evjt38y2u

July 31, 2023

1 Problem Statement

A real estate agent want help to predict the house price for regions in USA.He gave us the dataset to work on to use Linear Regression model.Create a model that helps him to estimate of what the house would sell for.

```
[ ]: import numpy as np
     import pandas as pd
     import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]: df=pd.read_csv("/content/10_USA_Housing.csv")
     df
```

```
[ ]:       Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
     0         79545.458574             5.682861                   7.009188
     1         79248.642455             6.002900                   6.730821
     2         61287.067179             5.865890                   8.512727
     3         63345.240046             7.188236                   5.586729
     4         59982.197226             5.040555                   7.839388
     …                  …                  …                         …
     4995      60567.944140             7.830362                   6.137356
     4996      78491.275435             6.999135                   6.576763
     4997      63390.686886             7.250591                   4.805081
     4998      68001.331235             5.534388                   7.130144
     4999      65510.581804             5.992305                   6.792336

           Avg. Area Number of Bedrooms  Area Population         Price  \
     0                             4.09     23086.800503  1.059034e+06
     1                             3.09     40173.072174  1.505891e+06
     2                             5.13     36882.159400  1.058988e+06
     3                             3.26     34310.242831  1.260617e+06
     4                             4.23     26354.109472  6.309435e+05
     …                                …              …            …
     4995                          3.46     22837.361035  1.060194e+06
     4996                          4.02     25616.115489  1.482618e+06
     4997                          2.13     33266.145490  1.030730e+06
     4998                          5.44     42625.620156  1.198657e+06
```

```
4999                          4.07     46501.283803  1.298950e+06

                                                Address
0     208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1     188 Johnson Views Suite 079\nLake Kathleen, CA…
2     9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3                             USS Barnett\nFPO AP 44820
4                            USNS Raymond\nFPO AE 09386
…                                                   …
4995                      USNS Williams\nFPO AP 30153-7653
4996                 PSC 9258, Box 8489\nAPO AA 42991-3352
4997  4215 Tracy Garden Suite 076\nJoshualand, VA 01…
4998                          USS Wallace\nFPO AE 73316
4999  37778 George Ridges Apt. 509\nEast Holly, NV 2…

[5000 rows x 7 columns]
```

[ ]:
```
df.head()
```

[ ]:
```
   Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
0      79545.458574             5.682861                   7.009188
1      79248.642455             6.002900                   6.730821
2      61287.067179             5.865890                   8.512727
3      63345.240046             7.188236                   5.586729
4      59982.197226             5.040555                   7.839388

   Avg. Area Number of Bedrooms  Area Population         Price  \
0                          4.09     23086.800503  1.059034e+06
1                          3.09     40173.072174  1.505891e+06
2                          5.13     36882.159400  1.058988e+06
3                          3.26     34310.242831  1.260617e+06
4                          4.23     26354.109472  6.309435e+05

                                             Address
0  208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1  188 Johnson Views Suite 079\nLake Kathleen, CA…
2  9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3                          USS Barnett\nFPO AP 44820
4                         USNS Raymond\nFPO AE 09386
```

2 Data Cleaning and Data Preprocessing

[ ]:
```
df.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
```

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

[ ]:
```
       Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
count       5000.000000          5000.000000                5000.000000
mean       68583.108984             5.977222                   6.987792
std        10657.991214             0.991456                   1.005833
min        17796.631190             2.644304                   3.236194
25%        61480.562388             5.322283                   6.299250
50%        68804.286404             5.970429                   7.002902
75%        75783.338666             6.650808                   7.665871
max       107701.748378             9.519088                  10.759588

       Avg. Area Number of Bedrooms  Area Population         Price
count                   5000.000000      5000.000000  5.000000e+03
mean                       3.981330     36163.516039  1.232073e+06
std                        1.234137      9925.650114  3.531176e+05
min                        2.000000       172.610686  1.593866e+04
25%                        3.140000     29403.928702  9.975771e+05
50%                        4.050000     36199.406689  1.232669e+06
75%                        4.490000     42861.290769  1.471210e+06
max                        6.500000     69621.713378  2.469066e+06
```

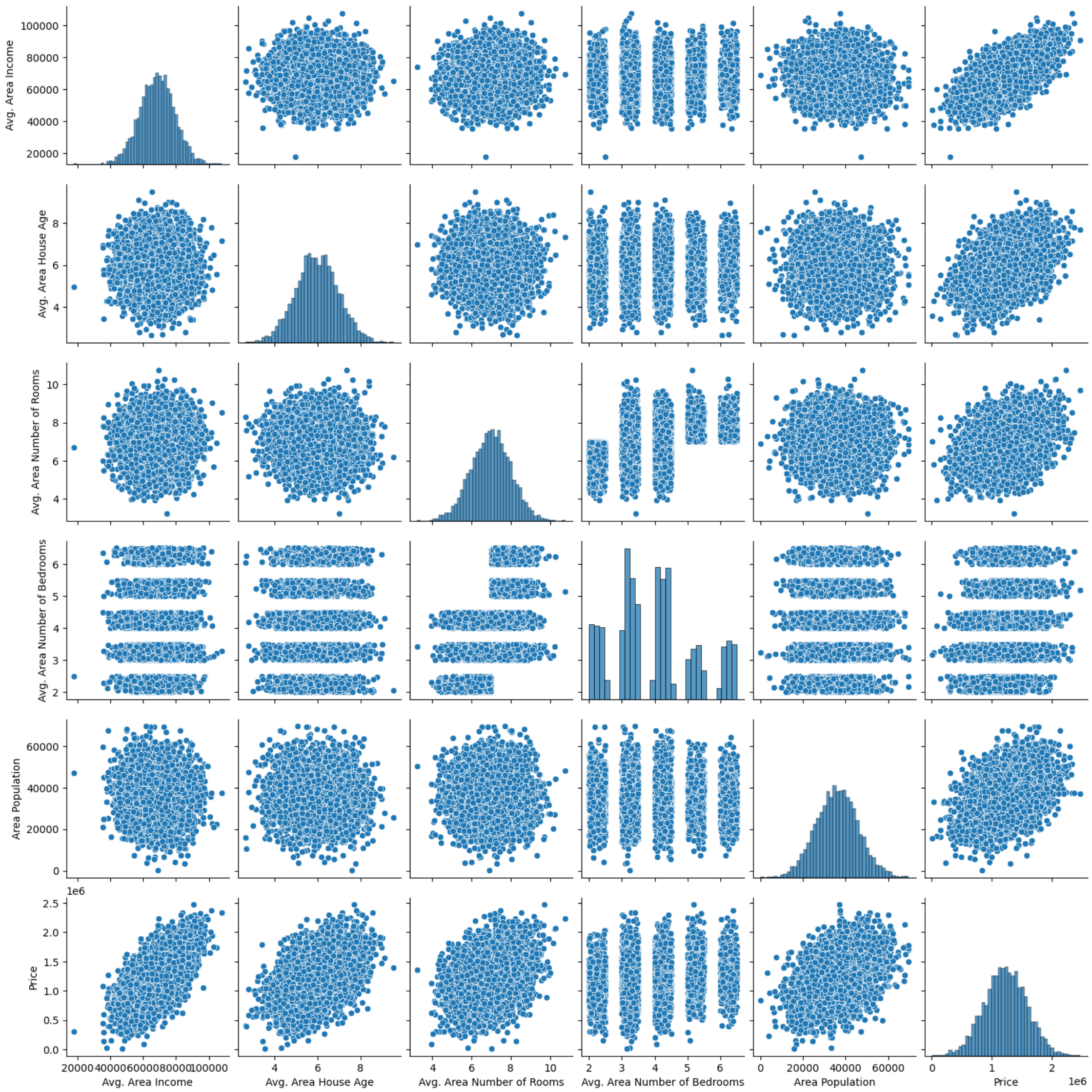
[ ]:
```
df.columns
```

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

3 EDA and Visualization

[ ]:
```
sns.pairplot(df)
```

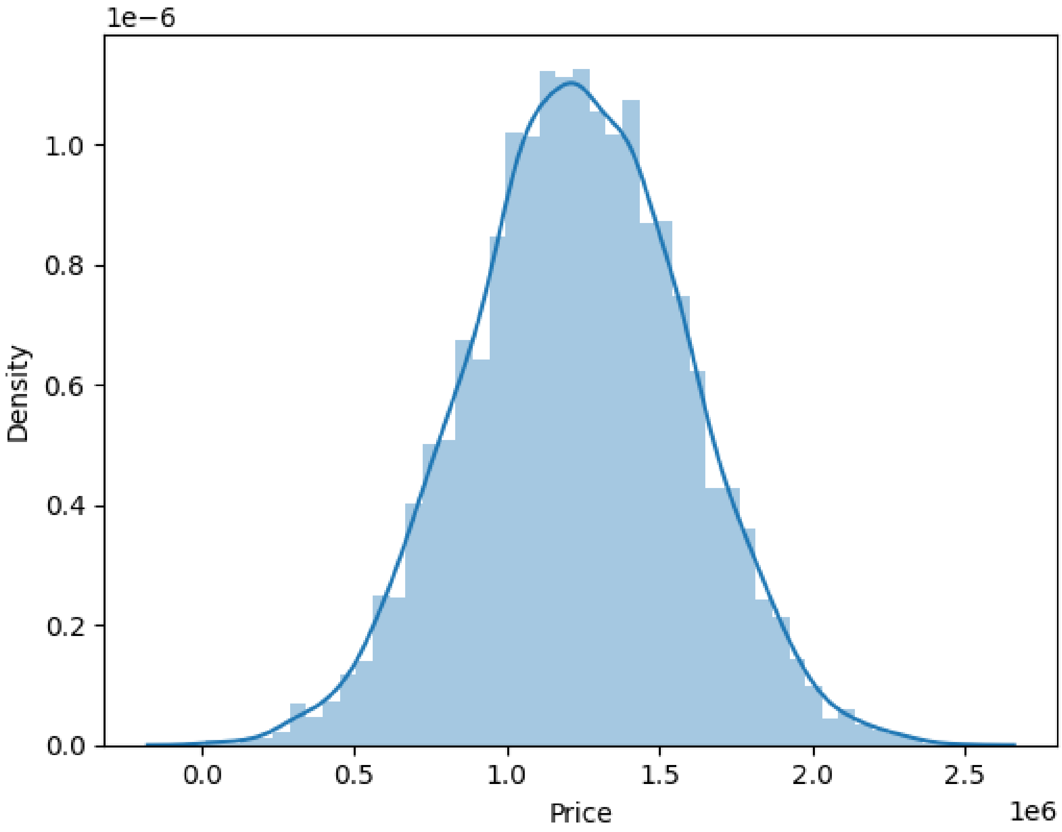
[ ]:
```
<seaborn.axisgrid.PairGrid at 0x7c2c00eca7d0>
```



```
[ ]: sns.distplot(df['Price'])
```

```
<ipython-input-8-87e11caeb2c4>: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

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

```
[ ]: <Axes: xlabel='Price', ylabel='Density'>
```



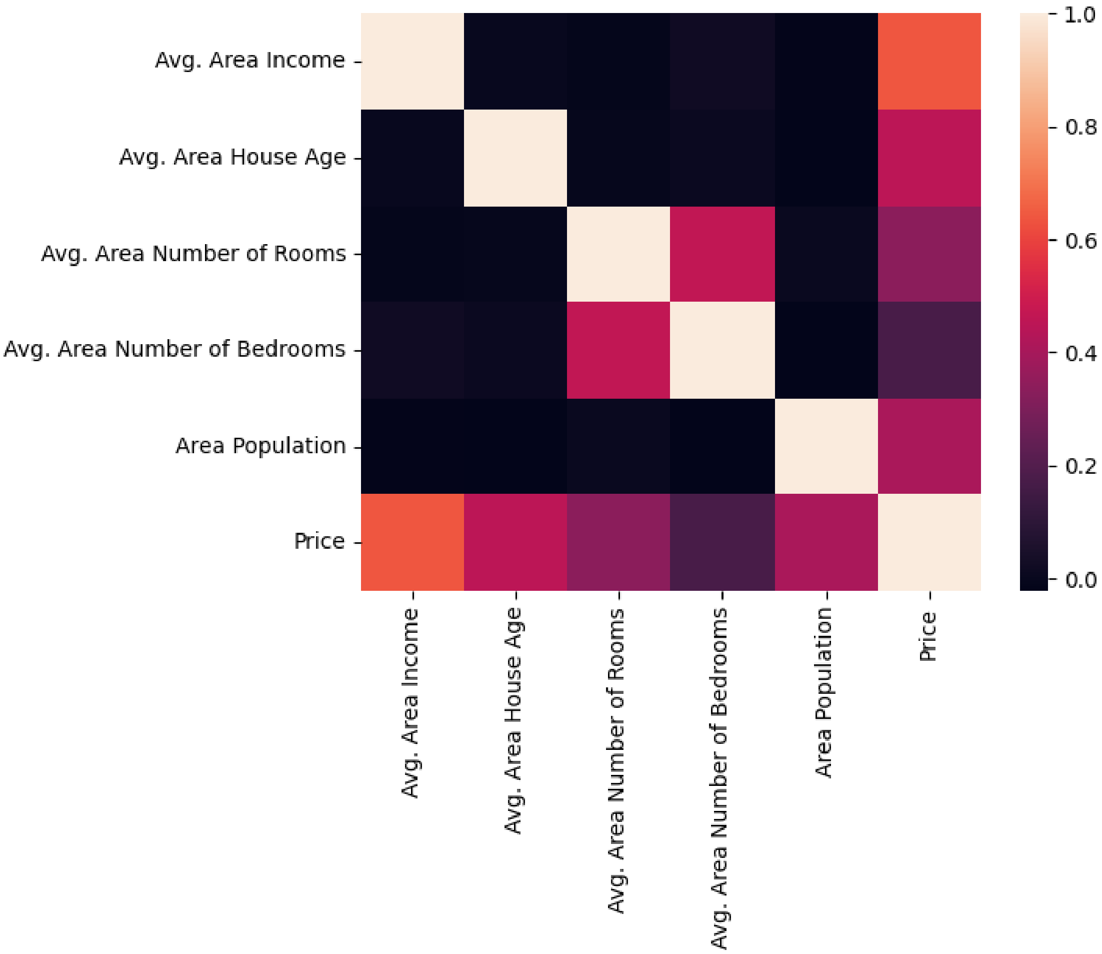
```
[ ]: df1=df[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
            'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
     df1
```

```
[ ]:       Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
     0         79545.458574             5.682861                   7.009188
     1         79248.642455             6.002900                   6.730821
     2         61287.067179             5.865890                   8.512727
     3         63345.240046             7.188236                   5.586729
     4         59982.197226             5.040555                   7.839388
     …                  …                  …                          …
     4995      60567.944140             7.830362                   6.137356
     4996      78491.275435             6.999135                   6.576763
     4997      63390.686886             7.250591                   4.805081
     4998      68001.331235             5.534388                   7.130144
     4999      65510.581804             5.992305                   6.792336

           Avg. Area Number of Bedrooms  Area Population         Price
```

```
0                              4.09     23086.800503  1.059034e+06
1                              3.09     40173.072174  1.505891e+06
2                              5.13     36882.159400  1.058988e+06
3                              3.26     34310.242831  1.260617e+06
4                              4.23     26354.109472  6.309435e+05
…                             …            …             …
4995                           3.46     22837.361035  1.060194e+06
4996                           4.02     25616.115489  1.482618e+06
4997                           2.13     33266.145490  1.030730e+06
4998                           5.44     42625.620156  1.198657e+06
4999                           4.07     46501.283803  1.298950e+06

[5000 rows x 6 columns]
```

[ ]:
```
sns.heatmap(df1.corr())
```

[ ]:
```
<Axes: >
```



To Train the Model -Model Building

We are going to train Linear Regression model;We need to spilt out data into two variables x and y where x is independent variable (input) and y is dependent variable on x(output) we could ignore address column as it is not requiired for our model

[ ]:
```
x=df1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
y=df1['Price']
```

[ ]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
```

[ ]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
print(lr.intercept_)
```

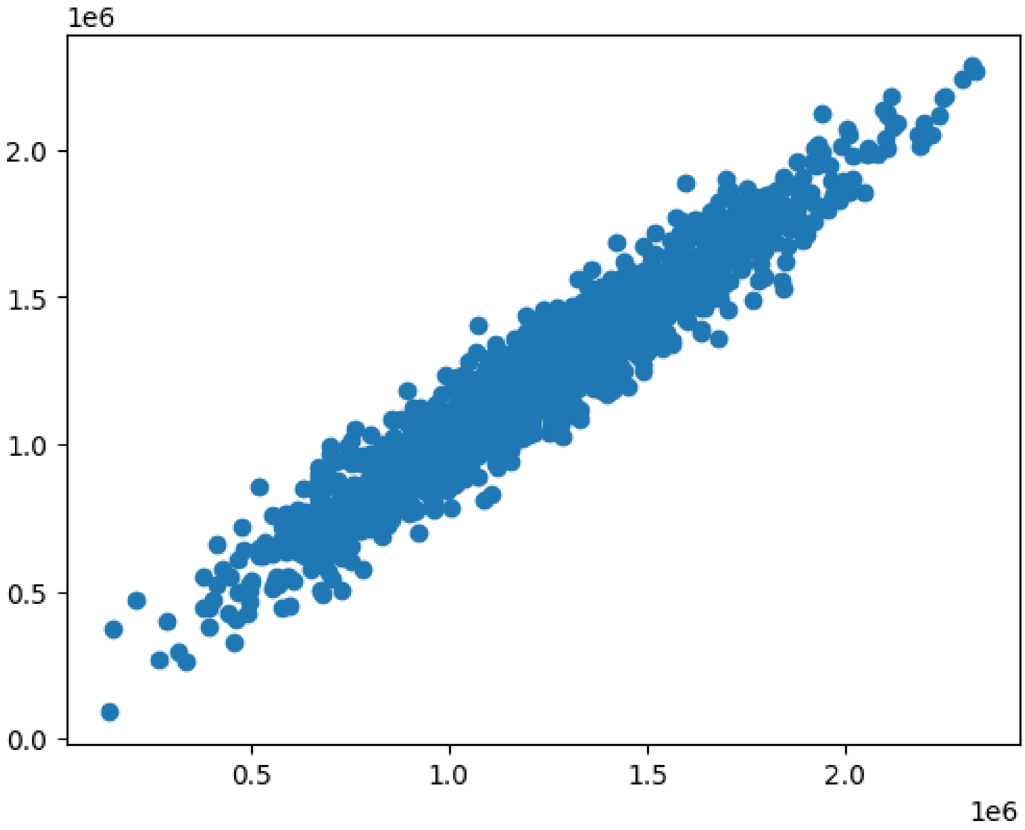
```
-2626342.537127545
```

[ ]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

[ ]:
```
                                Co-efficient
Avg. Area Income                   21.453704
Avg. Area House Age            164386.877564
Avg. Area Number of Rooms      121475.191879
Avg. Area Number of Bedrooms     1515.773131
Area Population                    15.196056
```

[ ]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x7c2bf8444550>
```



```
[ ]: lr.score(x_test,y_test)
```

```
[ ]: 0.9211174853391793
```

```
[ ]: lr.score(x_train,y_train)
```

```
[ ]: 0.9164618662113031
```

```
[ ]: from sklearn.linear_model import Ridge,Lasso
```

```
[ ]: rr=Ridge(alpha=10)
     rr.fit(x_train,y_train)
```

```
[ ]: Ridge(alpha=10)
```

```
[ ]: rr.score(x_test,y_test)
```

```
[ ]: 0.9210985679823257
```

```
[ ]: rr.score(x_train,y_train)
[ ]: 0.9164587767820217
```

```
[ ]: la=Lasso(alpha=10)
     la.fit(x_train,y_train)
```

```
[ ]: Lasso(alpha=10)
```

```
[ ]: la.score(x_test,y_test)
```

```
[ ]: 0.9211171631230292
```

```
[ ]: la.score(x_train,y_train)
```

```
[ ]: 0.9164618643978666
```

```
[ ]: from sklearn.linear_model import ElasticNet
     en=ElasticNet()
     en.fit(x_train,y_train)
```

```
[ ]: ElasticNet()
```

```
[ ]: en.coef_
```

```
[ ]: array([2.12263290e+01, 1.08602254e+05, 7.63297103e+04, 1.33174076e+04,
            1.51002255e+01])
```

```
[ ]: en.intercept_
```

```
[ ]: -2005738.7828657713
```

```
[ ]: prediction = en.predict(x_test)
     prediction
```

```
[ ]: array([1307199.01562841, 1064281.71599115, 1367394.6321102 , …,
            1169293.52382855, 1386953.32730034, 1316036.19617374])
```

```
[ ]: en.score(x_test,y_test)
```

```
[ ]: 0.8840155921177741
```

```
[ ]: from sklearn import metrics
```

```
[ ]: print("Mean Absolute Error: ", metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error:  99675.65202139244
```

```
[ ]: print("Mean Squared Error: ", metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error:  15340780042.516378
```

[ ]:

```
print("Root Mean Squared Error: ", np.sqrt(metrics.
 ↪mean_squared_error(y_test,prediction)))
```

```
Root Mean Squared Error:  123857.90262440415
```