Name : SHAHNAWAZ ALAM

@Bharat intern

Domain : Machine Learning Intern

Task1 : House Price Prediction

Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import metrics

%matplotlib inline
```

Importing Dataset

In [2]:

```
print("Importing data...")
HouseDF = pd.read_csv(r"C:\Users\md naiyer azam\Desktop\USA_Housing.csv")
print("Sucessfully imported.")
```

```
Importing data...
Sucessfully imported.
```

In [3]:

```
HouseDF.head()   # it is used to give first five row
```

Out[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\nFPO AE 09386

In [4]:

```
HouseDF.tail()
```

Out[4]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
4995	60567.94414	7.830362	6.137356	3.46	22837.36103	1060193.786	USNS Williams\nFPO AP 30153-7653
4996	78491.27543	6.999135	6.576763	4.02	25616.11549	1482617.729	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.68689	7.250591	4.805081	2.13	33266.14549	1030729.583	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.33124	5.534388	7.130144	5.44	42625.62016	1198656.872	USS Wallace\nFPO AE 73316
4999	65510.58180	5.992305	6.792336	4.07	46501.28380	1298950.480	37778 George Ridges Apt. 509\nEast Holly, NV 2...

In [5]:

```
HouseDF.shape   ##to get no. of rows and column(rows,column)
```

Out[5]:

```
(5000, 7)
```

In [6]:

```
#info of data
HouseDF.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Avg. Area Income              5000 non-null   float64
 1   Avg. Area House Age           5000 non-null   float64
 2   Avg. Area Number of Rooms     5000 non-null   float64
 3   Avg. Area Number of Bedrooms  5000 non-null   float64
 4   Area Population               5000 non-null   float64
 5   Price                         5000 non-null   float64
 6   Address                       5000 non-null   object 
dtypes: float64(6), object(1)
memory usage: 273.6+ KB
```

In [7]:

```
HouseDF.describe()
```

Out[7]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
min	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
25%	61480.562390	5.322283	6.299250	3.140000	29403.928700	9.975771e+05
50%	68804.286405	5.970429	7.002902	4.050000	36199.406690	1.232669e+06
75%	75783.338665	6.650808	7.665871	4.490000	42861.290770	1.471210e+06
max	107701.748400	9.519088	10.759588	6.500000	69621.713380	2.469066e+06

In [8]:

```
HouseDF.columns # to find the no of columns
```

Out[8]:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

Exploratory Data Analysis for House Price Prediction

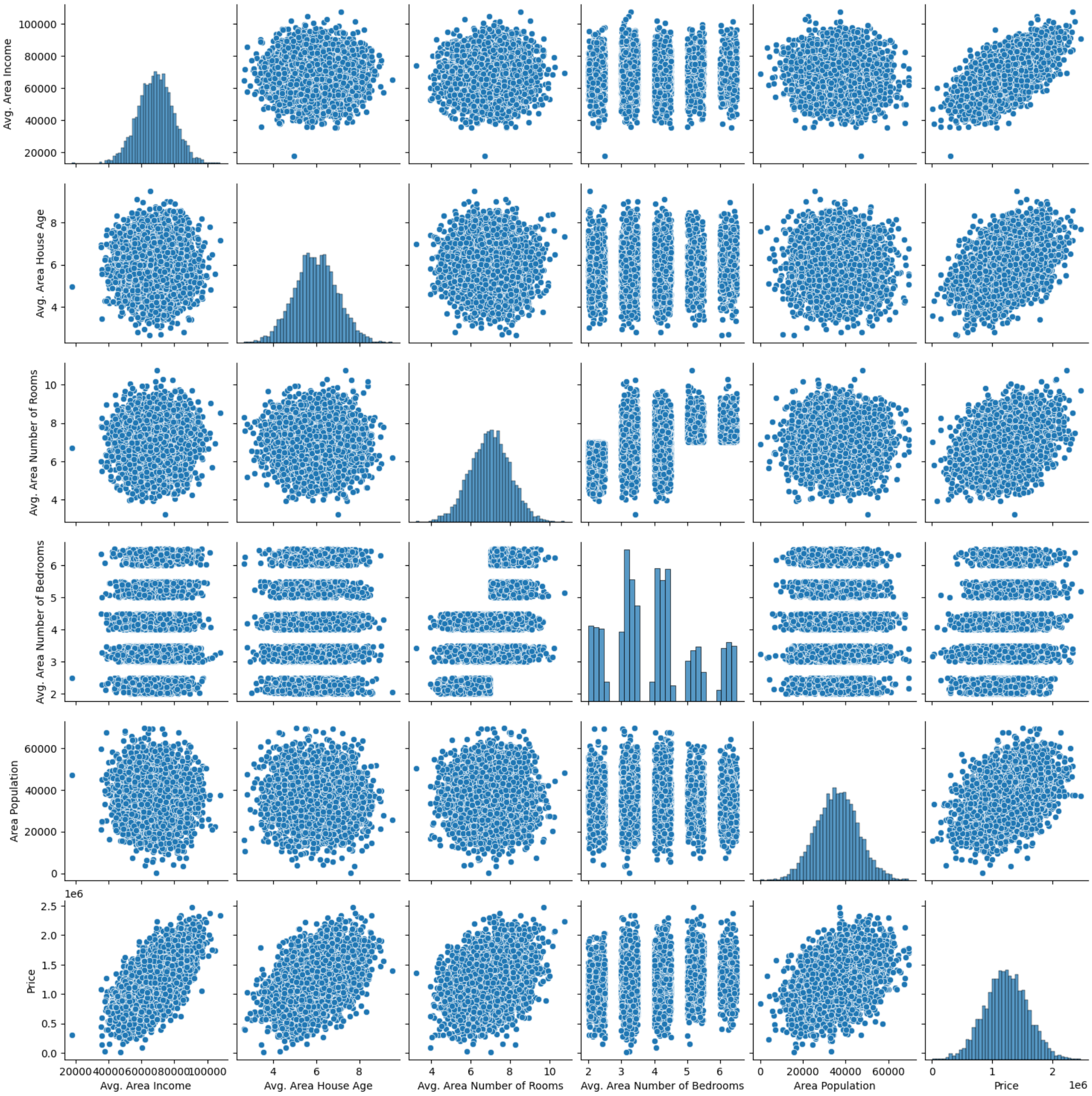
We will create some simple plot for visualizing the data.

In [9]:

```
sns.pairplot(HouseDF)  # to plot multiple pairwise bivariate distributions in a dataset
```

Out[9]:

<seaborn.axisgrid.PairGrid at 0x1c7febb5c00>



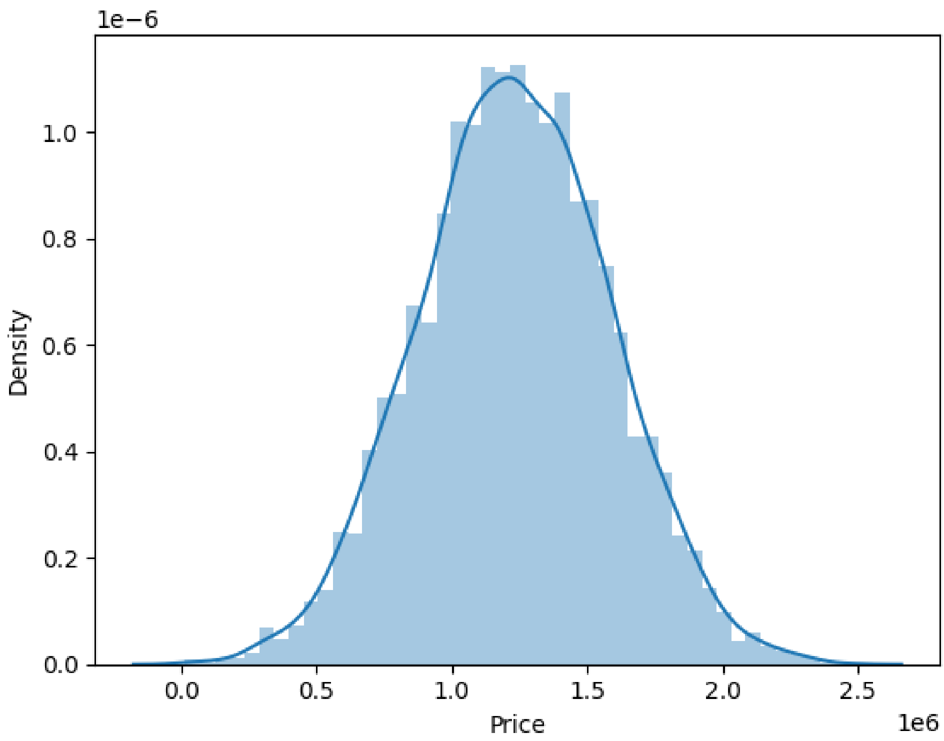
In [13]:

```
sns.distplot(HouseDF['Price'])
```

```
C:\Users\md naiyer azam\AppData\Local\Temp\ipykernel_17784\4158129596.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwa
skom/de44147ed2974457ad6372750bbe5751)

  sns.distplot(HouseDF['Price'])
```

Out[13]:

```
<AxesSubplot: xlabel='Price', ylabel='Density'>
```



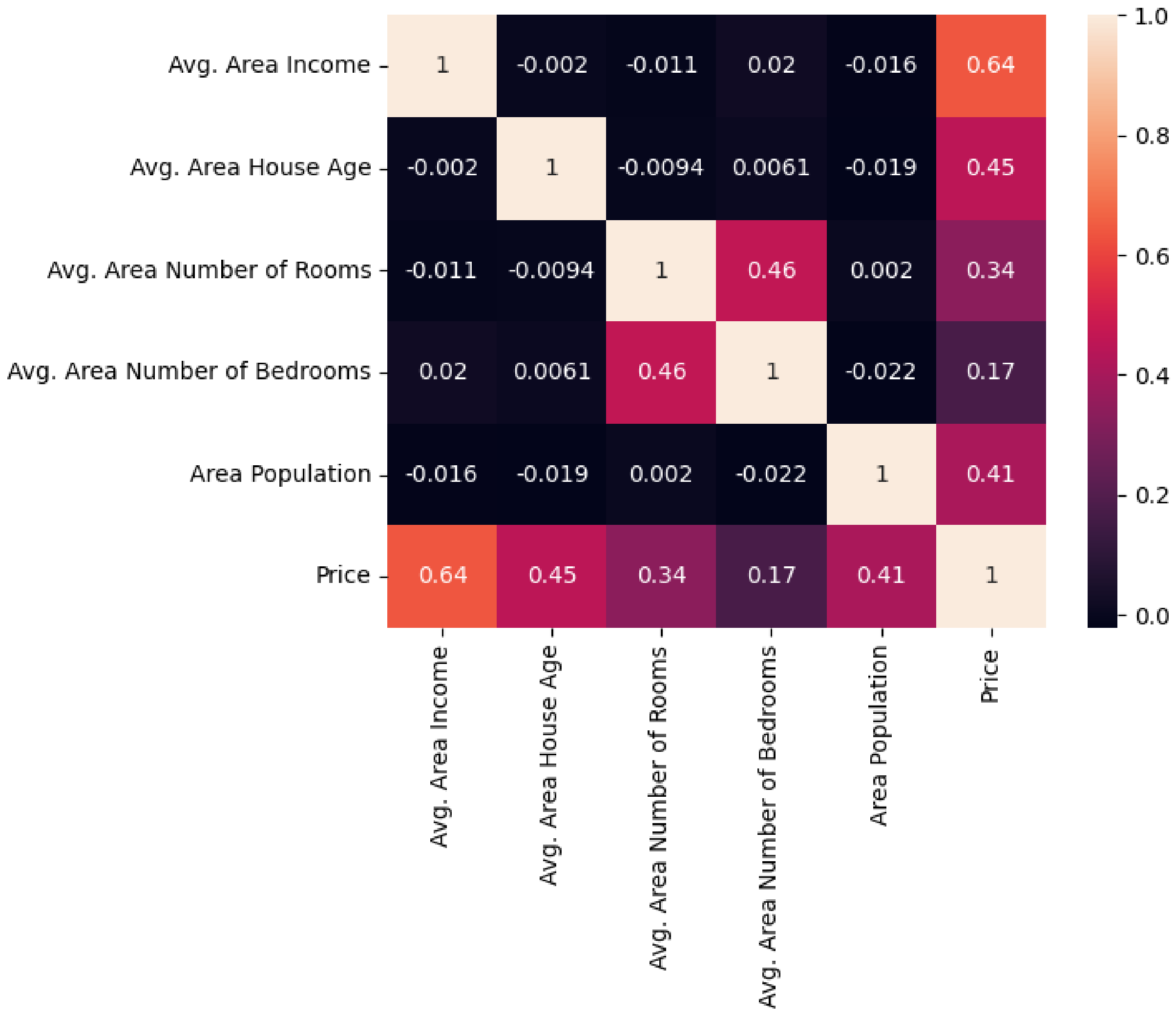
In [11]:

```
sns.heatmap(HouseDF.corr(), annot=True)
```

```
C:\Users\md naiyer azam\AppData\Local\Temp\ipykernel_17784\711344588.py:1: FutureWarning: The
default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will d
efault to False. Select only valid columns or specify the value of numeric_only to silence th
is warning.
  sns.heatmap(HouseDF.corr(), annot=True)
```

Out[11]:

<AxesSubplot: >



Get Data Ready For Training a Linear Regression Model

In [14]:

```
X = HouseDF[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
               'Avg. Area Number of Bedrooms', 'Area Population']]
```

In [16]:

```
y = HouseDF['Price']
```

Split Data into Train, Test

In [17]:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=2, random_state=101)
```

X_train and y_train contain data for the training model. X_test and y_test contain data for the testing model. X and y are features and target variable names.

Creating and Training the LinearRegression Model

We will import and create sklearn linearmodel LinearRegression object and fit the training dataset in it.

In [19]:

```
from sklearn.linear_model import LinearRegression
```

In [20]:

```
lm = LinearRegression()
```

In [21]:

```
lm.fit(X_train, y_train)
```

Out[21]:

```
▾ LinearRegression
LinearRegression()
```

LinearRegression Model Evaluation

Now let's evaluate the model by checking out its coefficients and how we can interpret them.

In [22]:

```
print(lm.intercept_)
```

```
-2637430.00820446
```

In [23]:

```
coeff_df = pd.DataFrame(lm.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[23]:

	Coefficient
Avg. Area Income	21.578668
Avg. Area House Age	165648.005908
Avg. Area Number of Rooms	120666.286155
Avg. Area Number of Bedrooms	1634.074656
Area Population	15.201783

Predictions from our Linear Regression Model

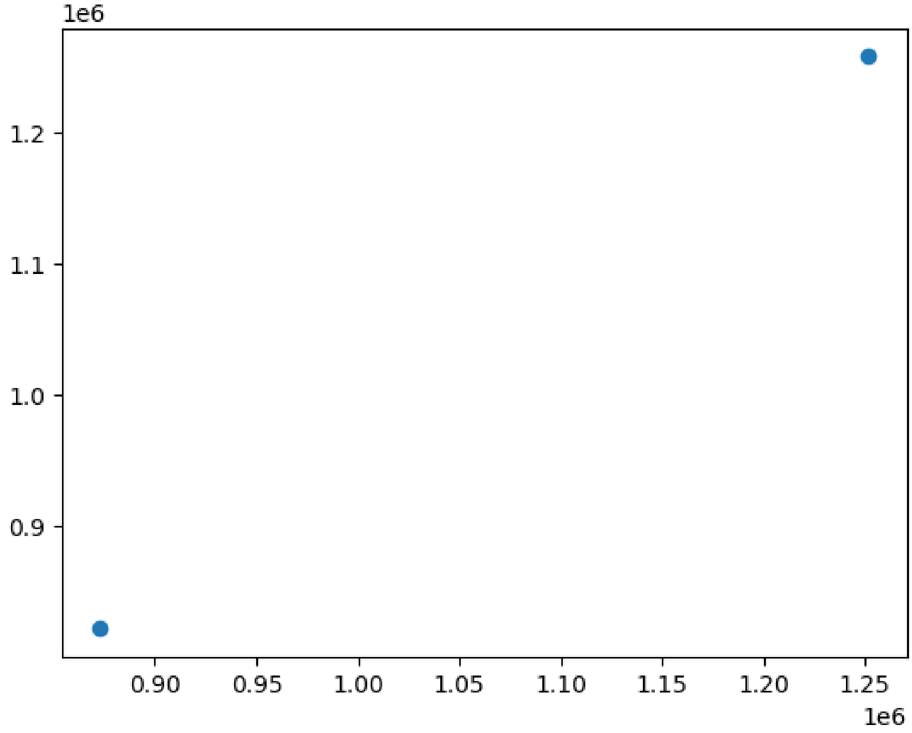
Let's find out the predictions of our test set and see how well it perform.

In [27]:

```
predictions = lm.predict(X_test)
plt.scatter(y_test,predictions)
```

Out[27]:

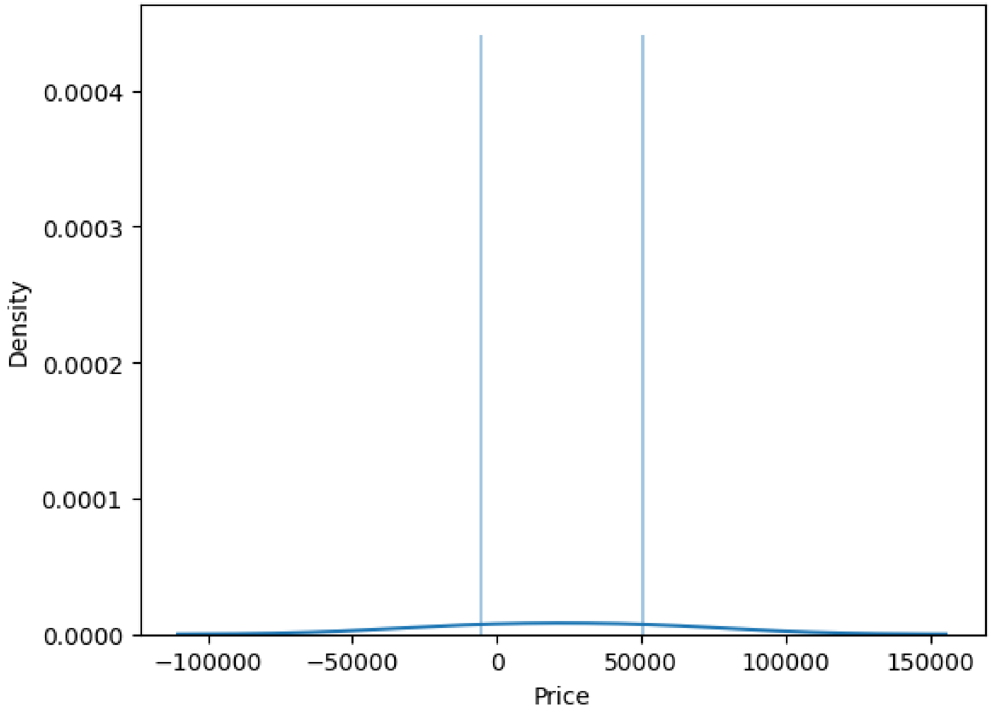
```
<matplotlib.collections.PathCollection at 0x1c7890a96f0>
```



In [26]:

```
sns.distplot((y_test-predictions),bins=50);
```

```
C:\Users\md naiyer azam\AppData\Local\Temp\ipykernel_17784\1326397652.py:1: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751 (https://gist.github.com/mwa
skom/de44147ed2974457ad6372750bbe5751)

  sns.distplot((y_test-predictions),bins=50);
```



Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparing these metrics:

MAE is the easiest to understand because it’s the average error.
MSE is more popular than MAE because MSE “punishes” larger errors, which tends to be useful in the real world.
RMSE is even more popular than MSE because RMSE is interpretable in the “y” units.
All of these are loss functions because we want to minimize them.

In [28]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 28317.853409252595
MSE: 1310436393.342802
RMSE: 36199.95018425857
```

Conclusion

```
We have created a Linear Regression Model which we help the real state agent for estimating the house
price.
```

End of the code

****Thank You***