STATISTICAL ANALYSIS OF HOUSING PRICES

Statistical Data Analysis Project

Mrudav Mehta

BS21DON042

S P Jain School of Global Management

TABLE OF CONTENTS

1. ACKNOWLEDGEMENT

It gives me immense pleasure to present the 'Project on Statistical Data Analysis', titled Statistical Analysis of Housing Prices. I would like to express my gratitude towards my professor, Dr. Suchismita Das, under whose guidance and constant supervision the project has been completed. The instructions given by her have been a major contribution towards the completion of my project.

2. SCOPE & OBJECTIVE OF THE PROJECT

This project aims to:

1. Understand the data in hand
2. Analyse the data completely and make suitable inferences
3. Build a regression model to accurately predict the housing prices based on the available data.

3. INTRODUCTION

- Consumer spending is inextricably related to the property market. When property values rise, homeowners benefit financially and gain confidence. Some people will borrow more against their home's worth to buy products and services, repair their property, replenish their pension, or pay off existing debt. When property values fall, homeowners risk having their home worth less than their mortgage balance. As a result, people are more prone to cut back on spending and put off making personal investments.
- Thus, real estate is a key generator of economic growth on a small scale as well as a large scale. It also becomes extremely vital to make accurate predictions in this field.
- A real estate business in the United States that maintains properties near a ski resort wants to enhance its property pricing procedures. The data was readily available on a variety of factors, including property size, location, the age of the house, and some other factors.
- The main goal of this entire exercise is to predict the housing prices of any property, based on the pre-determined factors.

4. DATA DESCRIPTION

- The dataset of Housing Prices has been retrieved from JMP User Community's Sample Data Library,[1] and has been exported to a CSV file as well.
- Attributes of the dataset:
    i. Price – The price column indicates the selling price of a particular property in 1000 US Dollars (example: Price of 373 indicates that the selling price of that house was 373,000 US Dollars). As mentioned earlier, 'Price' is the dependent variable, and the model aims to predict the price. Although based on the data 'Price' looks discrete, but it is actually a continuous variable.
    ii. Beds – 'Beds' indicates the number of bedrooms in a particular house. It is an independent factor and a discrete variable.
    iii. Baths – 'Baths' indicates the number of bathrooms in a particular house. A full bathroom contains a shower, a sink, and a toilet. A half-bathroom (values in the 'Bath' column with .5 attached) usually consists of a toilet and a sink, whereas a quarter-bathroom (values in the 'Bath' column with .75 attached) usually has any one of the aforementioned components. Hence, we consider 'Bath' to be a continuous variable, and it is also independent.

iv. Square Feet – 'Square Feet' is the total space in a house that can be used as a living space. The square footage does not include the area of the basement. It is a continuous variable and is independent as well.

v. Miles to Resort – As mentioned in the introduction, this is the data collected by a real estate company that holds properties near a ski resort. 'Miles to Resort' is the total distance (in miles) of a particular house from the downtown resort area. Since it is a distance measure, it is a continuous variable. It is also independent.

vi. Miles to Base – 'Miles to Base' is the total distance (in miles) from a particular house to the base the mountain at the ski resort. It is also continuous and independent.

vii. Acres – 'Acres' is the total lot size of a house, which is the boundary-to-boundary area of any house's plot. It is established after a survey done by a governing body, and it includes the square footage as well. It is a continuous and independent variable.

viii. Cars – 'Cars' indicates the total number of cars that can be accommodated by the garage of that particular house. If the value of 'Cars' is 0, it indicates that the house does not include a garage. It is a discrete (but numeric) and independent variable.

ix. Years Old – 'Years Old' indicates the age of the property in years at the time it was listed. It is a continuous and independent variable.

x. DoM – 'DoM' stands for 'Days on Market'. This column indicates the number of days a particular house was available for in the market, before it was sold. It is a discrete (but numeric) and independent variable.

- Using the .info() function of Python Pandas, we can see the information related to our dataset.

```
import pandas as pd
```

```
hp=pd.read_csv('Housing Prices.csv')
```

```
hp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Price            100 non-null    float64
 1   Beds             100 non-null    int64  
 2   Baths            100 non-null    float64
 3   Square Feet      100 non-null    int64  
 4   Miles to Resort  100 non-null    int64  
 5   Miles to Base    100 non-null    int64  
 6   Acres            100 non-null    float64
 7   Cars             100 non-null    int64  
 8   Years Old        100 non-null    int64  
 9   DoM              100 non-null    int64  
dtypes: float64(3), int64(7)
memory usage: 7.9 KB
```

The dataset contains 100 rows, with no null values.

- To get a better understanding of the dataset and what it looks like, here are the first 5 rows (using Python):

```
hp.head()
```

	Price	Beds	Baths	Square Feet	Miles to Resort	Miles to Base	Acres	Cars	Years Old	DoM
0	330.0	3	2.0	1771	15	20	0.23	2	4	127
1	400.0	3	2.0	1213	5	1	0.17	1	5	98
2	416.0	3	2.5	1884	2	7	0.18	2	16	105
3	420.0	3	2.0	1922	1	6	0.29	1	80	103
4	496.0	4	2.5	1858	0	5	0.52	2	9	39

Here, hp is the variable name given to the Python DataFrame 'Housing Prices.csv' (CSV version of the same JMP file).

- **Descriptive Statistics** – using the .describe() function of Python, we can get the basic descriptive statistics of each individual column; however, this function does not include the measures of skewness and kurtosis, hence the code for it is given separately.

```
hp.describe()
```

	Price	Beds	Baths	Square Feet	Miles to Resort	Miles to Base	Acres	Cars	Years Old	DoM
count	100.000000	100.000000	100.000000	100.000000	100.00000	100.000000	100.000000	100.000000	100.000000	100.000000
mean	398.952000	3.330000	2.397500	1932.320000	12.14000	13.370000	2.169100	1.480000	21.450000	121.690000
std	130.433066	1.155182	1.007607	667.175334	13.46077	13.591185	6.410047	1.058682	19.463187	93.230971
min	160.000000	1.000000	1.000000	768.000000	0.00000	1.000000	0.100000	0.000000	3.000000	16.000000
25%	300.000000	3.000000	2.000000	1452.000000	2.00000	5.000000	0.237500	0.000000	11.000000	61.000000
50%	384.000000	3.000000	2.000000	1892.000000	7.00000	7.000000	0.450000	2.000000	17.000000	96.500000
75%	515.750000	4.000000	2.750000	2265.000000	20.00000	15.000000	0.970000	2.000000	23.000000	150.000000
max	690.000000	6.000000	4.750000	3875.000000	52.00000	50.000000	40.000000	4.000000	80.000000	412.000000

Here std is the standard deviation, 25% is the first quartile, 50% is the median, and 75% is the third quartile.

$$Skewness = \gamma_1 = \frac{\mu_3}{\mu_2^{3/2}},$$

where $\mu_2 = E[(X - E(X))^2] = E(X^2) - (E(X))^2$

$\mu_3 = E[(X - E(X))^3] = E(X^3) - 3E(X^2)E(X) + 2(E(X))^3$

$$Kurtosis = \gamma_2 = \frac{\mu_4}{\mu_2^2} - 3.$$

where $\mu_4 = E[(X - E(X))^4] = E(X^4) - 4E(X^3)E(X) + 6E(X^2)\big(E(X)\big)^2 - 3(E(X))^4$

```
from scipy.stats import skew
for i in hp.columns:
    print("Skewness of",i,"=",skew(hp[i], axis=0, bias=True))
```

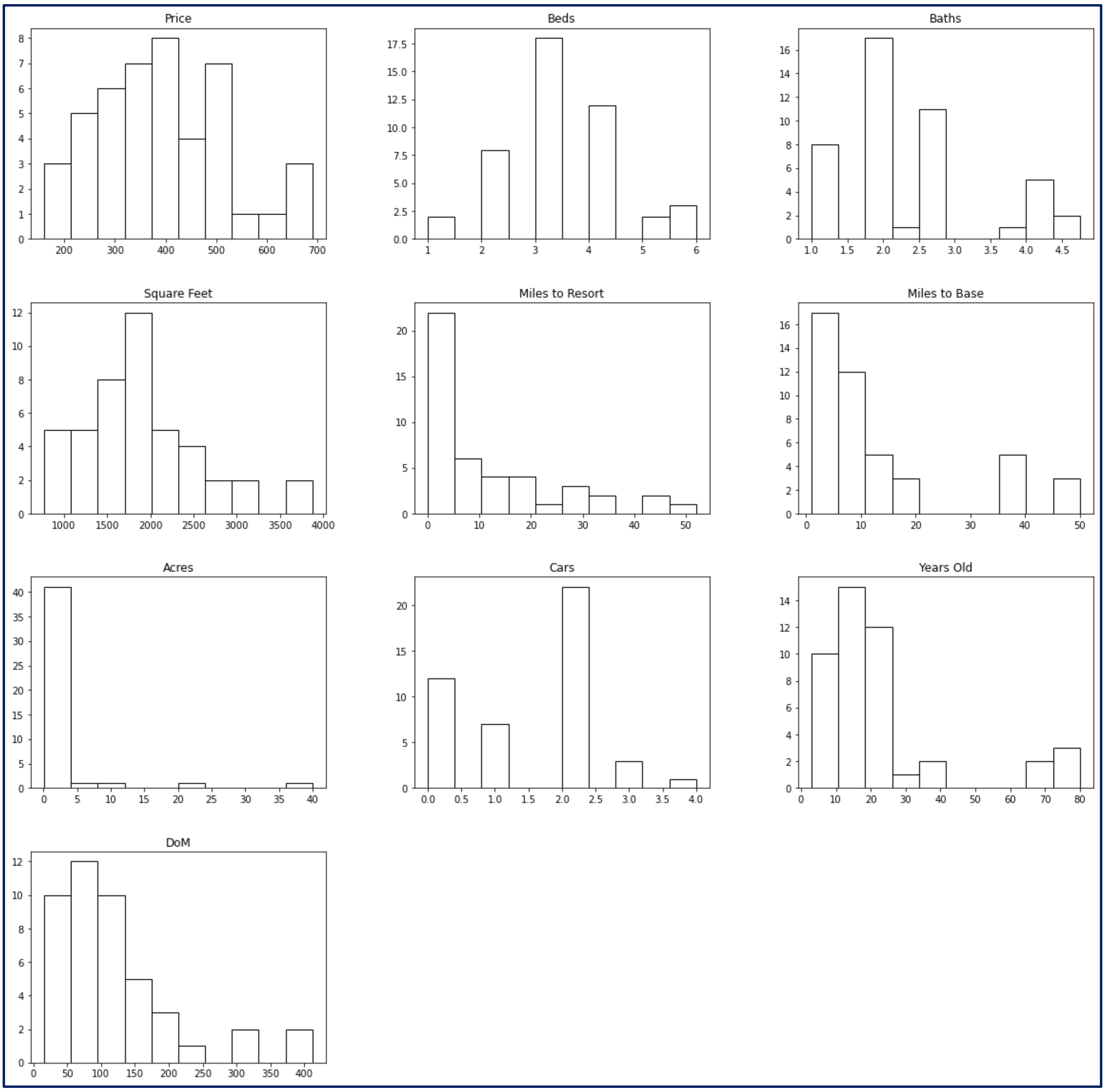
```
Skewness of Price = 0.25697422412547405
Skewness of Beds = 0.4782303743888308
Skewness of Baths = 0.5665763499999538
Skewness of Square Feet = 0.6140590060791865
Skewness of Miles to Resort = 1.2681972534753028
Skewness of Miles to Base = 1.5049514184702026
Skewness of Acres = 4.798317865924989
Skewness of Cars = -0.024311584904528755
Skewness of Years Old = 1.8951189703210303
Skewness of DoM = 1.6255404065469934
```

```
from scipy.stats import kurtosis
for j in hp.columns:
    print("Kurtosis of",j,"=",kurtosis(hp[j], axis=0, bias=True))
```

```
Kurtosis of Price = -0.6125917602846536
Kurtosis of Beds = 0.275292102061621 7
Kurtosis of Baths = -0.4488151578796389
Kurtosis of Square Feet = 0.2009916122799713
Kurtosis of Miles to Resort = 0.7427469380373943
Kurtosis of Miles to Base = 1.103593297460118
Kurtosis of Acres = 23.659198886028324
Kurtosis of Cars = -0.6222103589360266
Kurtosis of Years Old = 2.890560314098032
Kurtosis of DoM = 2.4396119534980683
```

Although these statistics may give us a rough measure of how our data is, visualising the data would truly give a good representation of the information available. For that, let's take a look at the histograms and boxplots of each variable.

```python
df.hist(bins=10,figsize=(20,20),grid=False,color='white',edgecolor='black')
```



The histograms visualise the shape of each distribution and explain the values of skewness and kurtosis (in simple terms, skewness indicates the symmetry of the distribution, and kurtosis indicates how peaked the distribution is) as well. For example, if we consider ‘Acres’, we can understand why it has the highest values for skewness and kurtosis.

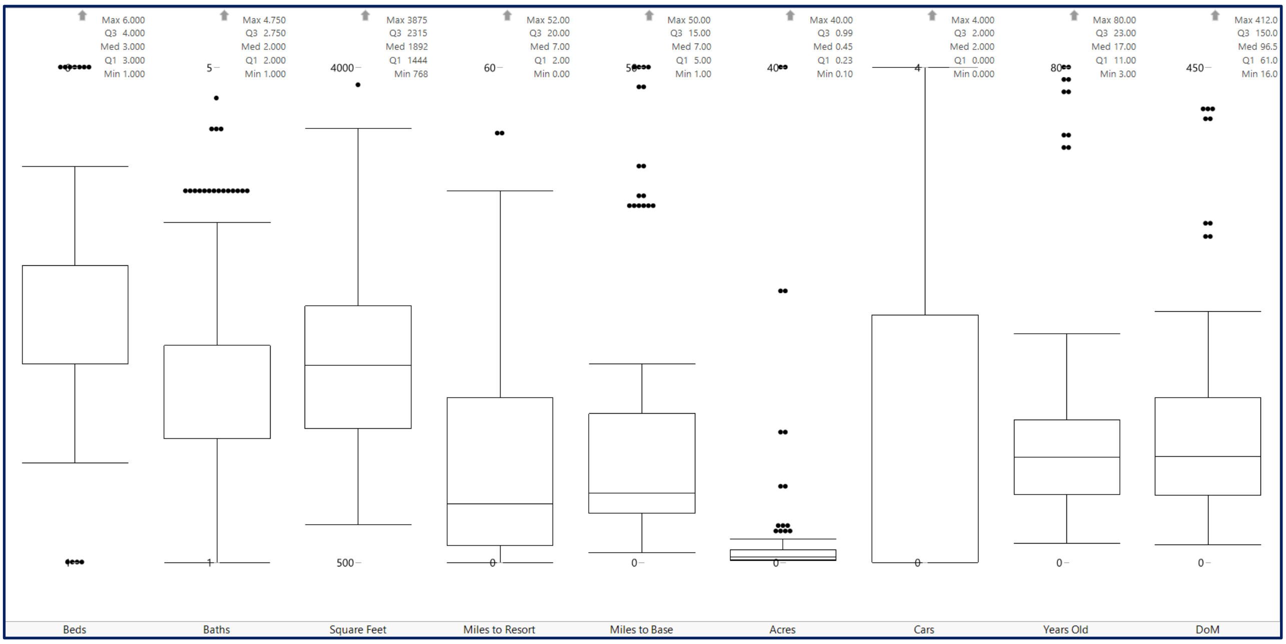
This part of the project was performed using JMP. The JSL (JMP Scripting Language) code which was used is as follows:

Script for Housing Prices - JMP Pro

Name: Boxplot1

Script:

```
Graph Builder(
    Size( 1431, 742 ),
    Show Control Panel( 0 ),
    Variables(
        X( :Beds ),
        X( :Baths, Position( 1 ) ),
        X( :Square Feet, Position( 1 ) ),
        X( :Miles to Resort, Position( 1 ) ),
        X( :Miles to Base, Position( 1 ) ),
        X( :Acres, Position( 1 ) ),
        X( :Cars, Position( 1 ) ),
        X( :Years Old, Position( 1 ) ),
        X( :DoM, Position( 1 ) )
    ),
    Elements(
        Box Plot(
            X( 1 ),
            X( 2 ),
            X( 3 ),
            X( 4 ),
            X( 5 ),
            X( 6 ),
            X( 7 ),
            X( 8 ),
            X( 9 ),
            Legend( 4 ),
            "5 Number Summary"n( 1 )
        )
    ),
    SendToReport(
        Dispatch(
            {},
            "Graph Builder",
            FrameBox,
            {DispatchSeg( ParallelAxisSeg( 1 ), {Transparency( 0.25 )} )}
        )
    )
)
```



It is quite evident that the data contains a lot of outliers. However, since the size of the data is not too big, getting rid of the outliers would not be viable; hence, before proceeding further the outliers have not been excluded.

- Correlations

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

r = correlation coefficient

x_i = values of the x-variable in a sample

$\bar{x}$ = mean of the values of the x-variable

y_i = values of the y-variable in a sample

$\bar{y}$ = mean of the values of the y-variable

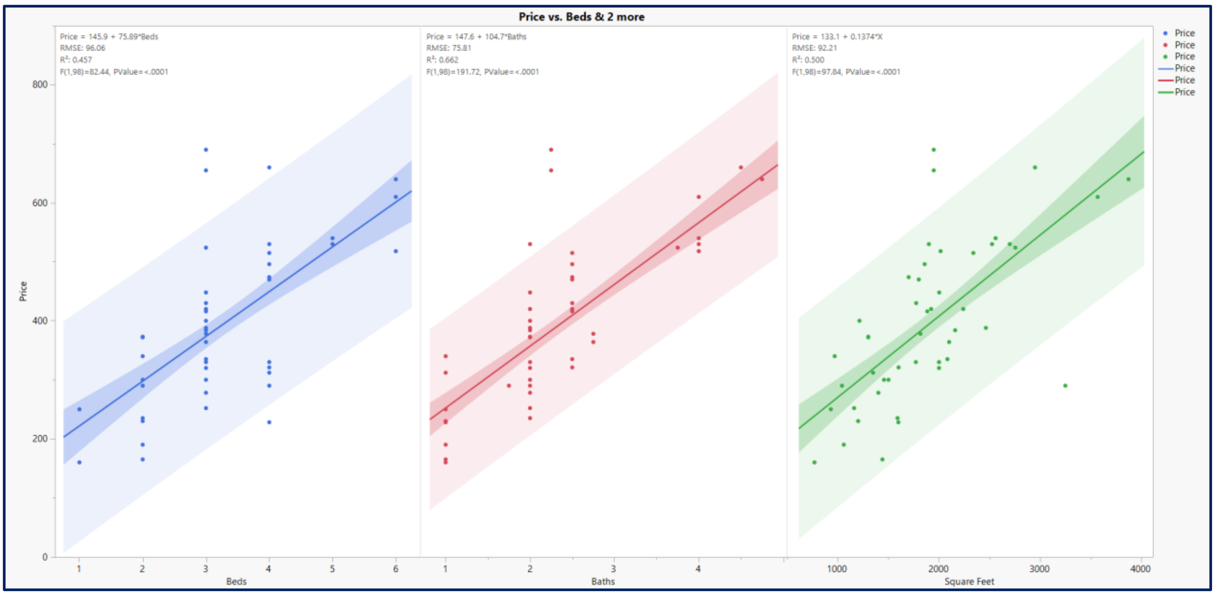
The Y variable will remain fixed throughout this entire exercise, which is 'Price'. The X variables are all the other independent variables of the dataset. To see the individual least square lines of 'Price' compared with other independent variables, we use the following formula:

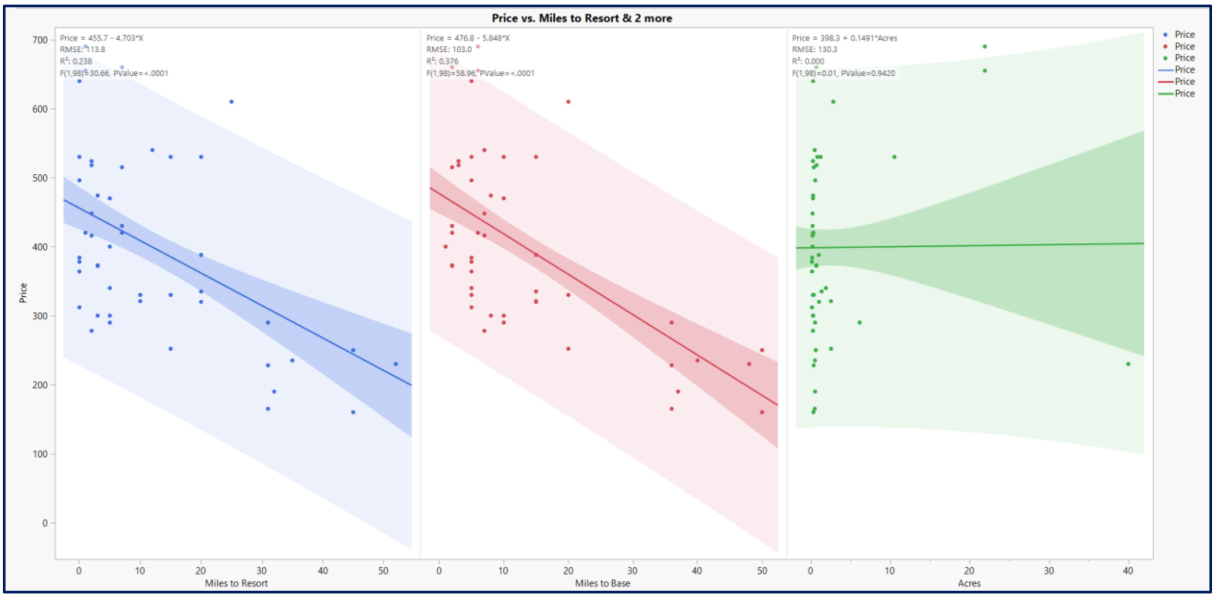
$$\hat{y} = mx + c$$

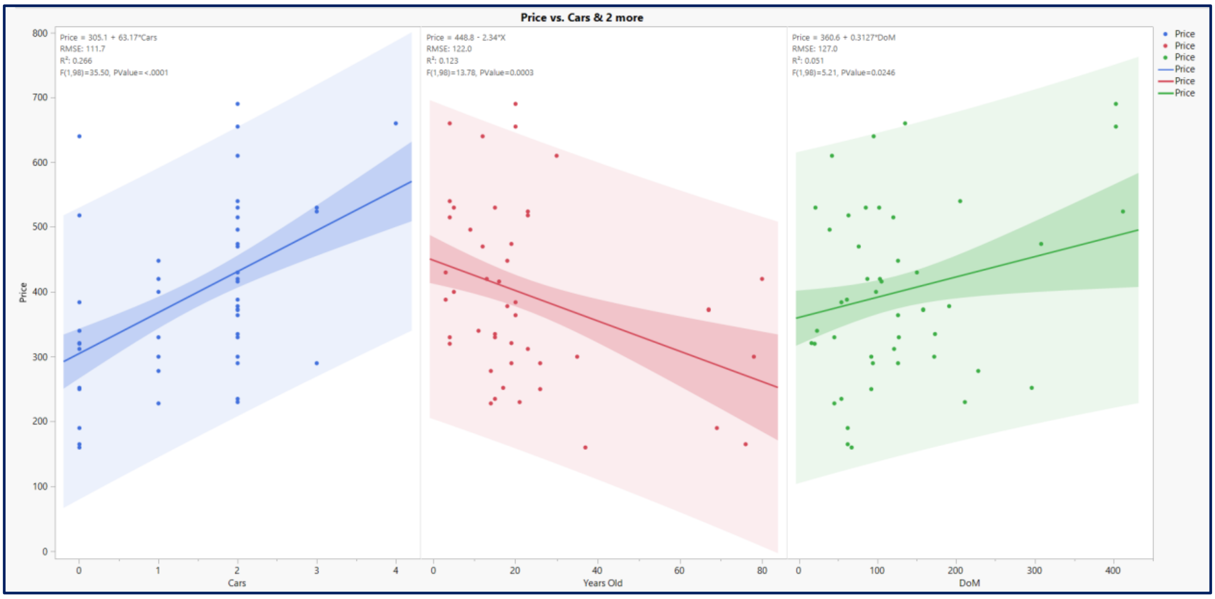
$$m = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$c = \bar{y} - m\bar{x}$$

Using this formula, let's have a look at the least square lines of Price vs. its other variables.


Price vs. Beds & 2 more
Price = 145.9 + 75.89*Beds
RMSE: 96.06
R²: 0.457
F(1,98)=82.44, PValue=<.0001
Price = 147.6 + 104.7*Baths
RMSE: 75.81
R²: 0.662
F(1,98)=191.72, PValue=<.0001
Price = 133.1 + 0.1374*X
RMSE: 92.21
R²: 0.500
F(1,98)=97.84, PValue=<.0001
Price
Beds
Baths
Square Feet


Price vs. Miles to Resort & 2 more
Price = 455.7 - 4.703*X
RMSE: 113.8
R²: 0.238
F(1,98)=30.66, PValue=<.0001
Price = 476.8 - 5.848*X
RMSE: 103.0
R²: 0.376
F(1,98)=58.96, PValue=<.0001
Price = 398.3 + 0.1491*Acres
RMSE: 130.3
R²: 0.000
F(1,98)=0.01, PValue=0.9420
Price
Miles to Resort
Miles to Base
Acres

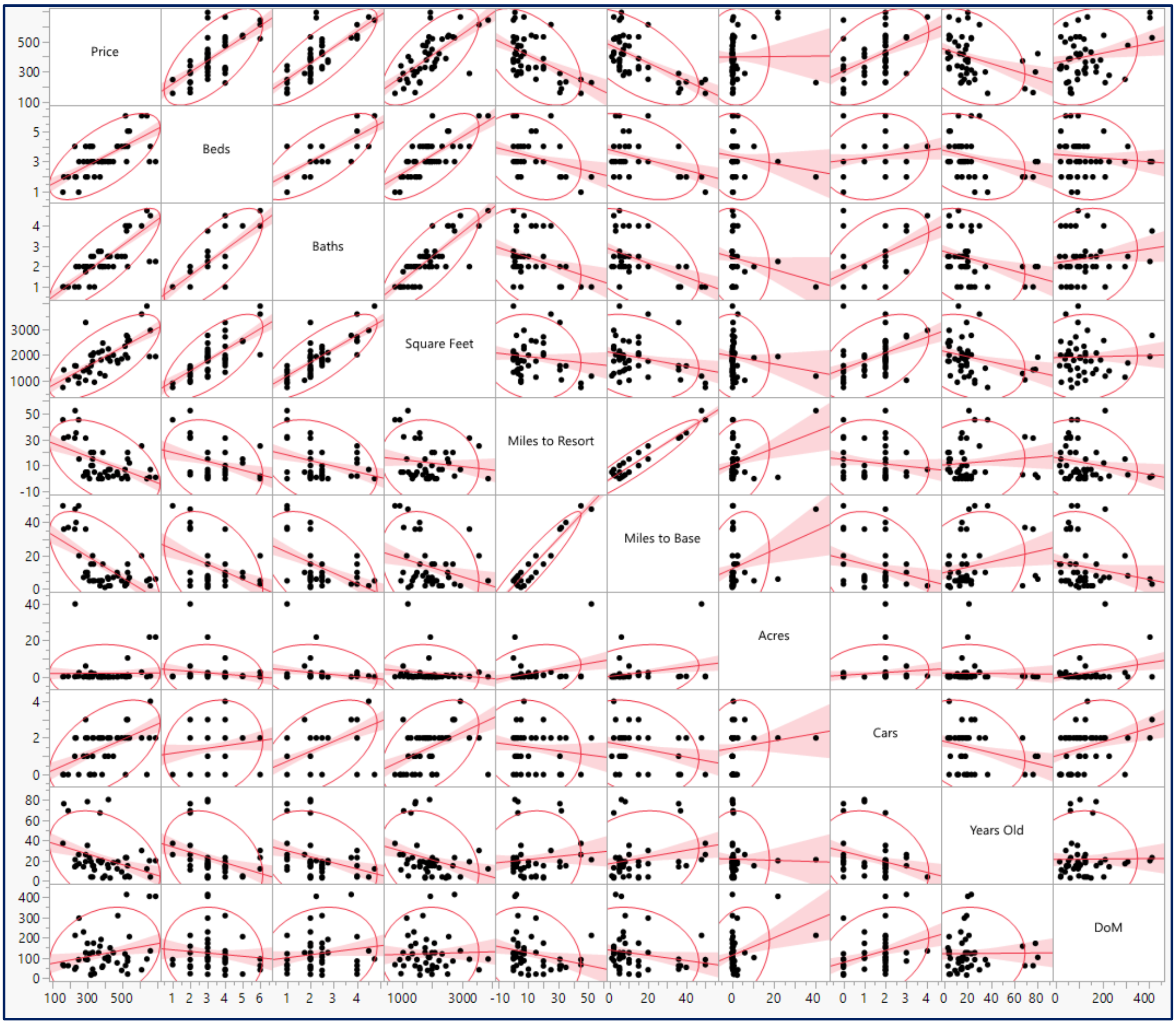

Price vs. Cars & 2 more
Price = 305.1 + 63.17*Cars
RMSE: 111.7
R²: 0.266
F(1,98)=35.50, PValue=<.0001
Price = 448.8 - 2.34*X
RMSE: 122.0
R²: 0.123
F(1,98)=13.78, PValue=0.0003
Price = 360.6 + 0.3127*DoM
RMSE: 127.0
R²: 0.051
F(1,98)=5.21, PValue=0.0246
Price
Cars
Years Old
DoM

- We can notice that Price is positively related to Beds, Baths, Square Feet, Cars, and DoM; and negatively related to Miles to Resort, Miles to Base, and Years Old.
- From this information, we can infer that the housing price is high for houses with a greater number of bedrooms and bathrooms, and with a higher liveable and garage area.
- The houses closer to the resort and its base are also more expensive comparatively.
- It can also be seen that newer houses fetch a higher price.
- This information can also be visualised in the form of a correlation matrix and a correlation scatterplot.
- Correlation matrix (using Python)

```
hp.corr().style.background_gradient(cmap='Greys')
```

	Price	Beds	Baths	Square Feet	Miles to Resort	Miles to Base	Acres	Cars	Years Old	DoM
Price	1.000000	0.671286	0.808437	0.702842	-0.487651	-0.610893	0.015671	0.514059	-0.349306	0.231682
Beds	0.671286	1.000000	0.725769	0.696964	-0.290123	-0.388084	-0.140068	0.149991	-0.308576	-0.097426
Baths	0.808437	0.725769	1.000000	0.782917	-0.321031	-0.463361	-0.189649	0.520042	-0.307434	0.159684
Square Feet	0.702842	0.696964	0.782917	1.000000	-0.130917	-0.273228	-0.143935	0.480701	-0.289013	0.030162
Miles to Resort	-0.487651	-0.290123	-0.321031	-0.130917	1.000000	0.943572	0.297746	-0.136602	0.106130	-0.223064
Miles to Base	-0.610893	-0.388084	-0.463361	-0.273228	0.943572	1.000000	0.269524	-0.251852	0.228742	-0.187330
Acres	0.015671	-0.140068	-0.189649	-0.143935	0.297746	0.269524	1.000000	0.121300	-0.020609	0.297332
Cars	0.514059	0.149991	0.520042	0.480701	-0.136602	-0.251852	0.121300	1.000000	-0.281677	0.334942
Years Old	-0.349306	-0.308576	-0.307434	-0.289013	0.106130	0.228742	-0.020609	-0.281677	1.000000	0.010793
DoM	0.231682	-0.097426	0.159684	0.030162	-0.223064	-0.187330	0.297332	0.334942	0.010793	1.000000

- Correlation scatterplot (using JMP)



The correlation matrix and correlation scatterplot further explain the earlier findings about ‘Price’ and its related components

5. MULTIPLE LINEAR REGRESSION

- Train-Test Split

  i. Before we perform the steps to create our model, it is important to split the data into 2 parts using a 70:30 ratio. To carry this out, we use R.

```
library(caTools)
set.seed(101)
split=sample.split(ï..Price,SplitRatio = 0.70)
train_data<-subset(hp,split==T)
test_data<-subset(hp,split==F)
```

Now that the data has been split, a multiple linear regression model can be made.

- Multiple Linear Regression Model

  - The Multiple Linear Regression model looks like this:

$$Y = \beta 0 + \beta 1x1 + \beta 2x2 + \cdots + \beta k{-}1xk{-}1 + \varepsilon$$

  - Here, we have 1 response variable, therefore Price = Y. There are 9 explanatory variables, which are Beds, Baths, Square Feet, Miles to Resort, Miles to Base, Acres, Cars, Years Old, and DoM, which assume the value for β. We also assume the error to be normally distributed

- After creating and running the model, these are the following β values

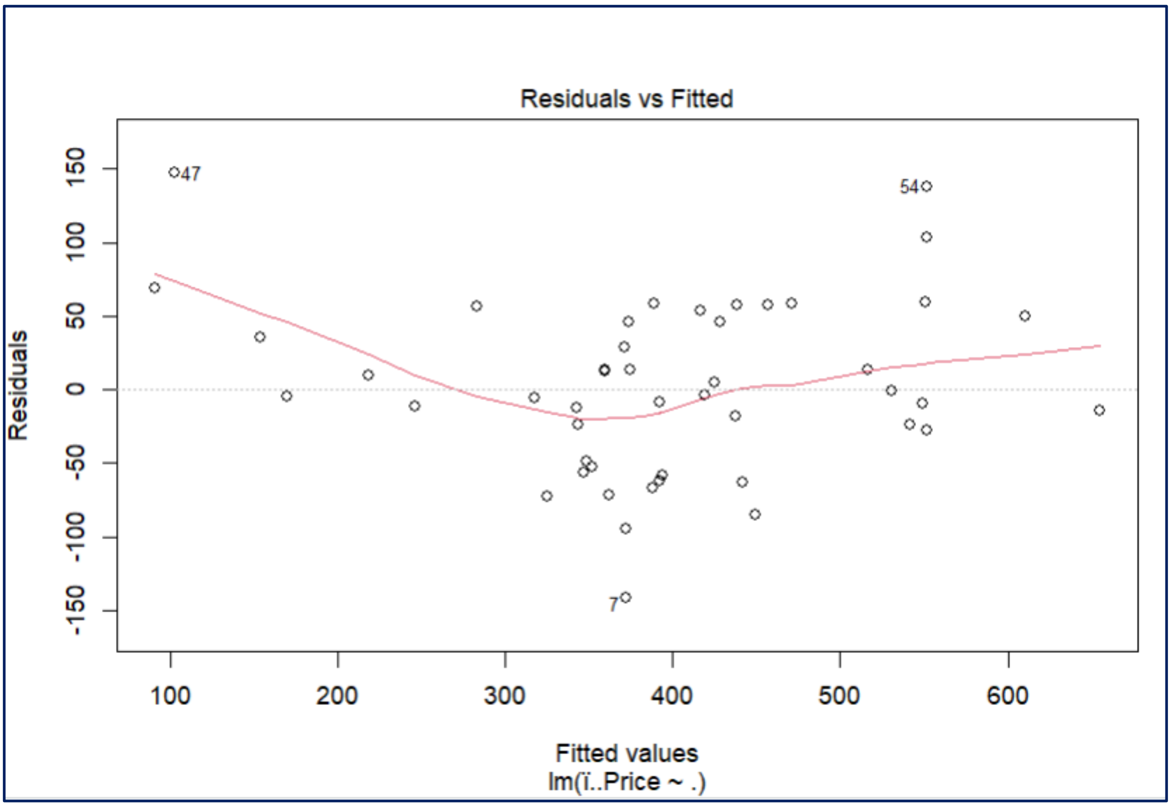
```
Call:
lm(formula = ï..Price ~ ., data = train_data)

Coefficients:
     (Intercept)            Beds           Baths     Square.Feet  Miles.to.Resort    Miles.to.Base           Acres            Cars
       175.33904        11.25319        57.99793         0.03750         -1.37805         -2.31066         5.94463         6.02647
       Years.Old             DoM
        -0.25252         0.04288
```

- The summary of this model is as follows:

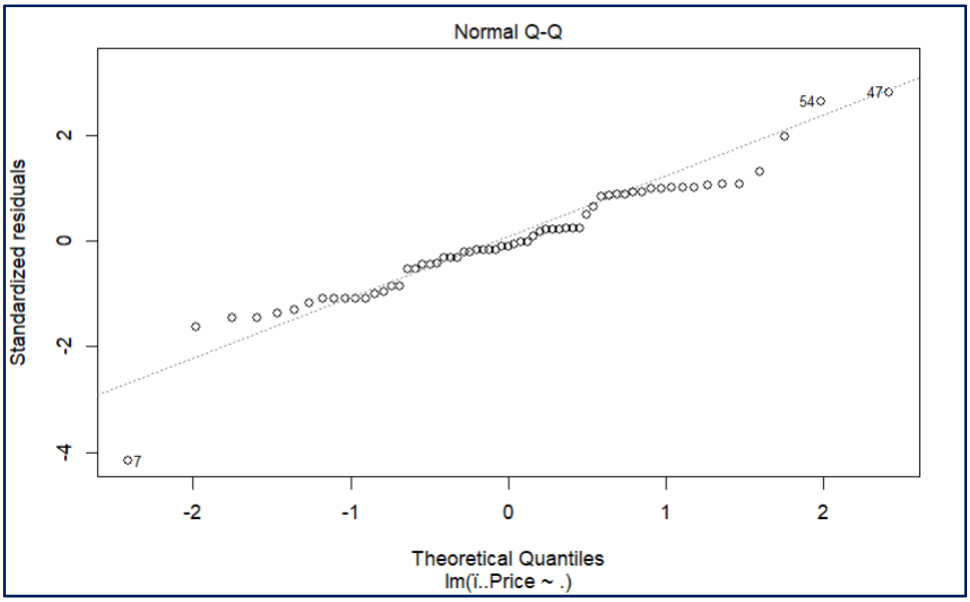
```
> summary(model1)

Call:
lm(formula = ï..Price ~ ., data = train_data)

Residuals:
     Min       1Q   Median       3Q      Max
-141.855  -38.128   -5.081   47.773  147.194

Coefficients:
                 Estimate Std. Error t value Pr(>|t|)
(Intercept)     175.33904   37.72612   4.648 2.26e-05 ***
Beds             11.25319   12.62756   0.891 0.376871
Baths            57.99793   16.08701   3.605 0.000689 ***
Square.Feet       0.03750    0.01995   1.880 0.065661 .
Miles.to.Resort  -1.37805    2.06717  -0.667 0.507898
Miles.to.Base    -2.31066    2.08910  -1.106 0.273697
Acres             5.94463    1.50543   3.949 0.000233 ***
Cars              6.02647   10.43910   0.577 0.566183
Years.Old        -0.25252    0.45633  -0.553 0.582338
DoM               0.04288    0.10473   0.409 0.683875
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 60.41 on 53 degrees of freedom
Multiple R-squared:  0.819,	Adjusted R-squared:  0.7883
F-statistic: 26.65 on 9 and 53 DF,  p-value: < 2.2e-16
```

Residual Standard Error	60.41 (53 df)
Multiple R-Squared	0.819
Adjusted R-Squared	0.7883
F-Statistic	26.65 (9 & 53 df)
p-value	<2.2e-16

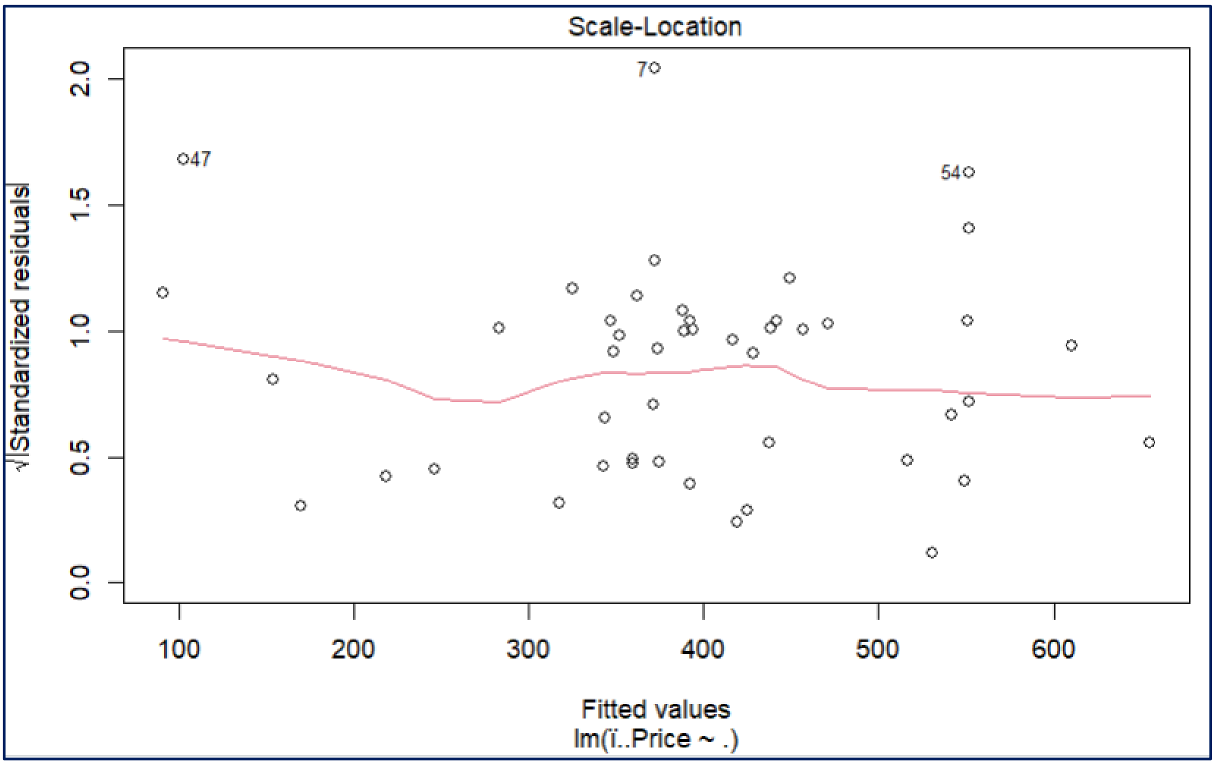
The summarization of the model indicates that the p-value is less than a significance level of 0.05. We can also visualise the model's validity using plots. 4 plots are as follows:



Residual vs Fitted plot shows the residuals (distance between actual and predicted values) plotted against the fitted values.

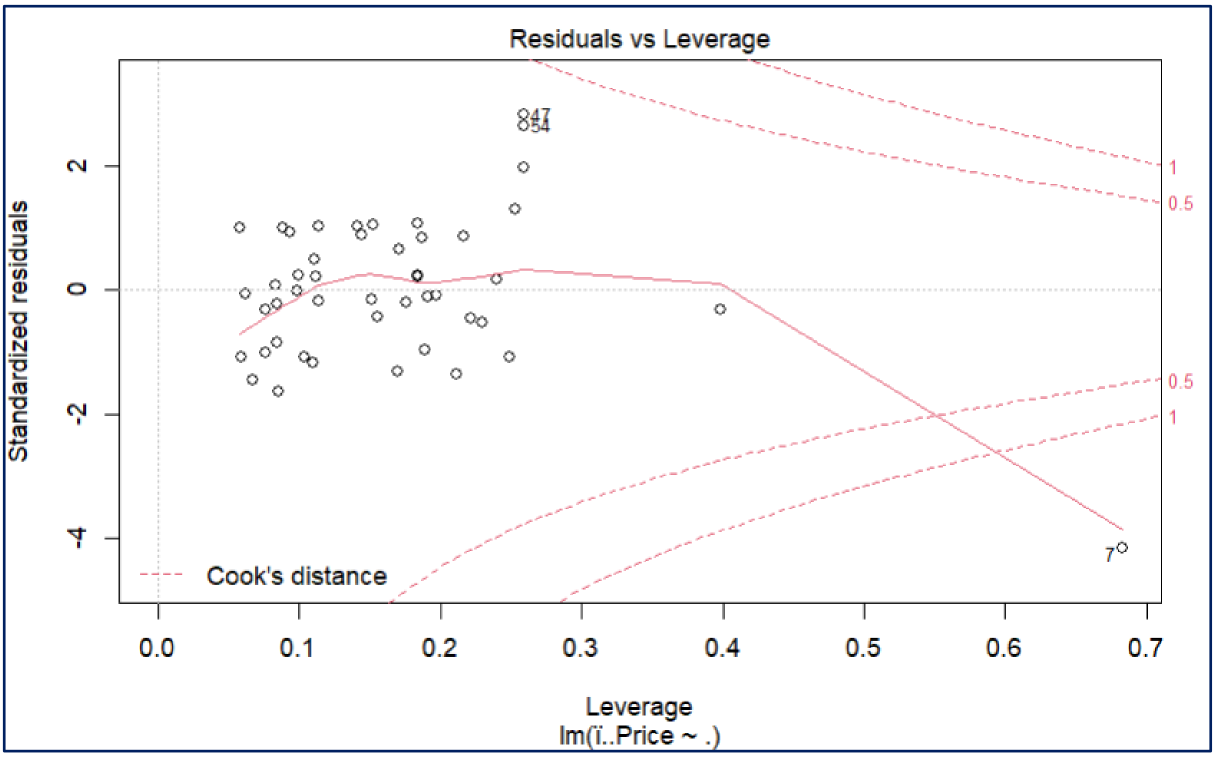


The Normal Q-Q plot shows whether or not the residuals are normally distributed. A straight line in this plot means that the errors are normall distributed



The Scale-Location plot shows the square root of standardized residuals instead of the residuals, against the fitted values of the dependent variable.

The Residual vs. Leverage plot plots standardized residuals against the leverage. The amount to which the coefficients in the regression model would vary if a specific observation was removed from the dataset is referred to as leverage. Here, point 7 lies outside the Cook's distance, hence it is an influential observation. This suggests that removing this observation from our dataset and fitting the regression model again would dramatically alter the model's coefficients.



- To further validate the model, we can use the NCV Test, which calculates a score test of the constant error variance hypothesis versus the alternative that the error variance varies with the response level (fitted values) or with a linear combination of predictors.

```
> ncvTest(model1) #NCV Test
Non-constant Variance Score Test
Variance formula: ~ fitted.values
Chisquare = 1.327438, Df = 1, p = 0.24926
```

```
vif(model1) #VIF
     Beds     Baths Square.Feet Miles.to.Resort Miles.to.Base     Acres      Cars Years.Old       DoM
 3.804825  4.629803    3.239736       11.865756     12.408407  1.525329  2.020131  1.259747  1.776952
```

- VIF (Variance Inflation Factor) is used to deal with the problem of multicollinearilty. Multicollinearilty occurs when there is high correlation between the independent variables. Because multicollinearity causes the estimated coefficients to have a significant variation, the coefficient estimates corresponding to those connected explanatory variables will not accurately reflect the true picture. They can become extremely sensitive to even minor model alterations.

- To tackle the problem of multicollinearity, the Akaike Information Crieterion can be used. It is also called 'step-wise regression'. Its formula is:

$$AIC = \frac{1}{n\hat{\sigma}^2}(RSS + 2d\hat{\sigma}^2)$$

n: number of observations

$\hat{\sigma}^2$: estimate of error or residual variance

d: number of x variables included in the model

RSS: Residual sum of squares

```
> library(MASS)
> model2=stepAIC(model1,direction = 'both') #Stepwise regression
Start:  AIC=525.86
ï..Price ~ Beds + Baths + Square.Feet + Miles.to.Resort + Miles.to.Base +
    Acres + Cars + Years.Old + DoM

                  Df Sum of Sq    RSS    AIC
- DoM              1       612 194029 524.06
- Years.Old        1      1118 194535 524.22
- Cars             1      1216 194634 524.25
- Miles.to.Resort  1      1622 195039 524.38
- Beds             1      2898 196316 524.79
- Miles.to.Base    1      4465 197882 525.29
<none>                         193417 525.86
- Square.Feet      1     12893 206311 527.92
- Baths            1     47434 240852 537.67
- Acres            1     56905 250322 540.10

Step:  AIC=524.06
ï..Price ~ Beds + Baths + Square.Feet + Miles.to.Resort + Miles.to.Base +
    Acres + Cars + Years.Old

                  Df Sum of Sq    RSS    AIC
- Years.Old        1      1053 195082 522.40
- Cars             1      1512 195542 522.54
- Beds             1      2380 196409 522.82
- Miles.to.Resort  1      2754 196783 522.94
- Miles.to.Base    1      3918 197947 523.32
<none>                         194029 524.06
+ DoM              1       612 193417 525.86
- Square.Feet      1     13159 207188 526.19
- Baths            1     54382 248411 537.62
- Acres            1     82640 276669 544.41

Step:  AIC=522.4
ï..Price ~ Beds + Baths + Square.Feet + Miles.to.Resort + Miles.to.Base +
    Acres + Cars

                  Df Sum of Sq    RSS    AIC
- Cars             1      1890 196973 521.00
- Miles.to.Resort  1      2152 197234 521.09
- Beds             1      3138 198220 521.40
- Miles.to.Base    1      5244 200327 522.07
<none>                         195082 522.40
+ Years.Old        1      1053 194029 524.06
+ DoM              1       547 194535 524.22
- Square.Feet      1     13352 208435 524.57
- Baths            1     53501 248583 535.67
- Acres            1     83438 278521 542.83
```

```
Step:  AIC=521
ï..Price ~ Beds + Baths + Square.Feet + Miles.to.Resort + Miles.to.Base +
    Acres

                  Df Sum of Sq    RSS    AIC
- Beds             1      1537 198509 519.49
- Miles.to.Resort  1      1681 198654 519.54
<none>                         196973 521.00
- Miles.to.Base    1      6593 203566 521.08
+ Cars             1      1890 195082 522.40
+ Years.Old        1      1431 195542 522.54
+ DoM              1       859 196113 522.73
- Square.Feet      1     18885 215857 524.77
- Baths            1     74140 271113 539.13
- Acres            1     97415 294387 544.32

Step:  AIC=519.49
ï..Price ~ Baths + Square.Feet + Miles.to.Resort + Miles.to.Base +
    Acres

                  Df Sum of Sq    RSS    AIC
- Miles.to.Resort  1      1868 200377 518.08
<none>                         198509 519.49
- Miles.to.Base    1      6430 204940 519.50
+ Years.Old        1      1860 196650 520.90
+ Beds             1      1537 196973 521.00
+ Cars             1       289 198220 521.40
+ DoM              1       116 198393 521.46
- Square.Feet      1     25680 224189 525.16
- Baths            1     95942 294451 542.33
- Acres            1     97663 296172 542.70

Step:  AIC=518.08
ï..Price ~ Baths + Square.Feet + Miles.to.Base + Acres

                  Df Sum of Sq    RSS    AIC
<none>                         200377 518.08
+ Miles.to.Resort  1      1868 198509 519.49
+ Beds             1      1723 198654 519.54
+ Years.Old        1      1025 199352 519.76
+ DoM              1       556 199821 519.91
+ Cars             1       115 200262 520.05
- Square.Feet      1     24465 224842 523.34
- Baths            1     94233 294610 540.37
- Acres            1     95909 296285 540.72
- Miles.to.Base    1    116520 316897 544.96
```

- Once the step-wise regression is done, we can re-run the VIF on model2, which was created for the AIC process.

```
> library(car)
> vif(model2) #VIF of new model
      Baths   Square.Feet Miles.to.Base         Acres
   2.959097      2.519954      1.325209      1.062223
```

The significant variables in the second model are Baths, Square Feet, Miles to Base, and Acres. Hence, the problem of multicollinearilty has been solved.

The summarization of model2 is as follows:

```
> summary(model2)

Call:
lm(formula = ï..Price ~ Baths + Square.Feet + Miles.to.Base + 
    Acres, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-153.09  -42.38   -4.16   44.29  145.62 

Coefficients:
               Estimate Std. Error t value Pr(>|t|)    
(Intercept)   190.63083   27.02787   7.053 2.39e-09 ***
Baths          65.35310   12.51336   5.223 2.49e-06 ***
Square.Feet     0.04556    0.01712   2.661   0.0101 *  
Miles.to.Base  -3.85774    0.66427  -5.808 2.82e-07 ***
Acres           6.44028    1.22232   5.269 2.10e-06 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 58.78 on 58 degrees of freedom
Multiple R-squared:  0.8125,	Adjusted R-squared:  0.7996 
F-statistic: 62.83 on 4 and 58 DF,  p-value: < 2.2e-16
```

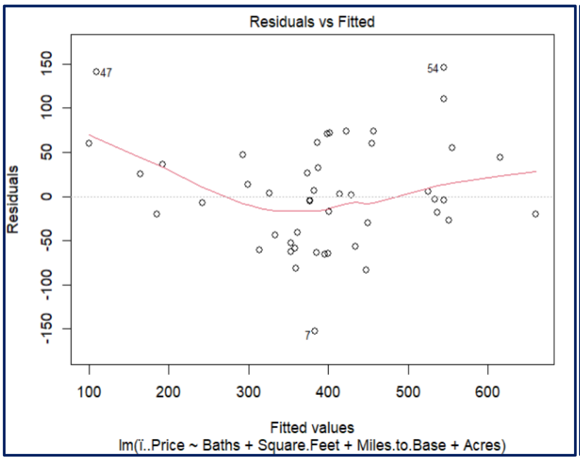
Residual Standard Error	58.78 (58 df)
Multiple R-Squared	0.8125
Adjusted R-Squared	0.7996
F-Statistic	62.83 (4 & 58 df)
p-value	<2.2e-16

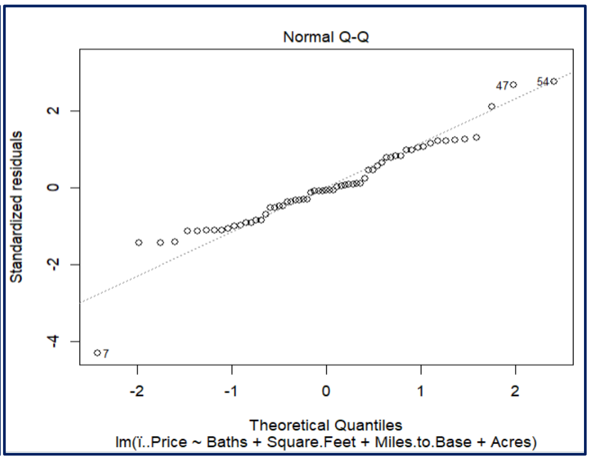
```
> ncvTest(model2) #NCV Test of new model (after AIC)
Non-constant Variance Score Test 
Variance formula: ~ fitted.values 
Chisquare = 0.6669324, Df = 1, p = 0.41412
```

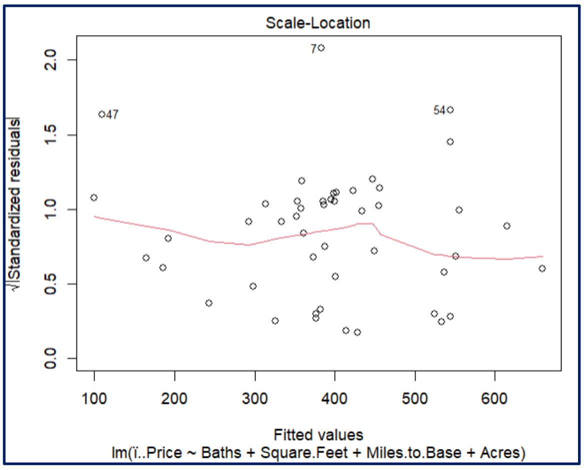
Compared to the earlier model, the new model has a better value for the adjusted R-Squared, hence it has proved to be a better fit for our data. Therefore, our final model equation is:

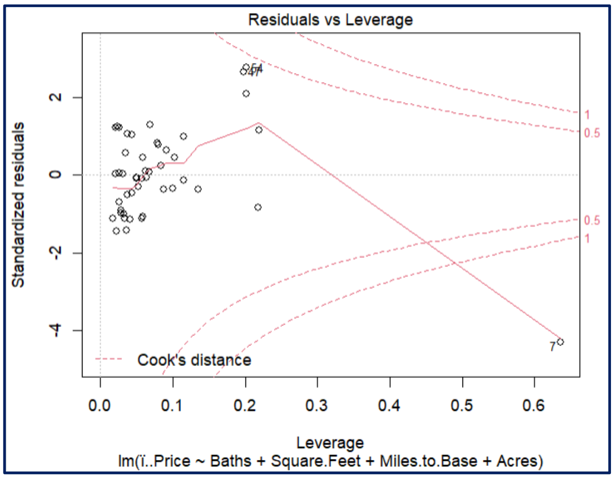
Y = 190.63 + (65.35*Baths) + (0.045*Sq.Ft.) + (-3.85*Mi. to Base) + (6.44*Acres)

Let's take a look at the 4 graphs that were plotted earlier, but this time – on model2.









6. HYPOTHESIS TESTING

- Hypothesis Testing using Analysis of Variance (ANOVA)
  - Our null hypothesis is that neither of the explanatory variables have any effect on the model, or statistically,

$$H_0 : \beta 0 = \beta 1 = \beta 2 = \beta 3 = \beta 4 = 0$$

  - The alternative hypothesis is that at least one explanatory variable shows some effect on the model, or statistically,

$$H_a : \beta j \neq 0$$

ANOVA using R:

```
> anova(model2) #ANOVA Table
Analysis of Variance Table

Response: ï..Price
             Df Sum Sq Mean Sq  F value    Pr(>F)    
Baths         1 680556  680556 196.9901 < 2.2e-16 ***
Square.Feet   1   5609    5609   1.6236    0.2077    
Miles.to.Base 1  86192   86192  24.9487 5.724e-06 ***
Acres         1  95909   95909  27.7612 2.105e-06 ***
Residuals    58 200377    3455                       
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

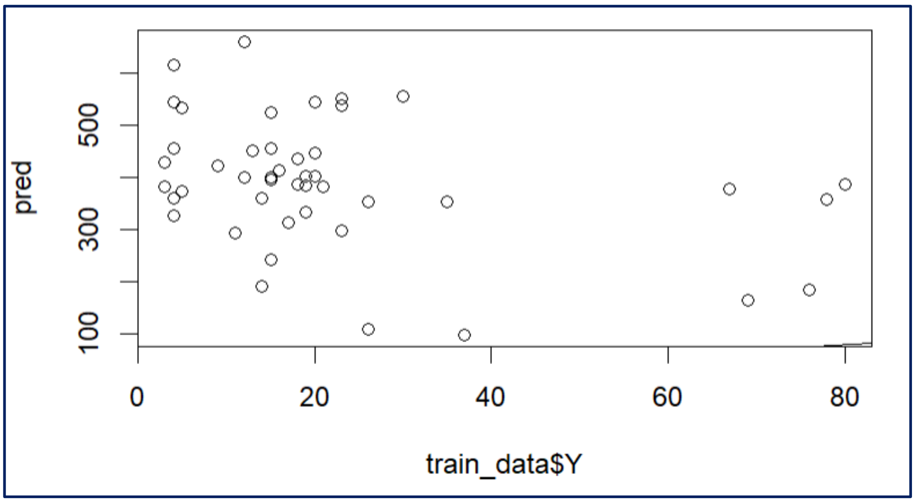
We can see that the output of the R code rejects the null hypothesis, as, as 3 variables have been deemed to be significant.

7. EVALUATION

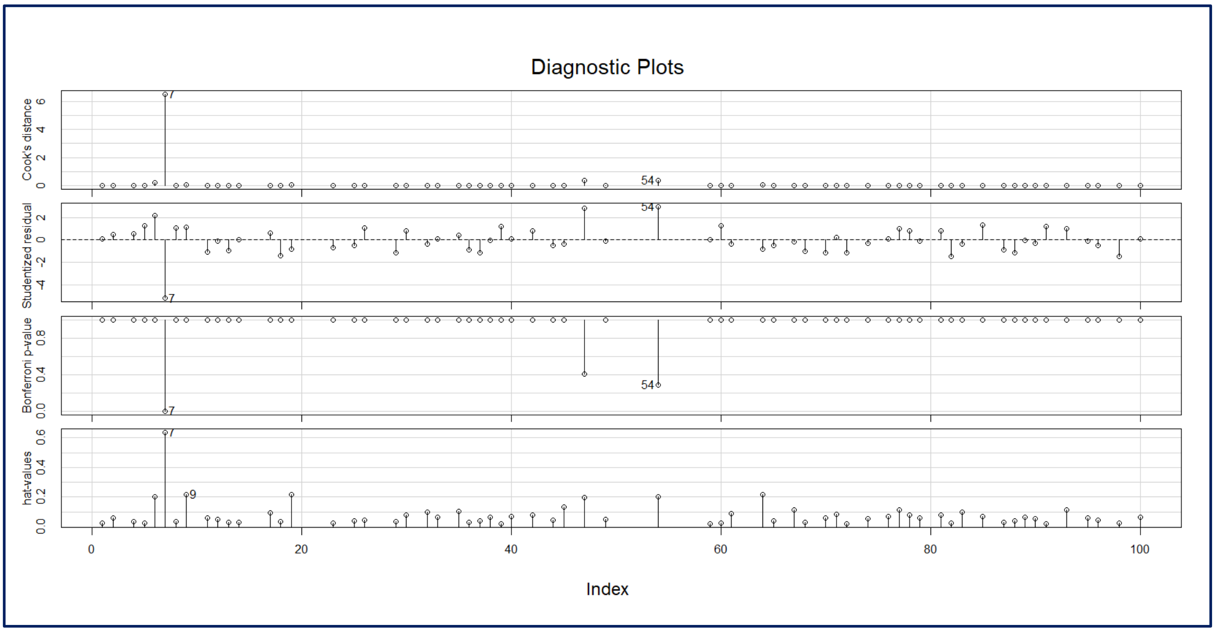
- In this section, let’s have a look at the evaluation metrics of our model on both, the training and testing data. The metrics used will be RMSE and MSE, which are Root of Mean Squared Error and Mean Squared Error respectively.
- A regression line's mean squared error (MSE) indicates how near it is to a set of points. It accomplishes this by squaring the distances between the points and the regression line (these lengths are the "errors"). Squaring is required to eliminate any negative signs. RMSE is just the square root of MSE, and it is a metric that is interpretable in terms of the Y units.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2 \qquad RMSE = \sqrt{\sum_{i=1}^{n}\frac{(\hat{y}_i - y_i)^2}{n}}$$

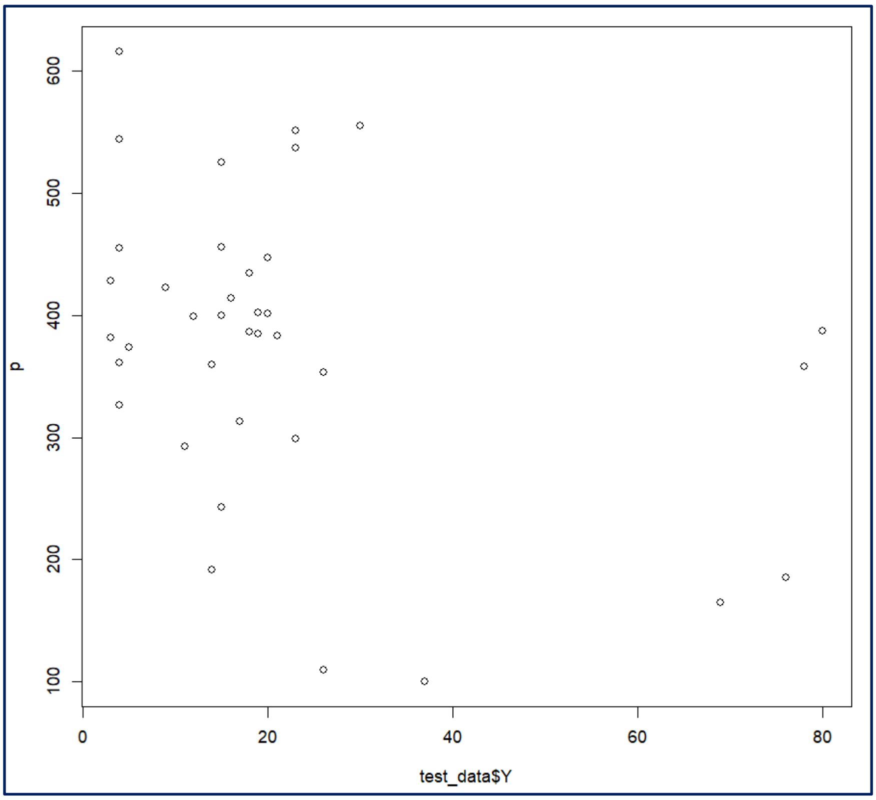
- For the training data, MSE = 3180.586 and RMSE = 56.3988



- Diagnostic Plot



- For the testing data, MSE = 142450.6 and RMSE = 377.4263



8. CONCLUSION

Housing prices and the real estate market are important drivers of the economy, using this model, we have predicted the housing prices based on its independent variables. The first model had a high degree of multicollinearity, which has addressed and sorted in the second model, which depicted a higher R-Squared adjusted value.

9. BIBLIOGRAPHY

https://community.jmp.com/

https://www.statology.org/

https://www.investopedia.com/terms/r/realestate.asp

Introduction to Statistics & Data Analytics (4th Edition) – Roxy Peck, Chris Olsen, Jay Devore.

10. APPENDIX

JSL Code, Python Code, and R Code attached along with the report.