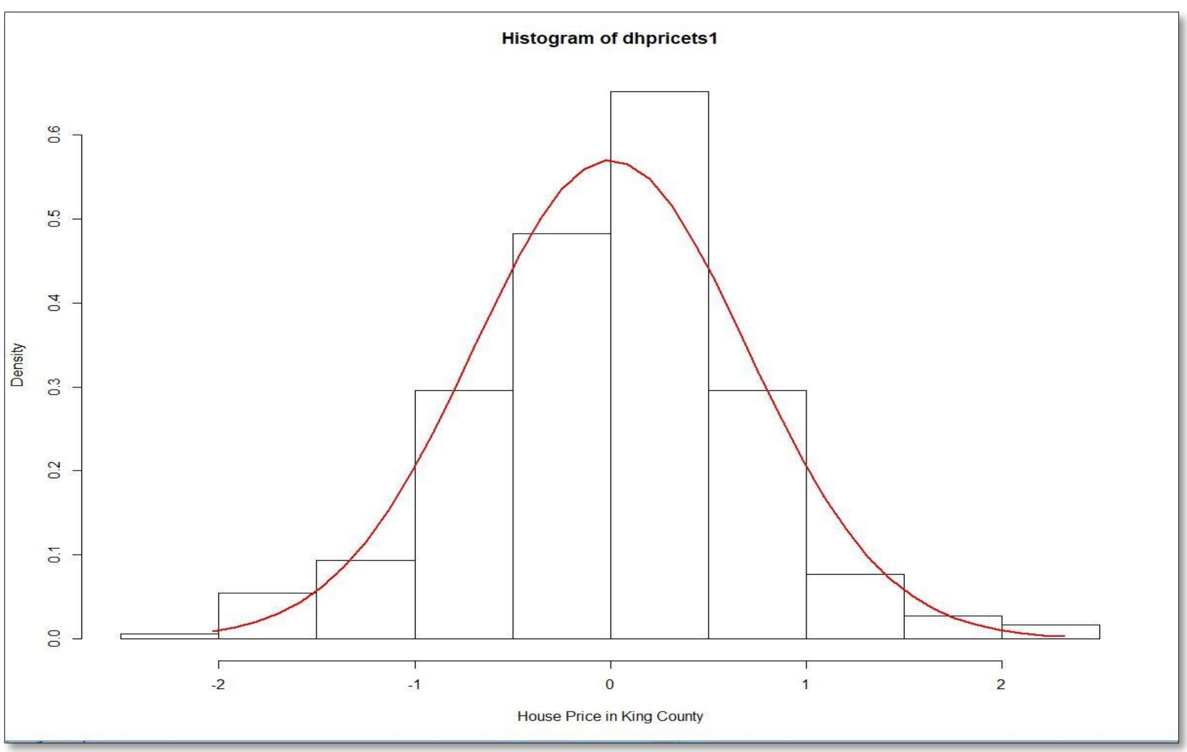
```
> #Using Ljung Box Test to check whether there is any White Gaussian Noise present in the data or not.
> Box.test(coredata(dhpricets1),lag=6,type = 'Ljung')

        Box-Ljung test

data:  coredata(dhpricets1)
X-squared = 76.832, df = 6, p-value = 1.61e-14

> Box.test(coredata(dhpricets1),lag=12,type = 'Ljung')

        Box-Ljung test

data:  coredata(dhpricets1)
X-squared = 99.421, df = 12, p-value = 7.772e-16

> Box.test(coredata(dhpricets1),lag=18,type = 'Ljung')

        Box-Ljung test

data:  coredata(dhpricets1)
X-squared = 111.34, df = 18, p-value = 1.776e-15

>
```

c. QQ plot post differencing:

```
> #Potting QQ plot to check if the data is normally distributed or not.
> qqnorm(dhpricets1)
> qqline(coredata(dhpricets1), col = "blue")
>
```



d. Histogram post differencing:

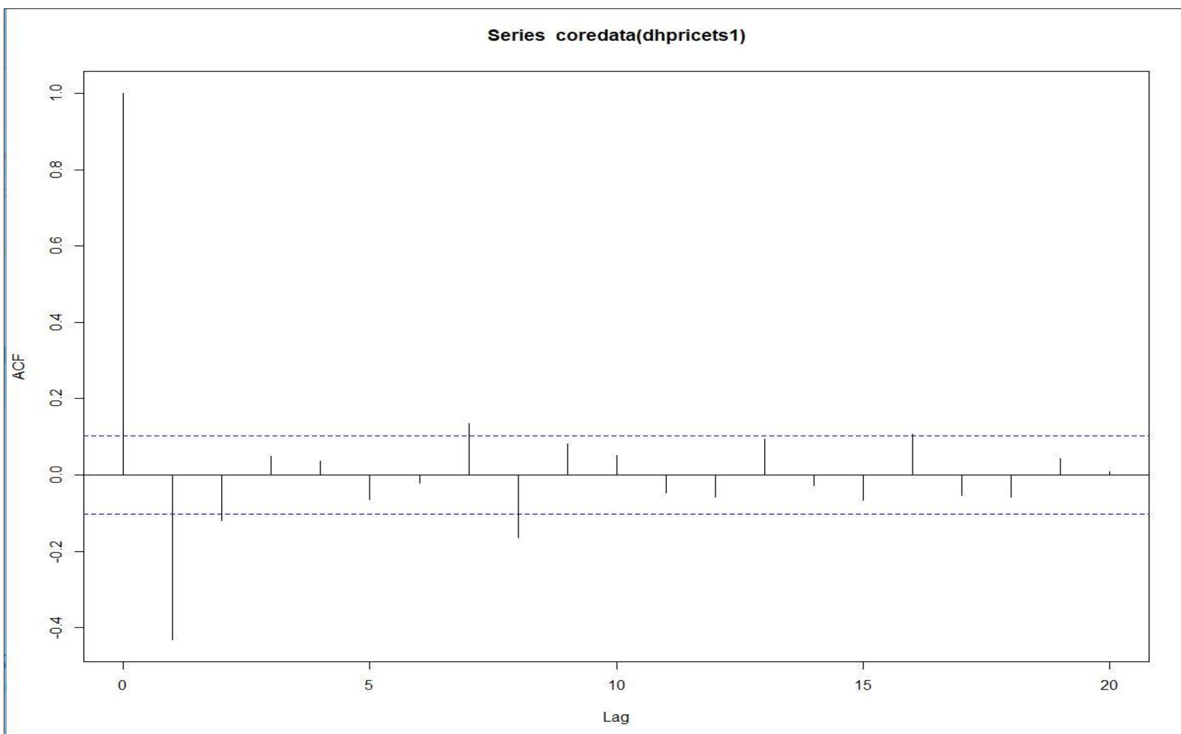


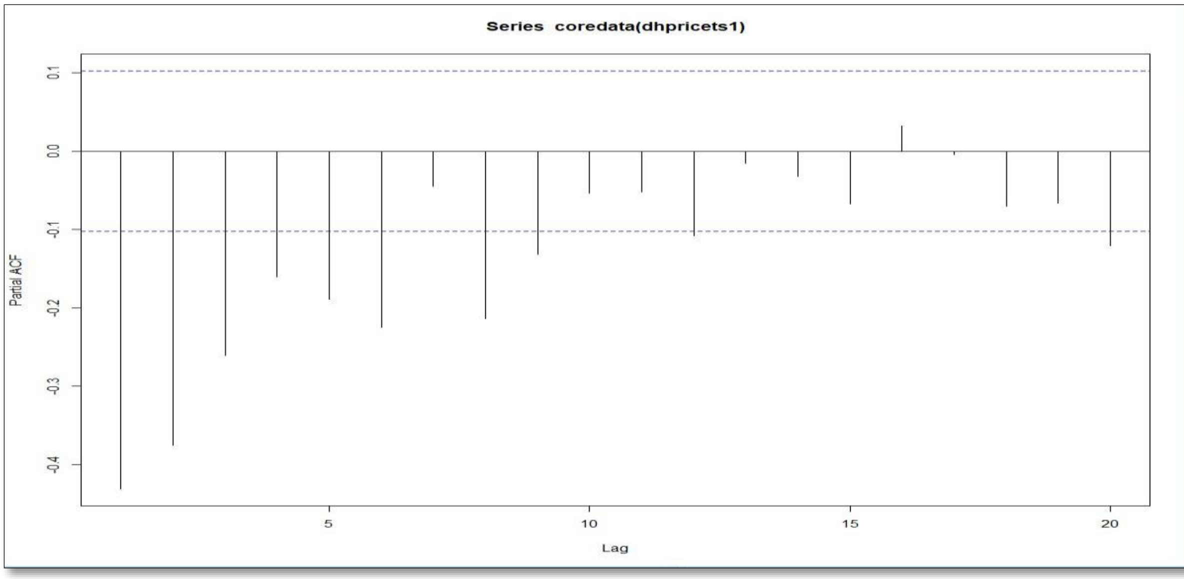
e. Jarque-Bera Test post differencing:

```
> #Checking Normality Test for the House Price.
> jarque.bera.test(dhpricets1)

        Jarque Bera Test

data:  dhpricets1
X-squared = 6.8688, df = 2, p-value = 0.03224

> |
```

f. ACF and PACF plot post differencing:

```
> acf(coredata(dhpricets1), plot=F,lag.max=20,na.action=na.pass)

Autocorrelations of series 'coredata(dhpricets1)', by lag

     0      1      2      3      4      5      6      7      8      9     10     11     12     13     14     15     16 
 1.000 -0.431 -0.120  0.050  0.037 -0.064 -0.021  0.135 -0.165  0.082  0.052 -0.046 -0.057  0.093 -0.027 -0.067  0.106 
    17     18     19     20 
-0.053 -0.057  0.043  0.009 
> |
```





Method 7: AR Model (1,0,0)

We built the AR model (p,d,q), In our case we get the p value as 1 from the PACF plot and built the AR model using arima function.

```
> #Building AR Model:
> m2_AR=arima(dhpricets1, c(1,0,0))
> m2_AR

Call:
arima(x = dhpricets1, order = c(1, 0, 0))

Coefficients:
          ar1  intercept
      -0.4416    -0.0010
s.e.   0.0476     0.0228

sigma^2 estimated as 0.3948:  log likelihood = -348.42,  aic = 702.83
```

From the above mode, AR has aic value as 702.83.

Predictions for AR model: We have predicted 10 values after May 2015.

```
> ##Forecasting for Future House Prices:
> predict(m2_AR,n.ahead=10,se.fit=T)
$pred
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] -0.3276199806  0.1432793062 -0.0646493059  0.0271629116 -0.0133773642  0.0045234556 -0.0033807666  0.0001093936
 [9] -0.0014317091 -0.0007512253

$se
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] 0.6283429 0.6868717 0.6977114 0.6998053 0.7002128 0.7002923 0.7003078 0.7003108 0.7003114 0.7003115

>
```

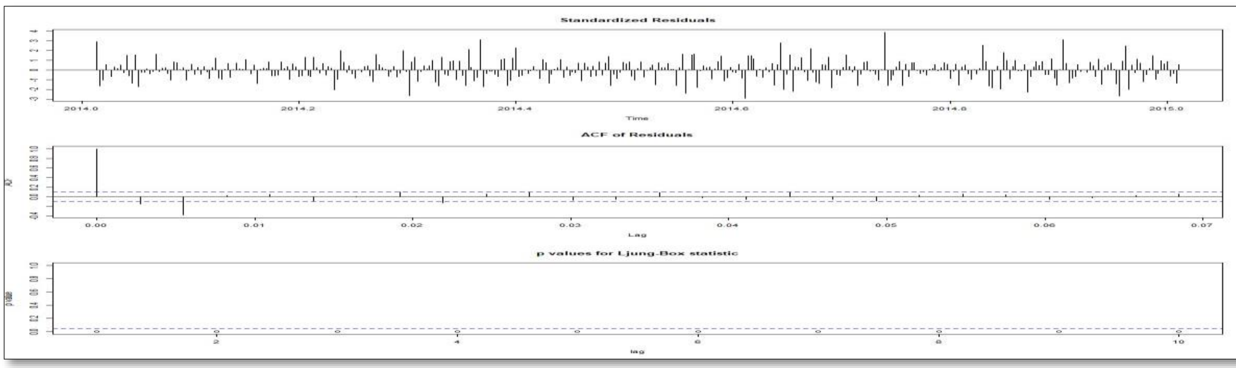
We had withheld last 100 records from our dataset as a testing data for AR model.

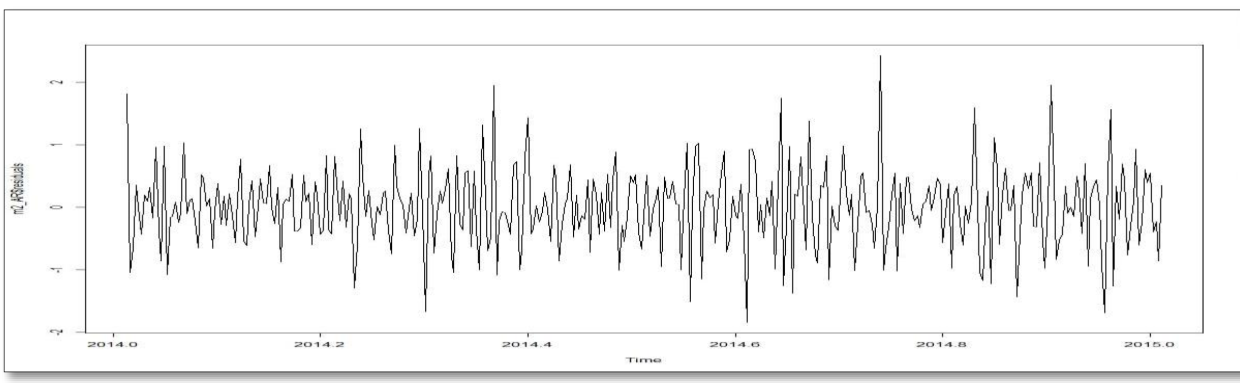
Model Evaluation: RMSE and MAE

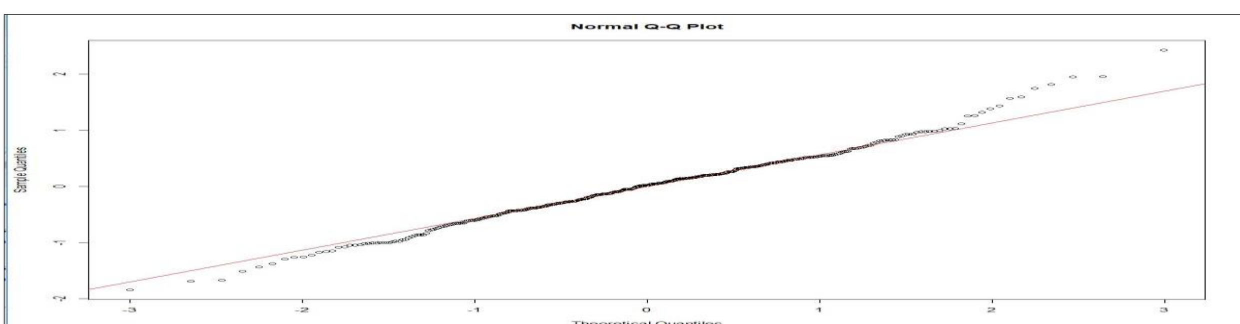
```
> accuracy(forecast(m2_AR),rate2)
                      ME      RMSE       MAE      MPE     MAPE       MASE      ACF1
Training set  0.001883693 0.6283429  0.483755 100.4007 189.5987  0.5335555 -0.155654
Test set     13.063150860 13.0748024 13.063151 100.0254 100.0254 14.4079452        NA
```

Residual Analysis for AR model:

The residue for the AR model are not white noise independent as per the Ljung-Box test.







Method 8: MA Model(0,0,1)

We built the MA model (p,d,q), In our case we get the q value as 1 from the ACF plot and built the MA model using arima function.

```
> #Building MA Model:
> m2_MA=arima(dhpricets1,order = c(0,0,1))
> m2_MA

Call:
arima(x = dhpricets1, order = c(0, 0, 1))

Coefficients:
          ma1  intercept
      -1.0000     -3e-04
s.e.   0.0081      3e-04

sigma^2 estimated as 0.2508:  log likelihood = -268.41,  aic = 542.81
> |
```

From the above mode, MA has aic value as 542.81.

Predictions for MA model: We have predicted 10 values after May 2015.

```
> predict(m2_MA,n.ahead=10,se.fit=T)
$pred
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] -0.1460625177 -0.0003451486 -0.0003451486 -0.0003451486 -0.0003451486 -0.0003451486 -0.0003451486 -0.0003451486
 [9] -0.0003451486 -0.0003451486

$se
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] 0.5014270 0.7081579 0.7081579 0.7081579 0.7081579 0.7081579 0.7081579 0.7081579 0.7081579 0.7081579

> |
```

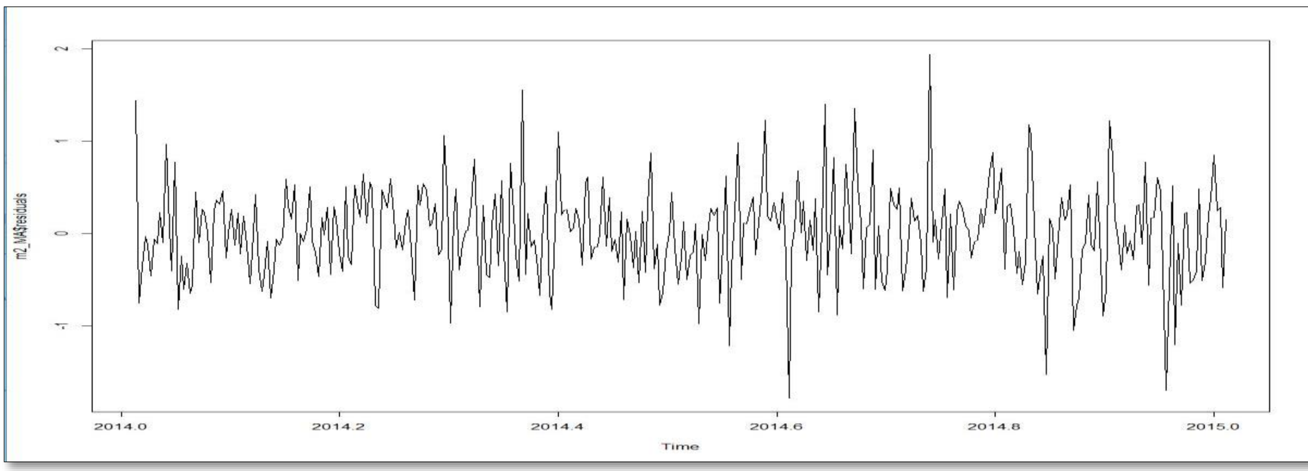
We had withheld last 100 records from our dataset as a testing data for AR model.

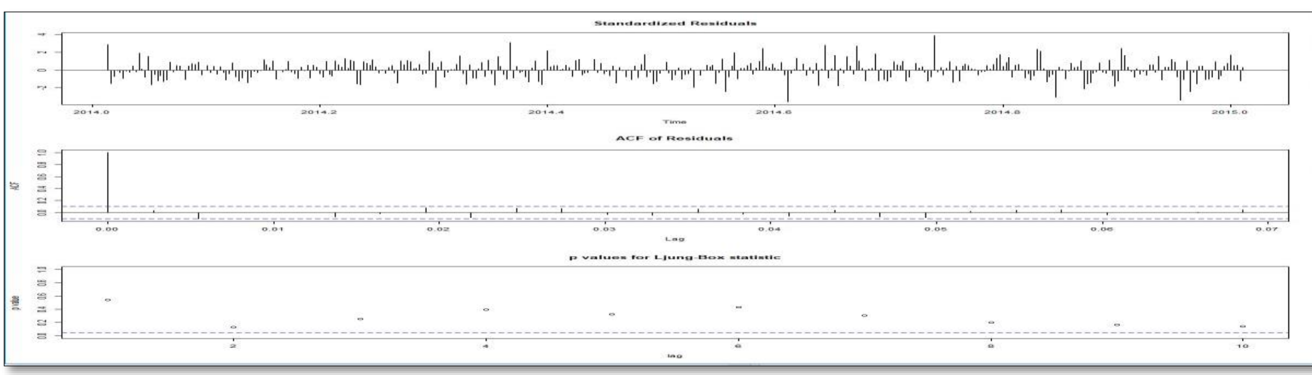
Model Evaluation: RMSE and MAE

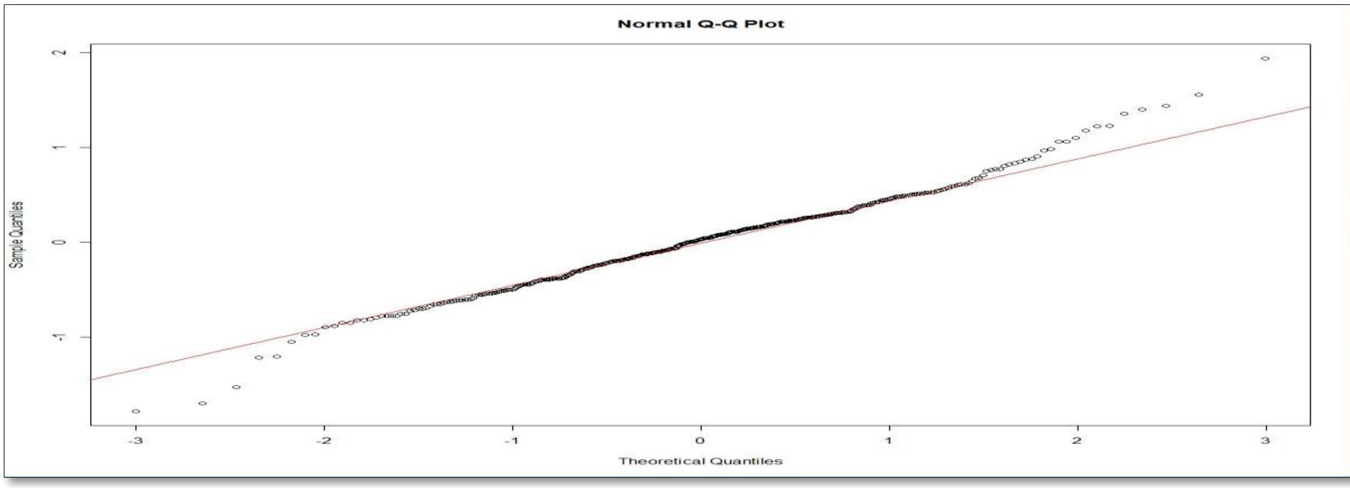
```
> accuracy(forecast(m2_MA),rate2)
                      ME       RMSE        MAE       MPE     MAPE       MASE       ACF1
Training set 0.002812124  0.5007556  0.3859924  49.91099 189.8945  0.4257286 0.03177383
Test set     13.061727496 13.0734098 13.0617275 100.01397 100.0140 14.4063753        NA
```

Residual Analysis for MA model:

```
> Box.test(m2_MA$residuals,lag=6,type='Ljung')

        Box-Ljung test

data:  m2_MA$residuals
X-squared = 5.9279, df = 6, p-value = 0.4313

> Box.test(m2_MA$residuals,lag=12,type='Ljung')

        Box-Ljung test

data:  m2_MA$residuals
X-squared = 15.702, df = 12, p-value = 0.2052

> Box.test(m2_MA$residuals,lag=18,type='Ljung')

        Box-Ljung test

data:  m2_MA$residuals
X-squared = 23.494, df = 18, p-value = 0.1723

>
```







The Residuals for MA model have p value > 0.05 that indicates residuals are white noise independent. This satisfies the residual analysis assumption.

Method 9: ARMA Model

We built the ARMA model (p,d,q), In our case we get the p value as 1 and q value as 2 from the ACF plot and built the ARMA model using arima function.

```
> m2_ARMA=arima(dhpricets1, order=c(1,0,2),method='ML',include.mean=T)
> m2_ARMA

Call:
arima(x = dhpricets1, order = c(1, 0, 2), include.mean = T, method = "ML")

Coefficients:
          ar1      ma1      ma2  intercept
      -0.8682  -0.0812  -0.9188     -3e-04
s.e.   0.0873   0.0692   0.0691      3e-04

sigma^2 estimated as 0.2483:  log likelihood = -266.65,  aic = 543.29
> 
```

From the above mode, MA has aic value as 543.29.

Predictions for MA model: We have predicted 10 values after May 2015.

```
> predict(m2_ARMA,n.ahead=10,se.fit=T)
$pred
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] -0.09298936 -0.09830247  0.08471195 -0.07418691  0.06377406 -0.05600798  0.04799055 -0.04230424  0.03609253
[10] -0.03197400

$se
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] 0.4990043 0.6871559 0.6887654 0.6899761 0.6908875 0.6915736 0.6920904 0.6924798 0.6927731 0.6929942

> 
```

The EACF matrix is as follows:

```
> #Build Model for ARMA using EACF Matrix
> source("EACF.R")
> EACF(dhpricets1)
[1] "EACF table"
      [,1]   [,2]   [,3]   [,4]   [,5]   [,6]
[1,] -0.43 -0.120  0.050  0.037 -0.064 -0.021
[2,] -0.52 -0.373  0.064  0.047 -0.053 -0.014
[3,] -0.49  0.135 -0.418 -0.058  0.023 -0.030
[4,] -0.46 -0.229 -0.290 -0.216 -0.022 -0.014
[5,] -0.50 -0.022 -0.268 -0.273  0.116 -0.016
[6,] -0.50  0.015 -0.385 -0.230 -0.081 -0.062
[1] " "
[1] "Simplified EACF: 2 denotes significance"
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    2    2    0    0    0    0
[2,]    2    2    0    0    0    0
[3,]    2    2    2    0    0    0
[4,]    2    2    2    2    0    0
[5,]    2    0    2    2    2    0
[6,]    2    0    2    2    0    0
>
```

Model Evaluation: RMSE and MAE

```
> accuracy(forecast(m2_ARMA),rate2)
                      ME       RMSE        MAE      MPE     MAPE       MASE        ACF1
Training set 0.003331647  0.4983241  0.3854077  46.9682 172.5924  0.4250838 -0.00950211
Test set    13.061717155 13.0735759 13.0617172 100.0126 100.0126 14.4063639          NA
```

Method 10: ARIMA Model

Similarly, we have built the ARIMA model with below screenshots

ARIMA Model:

```
> library(forecast)
> m2_ARIMA=auto.arima(dhpricets1,max.P=12,max.Q=12,ic="aic")
> m2_ARIMA
Series: dhpricets1
ARIMA(0,0,0) with zero mean

sigma^2 estimated as 0.488:  log likelihood=-386.97
AIC=775.95   AICc=775.96   BIC=779.85
>
```

Predictions based on ARIMA model:

```
> predict(m2_ARIMA,n.ahead=10,se.fit=T)
$pred
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] 0 0 0 0 0 0 0 0 0 0

$se
Time Series:
Start = c(2015, 6)
End = c(2015, 15)
Frequency = 365
 [1] 0.6985604 0.6985604 0.6985604 0.6985604 0.6985604 0.6985604 0.6985604 0.6985604 0.6985604 0.6985604
>
```

Accuracy ARIMA:

```
> accuracy(forecast(m2_ARIMA),rate2)
                      ME       RMSE        MAE MPE MAPE       MASE       ACF1
Training set 0.001364281  0.6985604  0.5380966 100  100  0.5934913 -0.4311825
Test set    13.059925173 13.0716228 13.0599252 100  100 14.4043874         NA
>
```

So, based on the models we built in time series we find that the **MA and ARIMA models are best** as per their RMSE and MAE values. However, the accuracy of MA model is best with aic value 542.81.

7. Conclusions and Future Work

7.1. Conclusions

Using Time series analysis, we can get the predicted prices for the future. However, from all the models we built using Regression Techniques and Time series, the Random Forest is the best working technique for our dataset.

The variables involved in building the Random Forest model are significant and influence the house prices greatly.

7.2. Limitations

The dataset is from May 2014 – 2015, implying that the data is old and lacks recent values and factors. Hence, the dataset might not be that apt in predict house prices and needs an update.

Moreover, in current times there are other factors also that influence the price of house like connectivity, public transport, crime rate etc. These variables will definitely assist in accurately predicting the price for future.

7.3. Potential Improvements or Future Work

We can use other models like CART, GLM etc and compare those models with existing models and derive to conclusion which model will be the best in predicting prices in King County area in Seattle.