# Project_2 - Python ML - Linear Regression - Home Price Prediction & Analysis

# Part 1- Data Exploration and Pre-Processing

importing the required libraries for linear Regression

In [17]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
```

1. Load the given dataset

In [18]:
```
df = pd.read_csv('Project2_Linear_Reg.csv')
df.head()
```

Out[18]:

| | Suburb | Address | Rooms | Type | Price | Method | SellerG | Date | Distance | Postcode | ... | Bathroom | Car | Landsize | BuildingArea |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Abbotsford | 68 Studley St | 2 | h | NaN | SS | Jellis | 03/09/2016 | 2.5 | 3067.0 | ... | 1.0 | 1.0 | 126.0 | NaN |
| 1 | Abbotsford | 85 Turner St | 2 | h | 1480000.0 | S | Biggin | 03/12/2016 | 2.5 | 3067.0 | ... | 1.0 | 1.0 | 202.0 | NaN |
| 2 | Abbotsford | 25 Bloomburg St | 2 | h | 1035000.0 | S | Biggin | 04/02/2016 | 2.5 | 3067.0 | ... | 1.0 | 0.0 | 156.0 | 79.0 |
| 3 | Abbotsford | 18/659 Victoria St | 3 | u | NaN | VB | Rounds | 04/02/2016 | 2.5 | 3067.0 | ... | 2.0 | 1.0 | 0.0 | NaN |
| 4 | Abbotsford | 5 Charles St | 3 | h | 1465000.0 | SP | Biggin | 04/03/2017 | 2.5 | 3067.0 | ... | 2.0 | 0.0 | 134.0 | 150.0 |

5 rows × 21 columns

2. print all the column names

In [19]:
```
df.columns
```

Out[19]:
```
Index(['Suburb', 'Address', 'Rooms', 'Type', 'Price', 'Method', 'SellerG',
       'Date', 'Distance', 'Postcode', 'Bedroom2', 'Bathroom', 'Car',
       'Landsize', 'BuildingArea', 'YearBuilt', 'CouncilArea', 'Lattitude',
       'Longtitude', 'Regionname', 'Propertycount'],
      dtype='object')
```

3. Describe the data

In [20]:
```
df.describe()
```

Out[20]:

| | Rooms | Price | Distance | Postcode | Bedroom2 | Bathroom | Car | Landsize | BuildingArea | Ye |
|---|---|---|---|---|---|---|---|---|---|---|
| count | 34857.000000 | 2.724700e+04 | 34856.000000 | 34856.000000 | 26640.000000 | 26631.000000 | 26129.000000 | 23047.000000 | 13742.00000 | 15551. |
| mean | 3.031012 | 1.050173e+06 | 11.184929 | 3116.062859 | 3.084647 | 1.624798 | 1.728845 | 593.598993 | 160.25640 | 1965. |
| std | 0.969933 | 6.414671e+05 | 6.788892 | 109.023903 | 0.980690 | 0.724212 | 1.010771 | 3398.841946 | 401.26706 | 37. |
| min | 1.000000 | 8.500000e+04 | 0.000000 | 3000.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.00000 | 1196. |
| 25% | 2.000000 | 6.350000e+05 | 6.400000 | 3051.000000 | 2.000000 | 1.000000 | 1.000000 | 224.000000 | 102.00000 | 1940. |
| 50% | 3.000000 | 8.700000e+05 | 10.300000 | 3103.000000 | 3.000000 | 2.000000 | 2.000000 | 521.000000 | 136.00000 | 1970. |
| 75% | 4.000000 | 1.295000e+06 | 14.000000 | 3156.000000 | 4.000000 | 2.000000 | 2.000000 | 670.000000 | 188.00000 | 2000. |
| max | 16.000000 | 1.120000e+07 | 48.100000 | 3978.000000 | 30.000000 | 12.000000 | 26.000000 | 433014.000000 | 44515.00000 | 2106. |

4. Drop address, date, postcode, YearBuilt, lattitude, longtitude columns

In [21]:
```
df.drop(['Address', 'Date', 'Postcode', 'YearBuilt', 'Lattitude', 'Longtitude' ], axis = 1, inplace=True)
df.head()
```

Out[21]:

| | Suburb | Rooms | Type | Price | Method | SellerG | Distance | Bedroom2 | Bathroom | Car | Landsize | BuildingArea | CouncilArea | Regionnan |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | Abbotsford | 2 | h | NaN | SS | Jellis | 2.5 | 2.0 | 1.0 | 1.0 | 126.0 | NaN | Yarra City Council | Northe Metropolit |
| 1 | Abbotsford | 2 | h | 1480000.0 | S | Biggin | 2.5 | 2.0 | 1.0 | 1.0 | 202.0 | NaN | Yarra City Council | Northe Metropolit |
| 2 | Abbotsford | 2 | h | 1035000.0 | S | Biggin | 2.5 | 2.0 | 1.0 | 0.0 | 156.0 | 79.0 | Yarra City Council | Northe Metropolit |
| 3 | Abbotsford | 3 | u | NaN | VB | Rounds | 2.5 | 3.0 | 2.0 | 1.0 | 0.0 | NaN | Yarra City Council | Northe Metropolit |
| 4 | Abbotsford | 3 | h | 1465000.0 | SP | Biggin | 2.5 | 3.0 | 2.0 | 0.0 | 134.0 | 150.0 | Yarra City Council | Northe Metropolit |

5. Find the count of null value in each column

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
Suburb              0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Distance            1
Bedroom2         8217
Bathroom         8226
Car              8728
Landsize        11810
BuildingArea    21115
CouncilArea         3
Regionname          3
Propertycount       3
dtype: int64
```

6. Fill the null value of property count, distance, Bedroom2, Bathroom, Car with 0

In [23]:
```
values = {'Propertycount':0, 'Distance':0, 'Bedroom2':0, 'Bathroom':0, 'Car':0}
df.fillna(value=values, inplace=True)
df.isnull().sum()
```

Out[23]:
```
Suburb              0
Rooms               0
Type                0
Price            7610
Method              0
SellerG             0
Distance            0
Bedroom2            0
Bathroom            0
Car                 0
Landsize        11810
BuildingArea    21115
CouncilArea         3
Regionname          3
Propertycount       0
dtype: int64
```

7. Fill Null value of land size and building area columns with Mean

In [24]:
```
df['Landsize'].fillna(df['Landsize'].mean(), inplace=True)
df['BuildingArea'].fillna(df['BuildingArea'].mean(), inplace=True)
df.isnull().sum()
```

Out[24]:
```
Suburb             0
Rooms              0
Type               0
Price           7610
Method             0
SellerG            0
Distance           0
Bedroom2           0
Bathroom           0
Car                0
Landsize           0
BuildingArea       0
CouncilArea        3
Regionname         3
Propertycount      0
dtype: int64
```

In [25]:
```
df = df.dropna()
df.isnull().sum()
```

Out[25]:
```
Suburb          0
Rooms           0
Type            0
Price           0
Method          0
SellerG         0
Distance        0
Bedroom2        0
Bathroom        0
Car             0
Landsize        0
BuildingArea    0
CouncilArea     0
Regionname      0
Propertycount   0
dtype: int64
```

8. Find the unique value in method column

In [26]:
```
df['Method'].unique()
```

Out[26]:
```
array(['S', 'SP', 'PI', 'VB', 'SA'], dtype=object)
```

9. Create a dummy data for categorical data

In [27]:
```
df = pd.get_dummies(df, drop_first=True)
df.head()
```

Out[27]:

| | Rooms | Price | Distance | Bedroom2 | Bathroom | Car | Landsize | BuildingArea | Propertycount | Suburb_Aberfeldie | ... | CouncilArea_Wyndhar City Counc |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1** | 2 | 1480000.0 | 2.5 | 2.0 | 1.0 | 1.0 | 202.0 | 160.2564 | 4019.0 | 0 | ... | |
| **2** | 2 | 1035000.0 | 2.5 | 2.0 | 1.0 | 0.0 | 156.0 | 79.0000 | 4019.0 | 0 | ... | |
| **4** | 3 | 1465000.0 | 2.5 | 3.0 | 2.0 | 0.0 | 134.0 | 150.0000 | 4019.0 | 0 | ... | |
| **5** | 3 | 850000.0 | 2.5 | 3.0 | 2.0 | 1.0 | 94.0 | 160.2564 | 4019.0 | 0 | ... | |
| **6** | 4 | 1600000.0 | 2.5 | 3.0 | 1.0 | 2.0 | 120.0 | 142.0000 | 4019.0 | 0 | ... | |

5 rows × 745 columns

# Part 2- Working with Model

1. Create the target data and feature data where target data is price

In [28]:
```
predictor = df.drop('Price', axis=1)
```

```
predictor
predictor.shape
```

Out[28]: (27244, 744)

In [29]:
```
target = df[['Price']]
target
target.shape
```

Out[29]: (27244, 1)

2. Create a linear regression model for Target and feature data

3. Check if the model is overfitting or underfitting or it is accurate

4. If the model is overfitting then apply ridge and lasso regression algorithms

In [31]:
```
from sklearn.model_selection import train_test_split
predictor_train, predictor_test, target_train, target_test = train_test_split(predictor, target, test_size=0.2, r
```

In [32]:
```
predictor_train.shape
```

Out[32]: (21795, 744)

In [33]:
```
target_train.shape
```

Out[33]: (21795, 1)

In [34]:
```
predictor_test.shape
```

Out[34]: (5449, 744)

In [35]:
```
target_test.shape
```

Out[35]: (5449, 1)

In [36]:
```
rm = LinearRegression()
```

In [37]:
```
rm.fit(predictor_train, target_train)
```

Out[37]: LinearRegression()

In [38]:
```
Predicted_Price = rm.predict(predictor_test)
Predicted_Price
```

Out[38]:
```
array([[1494024.20183109],
       [  90165.42939004],
       [2154199.85700686],
       ...,
       [1143306.74101987],
       [ 466825.087058  ],
       [ 657391.91892199]])
```

In [39]:
```
rm.score(predictor_test, target_test)
```

Out[39]: 0.6791636224336604

In [40]:
```
rm.score(predictor_train, target_train)
```

Out[40]: 0.6785830616655293

In [42]:
```
from sklearn.linear_model import Ridge
from sklearn.linear_model import Lasso
```

In [47]:
```
ridge_reg= Ridge(alpha=50)
ridge_reg.fit(predictor_train, target_train)
```

Out[47]: Ridge(alpha=50)

In [48]:
```
ridge_reg.score(predictor_test, target_test)
```

Out[48]: 0.6716963618076129

In [49]:
```
ridge_reg.score(predictor_train,target_train)
```

Out[49]: 0.6629977947456143

In [51]:
```
import warnings
warnings.filterwarnings('ignore')
```

In [52]:
```
lasso_reg= Lasso(alpha=50)
lasso_reg.fit(predictor_train, target_train)
```

Out[52]: Lasso(alpha=50)

In [53]:
```
lasso_reg.score(predictor_test, target_test)
```

Out[53]: 0.6811869272179951

In [54]:
```
lasso_reg.score(predictor_train,target_train)
```

Out[54]: 0.6741500974953867

5. Extract slope and intercept value from the model

In [55]:
```
print('Slope is:' ,rm.coef_)
print('Intercept is:', rm.intercept_)
```

```
Slope is: [[ 2.68003373e+05 -4.93117297e+04 -8.83708650e+04  1.27907805e+05
   4.20480554e+04  2.31334303e+00  3.65060295e+01  5.92224869e-01
   1.82558558e+05 -5.45960590e+04 -2.13645152e+04  2.74034622e+05
   7.42045136e+04  2.05901629e+05  1.59108788e+05 -9.15459105e+04
  -1.67416178e+05 -3.17204812e+04  9.87600738e+04 -1.16524895e+05
  -5.63662010e+04 -5.21320301e+03 -4.08503517e+04 -2.63204674e+05
  -1.33093000e+05 -2.27698641e+05  5.34175704e+04 -1.29964830e+05
   1.39333702e+05  2.71463177e+04  9.40545329e+04  2.15540157e+04
   1.71363354e-06  5.08935061e+04  1.83444371e+05 -3.80426364e+05
   7.56436761e+04 -2.26136409e+04  2.30935339e+04  2.66193443e+05
   1.53466019e+05 -1.34260050e+05 -8.59536545e+04  3.63518901e+05
```

```
 1.90189509e+05  2.81964028e+05  1.85929321e+05 -1.29925076e+05
-1.01703979e+05  3.32423373e+05 -1.53241803e+05 -5.65811752e+04
-6.33841811e+04  3.67792939e+04  3.95620706e+04  2.25731025e+04
-2.37432042e+04 -2.28233694e-06 -1.89376675e+05  1.31197113e+05
-1.63207487e+05  5.81304532e+04 -2.11360509e+05 -2.83869490e+05
-7.12037987e+04 -4.03423836e+04 -8.94200519e+04  6.57257399e+04
-1.24475177e+05  5.61138920e+05  8.12776174e+04  2.65516941e+05
-9.38113080e+03 -1.89585031e+05  3.98824898e+05 -9.64388052e+04
-2.17463638e+05  9.30874414e+04  2.30774643e+03  1.50222899e+04
-7.65250597e+04  3.31570823e+05 -5.57826855e+04 -1.96132736e+05
-2.03673326e+05 -4.01955262e+05 -8.88227946e+04 -2.44517993e+05
 1.21153014e+05  2.09586570e+05 -1.10707879e+05 -2.23976951e+05
-1.05097842e+05  3.07772295e+04 -8.86219395e+04  1.63695714e+04
 5.90831525e+04  1.63559367e+04 -5.75330351e+04 -1.25072371e+04
-1.21817540e+04  7.63404560e+04 -1.56828612e+05 -4.09769834e+04
 9.63257942e+04 -3.01267171e+05  1.02576258e+05 -7.87376008e+04
 5.07154614e+04  2.86978241e+05 -2.88897859e+04  1.86610271e+04
-1.78191607e+05 -1.76824361e+05  3.00213318e+05 -1.77947065e+05
-3.16089958e+05  7.05498263e+03  3.79238634e+04  1.18367892e+05
-2.91960586e+05 -1.61996381e+05  3.20720296e+05  3.07994311e+05
 2.40602776e+05  9.80429593e+04 -1.29639860e+04 -2.26875707e+05
 1.59753010e+04 -1.43219988e+05 -2.41764076e+05 -1.41477923e+04
 1.16893646e+05  5.05046514e+04  2.44457350e+04 -2.25282717e+05
 1.14332875e+05  1.11010852e+05 -6.50660555e+04  1.94684603e+05
-1.66235259e-06  1.67970104e+05  2.40604167e+05 -1.03714293e+05
-4.59658500e+04 -7.95129892e+04  2.68235382e+05 -8.09151431e+04
 3.99900112e+05 -2.03717422e+05  1.44987813e+05  1.99716118e+05
-2.94591735e+05 -3.66282351e+04 -9.47039369e+04  2.92018485e+05
-5.19086983e+04 -2.32602430e+05  5.99494665e+04  5.88374173e+04
-1.32444748e+05  1.65692724e+05  1.56420817e+05 -2.47463684e+05
-2.86799275e+04 -2.74510798e+05 -1.52802134e+05  3.30487681e+05
-1.09001493e+05  4.33882971e+04 -1.54008268e+05 -3.99407871e+05
-4.91622092e+05 -9.54503705e+04 -1.55956589e+04  1.42424266e+03
 7.28792496e+04 -1.31628992e+05 -2.00799606e+05 -4.65776863e+04
 4.34828481e+05 -1.37251303e+05 -3.49659240e-06 -1.34904457e+05
 4.25229766e+04 -4.12693857e+04  6.04990216e+04  1.22215360e+05
-2.08027276e+04 -8.77978081e+04 -7.77184948e+04 -1.68021573e+05
 4.08806119e+04  8.89348036e+04 -8.43408539e+04 -7.81190679e+04
 5.17194304e+04 -5.74498889e+04 -7.28095754e+04 -5.68635646e+04
-1.13901569e+05  9.24192551e+04 -1.48548931e+03 -1.64559682e+05
 8.10438730e+03  9.48846149e+04  7.15912808e+04  6.00748176e+04
-7.61544722e+04  3.93990223e+05  1.53332553e+04 -2.41364227e+04
 2.30790712e+05 -2.65080641e+04  2.71678263e+04 -1.51971746e+04
 7.40015162e+04 -9.47150339e+04  1.04856991e+05  1.46120272e+05
-2.87494631e+03  5.56117837e+05 -1.18644356e+05  1.02091224e+05
 1.47806958e+04  5.53721205e+04 -8.93849379e+04 -3.08977108e+05
 5.93259507e+04 -1.15182447e+05 -1.66962641e+05  1.51979002e+05
 2.05907412e+04  2.06346789e+05 -4.17941654e+04  4.26137653e+02
 8.28033967e+04 -1.69524661e+05 -5.75241524e+04  8.07260521e+04
-1.03200296e+03 -3.99385661e+04  5.21796767e+05  1.57314746e+05
 9.25980163e+04  1.57322139e+04  7.14933452e+04  1.06326942e+04
-5.27815014e+04 -3.92057023e+05  1.33851307e+05  9.21505721e+04
 2.49649496e+05  2.01525957e+05  3.65375916e+05 -8.79832179e+03
 1.73585615e+05  2.53999673e+05  2.60489317e+05 -1.88358437e+05
-7.57151334e+04 -3.11002936e+05 -6.41256087e+04  5.25839437e+05
 6.79468951e+04 -5.72862829e+04  4.27959722e+04  2.56230648e+05
 4.45872851e+04  4.35977011e+04 -1.11874135e+05 -1.41311708e+05
 4.25079262e+05 -2.35609947e+05 -1.83463278e+05 -5.61343303e+04
-1.75217660e+05  1.85272196e+05  8.47247041e+04 -1.58773490e+05
 1.71845507e+05  3.42744243e+04  3.58178589e+04  3.26950642e+05
-2.15570754e+05 -8.76864429e+04 -1.95954472e+05 -5.81548332e+04
-9.46792877e+04 -3.83566131e+05 -6.73688540e+04 -1.15128959e+05
 3.32179997e+04  6.53816059e+04 -3.25526437e+05 -1.44061703e+05
 7.11835207e+04 -1.82060696e+05  4.97889556e+05  6.36556655e+04
-6.22904733e+04 -1.35479654e+05  4.29436725e+03  1.10461888e+05
-5.46687322e+04 -9.47554869e+04 -3.96427021e+04  6.05839887e-07
 1.43430320e+05 -1.67023855e+05  2.24307119e+05 -3.88727990e+04
 5.50124745e+04  1.35309199e+05 -3.39054900e+04  6.82278310e+03
 5.90387638e+04  1.30443510e+05 -6.93747122e+04 -2.43309859e+04
-3.77260391e+05  1.98866356e+05  1.93825330e+05 -2.81240204e+05
-2.62765985e+05  1.88276207e+05  4.16373950e-07 -4.16614846e+03
 6.05271921e+04 -1.16000088e+05 -2.66042564e+05 -1.58343927e+05
-6.26364679e+04 -6.75395834e+04  5.34871185e+04 -6.16159986e+04
-1.70514492e+05  7.83999000e+04  4.03773761e+05 -8.94657071e+04
 2.93011248e+05  2.10830630e+04 -1.58481282e+05  1.67969784e+05
 3.73337416e+04  3.67820525e+05 -1.11427241e+05 -1.29686669e-06
 4.77152114e+04 -1.54599258e+04 -1.05512199e+05 -2.75849377e+05
-4.78413673e+05  7.62100991e+04  7.82443987e+04  3.93689383e+04
 1.87416836e+04 -3.53213276e+04  2.15054356e+05  6.19883487e+05
 3.93712315e+05  1.30228000e-06  5.41210635e+05  1.20735962e+05
-6.84344706e+04 -1.73991346e+04  1.62669224e+05 -1.54361644e+05
 1.84253012e+05  2.39797661e+05  6.97482408e+03 -7.34328023e+04
 1.89448373e+04  4.35703645e+05  2.05520664e+05  6.22178465e+04
```

```
 3.52021602e+05  1.87080790e+05  1.18294807e+05  4.53652071e+05
-6.68181341e+03  5.59501782e+04 -6.52462661e+02  1.07025362e+05
 2.53995217e+05  1.25122772e+05 -1.61674061e+05  6.76559558e+03
 1.17511198e+05  2.36165955e+05  1.17697971e+05  8.74790060e+05
 1.89909150e+05 -1.65179435e+04 -5.81075466e+04 -1.25018320e+05
-3.74397187e+04  1.14123441e+05  1.10840326e+05  1.45111958e+05
 3.95721923e+04  1.85832022e+05 -1.25751174e+05  1.91634357e+04
 1.25689146e+05 -2.49499852e+04  3.68493423e+05 -5.06697688e-07
-8.00676041e+04  1.21734607e+05 -1.08356471e-06  5.13357672e+05
 1.55127814e+05  3.44077125e-06  1.87287270e+05  3.18527838e+05
 1.72804930e+05 -2.26039991e+05  6.76568245e+04 -2.29925886e+05
 1.63490424e+05 -1.87393164e+05  1.25157233e+05 -8.65612645e+03
 3.27483909e+05 -3.24715669e+04  2.62032582e+05  1.83880359e+05
 2.14255882e+05  1.66131675e+05  1.74180605e-06  7.69901209e+03
-1.22358387e+05  6.53185768e+04  6.06946180e+04  8.52677520e+04
 7.57883528e+04  5.43276680e+04  1.46939213e+06  2.41422791e+05
-8.81399105e+04  5.81033886e+04  4.09914159e+03 -4.89924449e+05
-4.32479797e+04 -1.07315431e+05 -4.43386507e+04  1.10024198e+05
 1.34229217e+05  4.05393305e+05  4.62911395e+04  1.10936617e+05
 1.89280433e+05  2.17131348e+05  1.93516438e+05  1.18738353e+05
-4.13759147e+04 -3.75957461e-06  3.29143950e-06  1.62796821e+06
 9.94861290e+04  1.55212903e+05  6.71608173e+04  1.06746874e+05
-7.92508675e+04 -2.11999920e+05 -2.09710628e+05  5.62487570e+04
-4.04668320e+05 -8.96827672e+04  1.41706281e+05  1.71221718e+05
 9.65592883e+04  5.91705321e+05  1.17043853e+05  1.00969128e+06
-9.77237795e+04 -1.57074091e+05  1.14018610e+05 -4.28542114e+04
 2.59245697e+04 -9.47589661e+03  6.87091778e+04 -1.08448904e+05
 2.91292262e+05 -5.94874186e+03 -1.00717738e+05  2.79276513e+05
 1.90273624e+05  2.01067847e+05  7.29729038e+04 -3.46770935e+03
 2.75631395e+04  3.09020116e+04  1.86640665e+05 -4.84928023e-07
 1.48106973e+05  1.29052887e+05  1.93090726e+05  1.10878195e+05
 1.52518762e+05 -4.66729274e+05  3.06019892e+04  2.50792206e+05
 1.60772087e+05  5.46313743e+04 -2.80073793e+05  5.71403697e+05
 2.43825396e+05  1.42276195e+05  4.13013994e+05  4.29440846e+04
 1.38820342e+05 -3.56022872e+04  1.09680169e+05  1.05615376e+05
 7.27465435e+05 -1.38318126e+05  2.09682407e+05  2.23207900e+04
-3.22712963e+04  1.48658251e+05  4.86948810e+04  2.03235410e+05
 2.68668893e+05  1.43693481e+05  3.48120582e+05 -5.11242064e+05
 5.35563963e+04  2.41779734e+05  1.82458845e+04  2.86596226e+04
-3.12532463e+04 -1.13714793e+05  7.23391592e+04  3.30487681e+05
 4.33392520e+05 -7.50813417e+04  1.13888601e+05  3.73516077e+05
 2.33445388e+05  1.88095380e+04  1.60448807e+05 -1.66100574e+05
 5.52390702e-07  4.32381893e+04  1.84104628e+05  1.07995872e+05
 2.68802978e-07  2.36663993e+05  1.49788018e+05  2.04402121e+04
 2.00753682e+05  1.15571536e+05  7.71474272e+05  2.56958526e+05
 9.99433517e+04 -2.01797419e+05  1.42412830e+05  1.53510163e+05
 1.86427908e+05  3.80632148e+04  1.12502006e+05  8.58487840e+04
 3.07048413e+05  8.79784778e+05  6.79310769e+04 -1.52868832e+05
 2.45548372e+05  9.57495937e+04  3.44570177e+05  1.08619732e+05
-1.49350941e+05  1.94264679e+05  3.75368523e+05  1.30466535e+05
 5.14424567e+04  1.39031444e+05  7.31240841e+04  1.05738785e+05
-2.49532722e+05  9.89785428e+04  5.44393137e+04  2.45022285e+04
 9.34403943e-07  1.04274760e+05  1.75662861e+05  3.17917684e+05
 2.48628836e+05  3.16985719e+05  6.00169186e+05  1.71983300e+04
 1.36615625e+05  1.33243289e+05  1.31536526e+05 -4.21648101e+04
 2.71106694e+05  9.84636629e+04 -1.34775973e+04  2.59855273e+05
-5.12446662e+05 -3.40369920e+05  3.77816551e+04  1.43722759e+05
 5.26573127e+04  1.41462369e+05  4.74818987e+04  3.18336248e+05
 1.09599646e+05  9.31447561e+04 -8.22797676e+04  7.10763741e+04
 6.13766397e+04 -4.47440664e+05  5.04152925e+05  1.52719365e+05
 1.07392637e+05  1.74957169e+05  1.09808869e+05  1.01797982e+05
 1.18492547e+05  8.30782468e+04 -2.17758743e+04  5.82081454e+04
-1.28544449e+05 -7.84872100e-07 -4.10936014e+05  1.98580937e+05
 2.14988062e+05  1.59871758e+05 -5.18182936e+05  2.28960503e+05
-3.37909975e+05  2.72810775e+04 -4.50953540e+05 -2.05269749e+05
 1.17824990e+05  8.31891396e+04 -3.94192855e+05  9.55699199e+05
 1.43599385e+05  2.16182369e+05  4.16020257e+04  7.76490197e-07
 9.07083002e+04 -2.56100765e+05  1.10213125e+05  9.35579237e+03
 1.68822635e+05  6.08060945e+05 -7.34124682e+04  1.93447369e+05
-7.42408822e+04 -3.33774079e+04 -4.41807023e+05 -4.96396681e+04
 1.80170010e+05  1.22014672e+05 -2.20394873e+05 -2.56317435e-07
 3.70165450e+05  1.98075939e+05  2.51929163e+04  7.12752119e+05
-9.56196745e+04  5.72367026e+03 -1.15394969e+04 -1.91968929e+05
 1.65121670e+05  1.81579916e+03  9.63787016e+05  7.38393414e+04
 1.23662148e+05  9.86881017e+05  2.22116996e+05 -2.54527133e+04
-2.45679068e+04  3.09852338e+05  6.73500046e+04  1.16751914e+05
 1.43686126e+05  2.25278431e+05  2.18434104e+04  3.11319276e+04
 1.00196983e+05  6.20992559e+03 -2.39642400e+05  1.12231704e+05
 0.00000000e+00  1.19643525e+05  7.35864564e+04  2.25831594e+05
 2.98511388e+05 -2.14330086e+04  1.95081656e+05 -3.90353976e+04
 0.00000000e+00  1.72985067e+05  0.00000000e+00  1.01571976e+05
 8.13380494e+04 -1.83449981e+05  1.73480202e+05  2.22375051e+05
-2.66098187e+04  4.31465606e+05  3.15866270e+05 -2.30492238e+05
```

```
   2.91174321e+05  2.05700584e+05 -4.97608758e+03  3.64257894e+05
   1.17667467e+05 -1.82236936e+04 -1.24322993e+05 -7.09972722e+04
   1.61799822e+05  7.46323934e+04  5.62607031e+05  7.98445559e+04
  -2.08468904e+05  1.32088119e+05 -6.81447128e+02 -1.29004719e+05
   1.98866356e+05  3.59912265e+04 -2.91476669e+04  1.04133032e+05
  -7.52647168e+04 -9.18837823e+04  7.63537658e+04  2.59580239e+05
   7.70865005e+04  4.87686058e+04 -1.84228503e+05 -4.46897267e+04
   6.78080012e+04  3.10012600e+05 -2.40955844e+05  3.37154925e+05
   1.71529387e+05 -9.14324275e+02 -1.89156990e+05  2.13886067e+05]]
Intercept is: [781959.19165927]
```

### 6. Display Mean Squared Error

In [56]:
```
mse = mean_squared_error(target_test,Predicted_Price)
print('mean squared error is: ', mse)
```

```
mean squared error is:  136132361896.98344
```

### 7. Display Mean Absolute Error

In [57]:
```
mae = mean_absolute_error(target_test,Predicted_Price)
print('mean absolute error is: ', mae)
```

```
mean absolute error is:  234209.03292221262
```

### 8. Display Root mean Squared error

In [58]:
```
rmse = np.sqrt(mean_squared_error(target_test,Predicted_Price))
print('root mean squared error is: ', rmse)
```

```
root mean squared error is:  368961.1929417286
```

### 9. Display R2 score

In [59]:
```
r2 = r2_score(target_test, Predicted_Price)
print('r2_score is: ', r2)
```

```
r2_score is:  0.6791636224336604
```