In [73]:
```
##Importing libraries that I use
```

In [259…
```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
import matplotlib.pyplot as plt
from sklearn.neighbors import LocalOutlierFactor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
warnings.filterwarnings("ignore")
```

In [260…
```
##Reading csv file using read_csv function in Pandas libraries
```

In [261…
```
df = pd.read_csv("USA_Housing.csv")
```

In [262…
```
##Overviewing the dataset
```

In [263…
```
df.head()
```

Out[263…

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

In [264…
```
df.tail()
```

Out[264…

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

In [265...
```
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
```

In [266...
```
##Dropping Address column
```

In [267...
```
del df['Address']
```

In [268...
```
df.describe()
```

Out[268...

	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

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
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%	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
50%	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
75%	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
max	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

In [269…
```
##Converting exponential number to integers in Price column
```

In [312…
```
df['Price'] = df['Price'].astype('int32')
df['Price']
```

Out[312…
```
0       1059033
1       1505890
2       1058987
3       1260616
4        630943
         ...
4995    1060193
4996    1482617
4997    1030729
4998    1198656
4999    1298950
Name: Price, Length: 5000, dtype: int32
```

In [271…
```
df
```

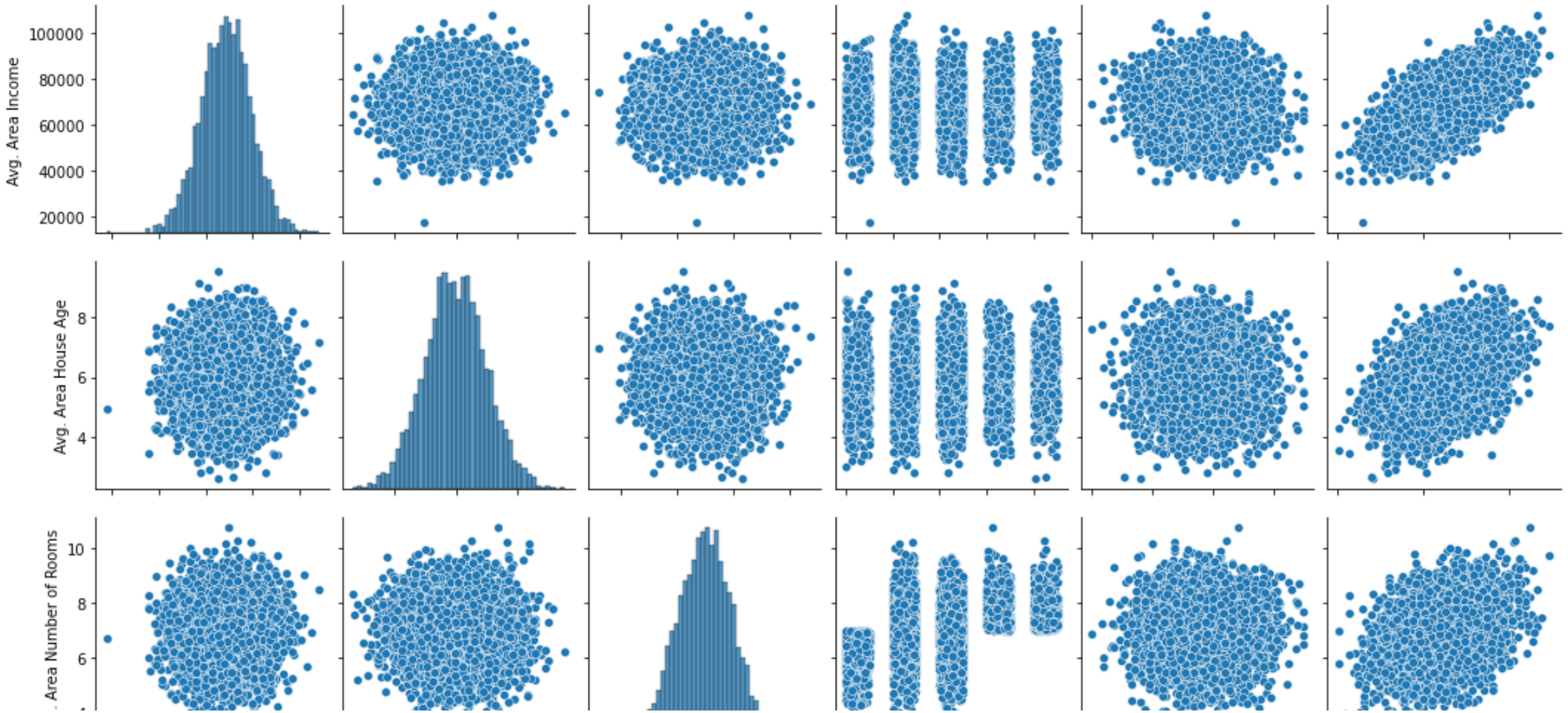
Out[271…

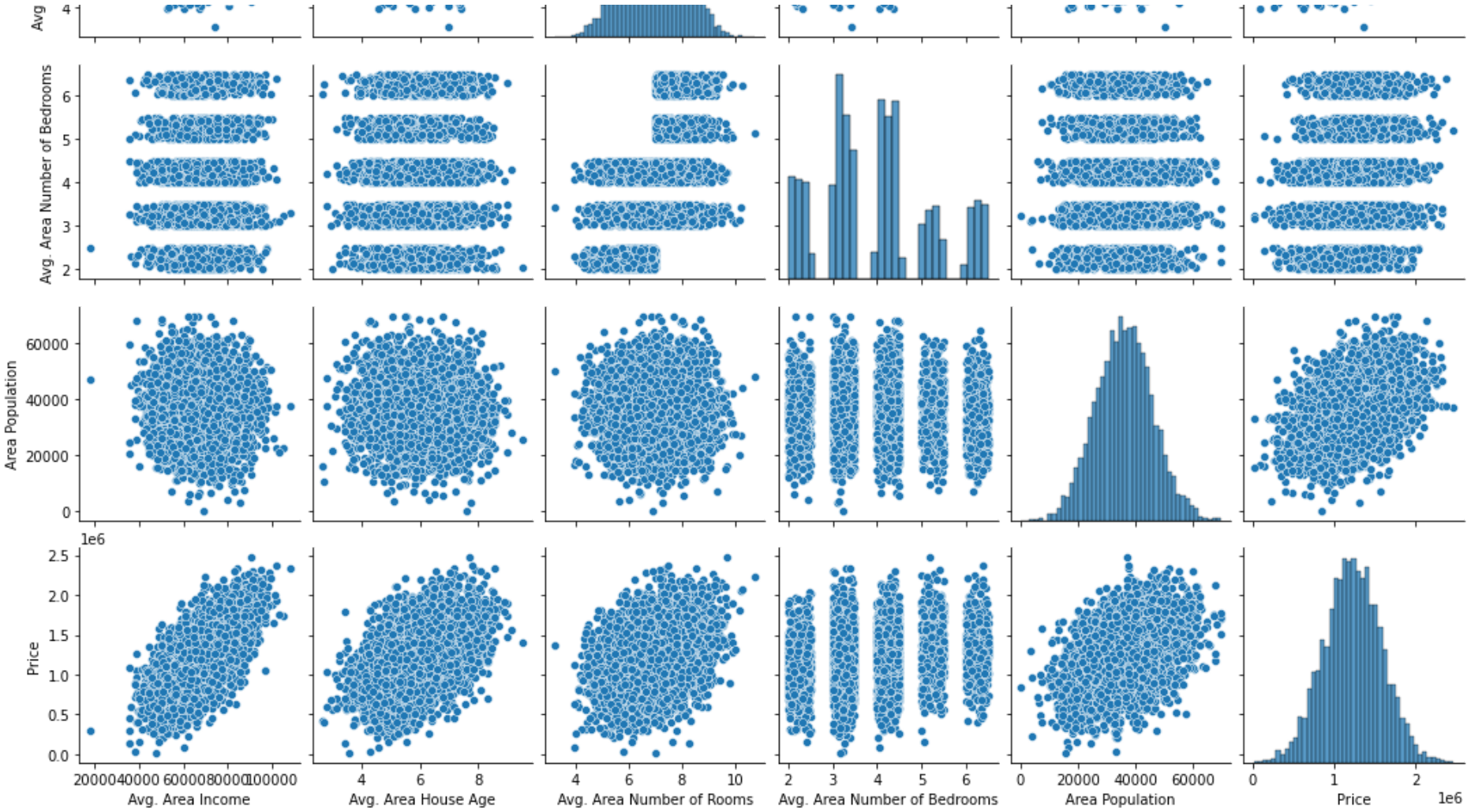
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1059033
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1505890
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1058987
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1260616
4	59982.197226	5.040555	7.839388	4.23	26354.109472	630943

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1060193
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1482617
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1030729
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1198656
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1298950

5000 rows × 6 columns

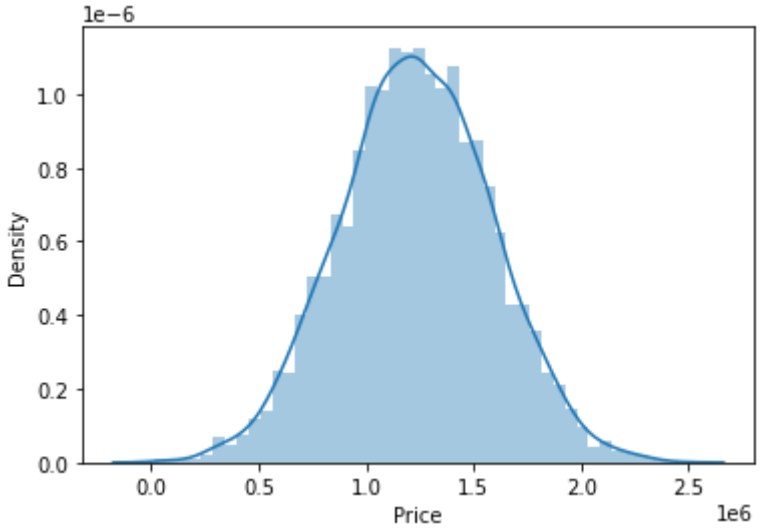
In [272…
```
sns.pairplot(df,markers=['o','s','D']);
```





In [273…
```
##Showing dependent variable using distplot in Seaborn library
```

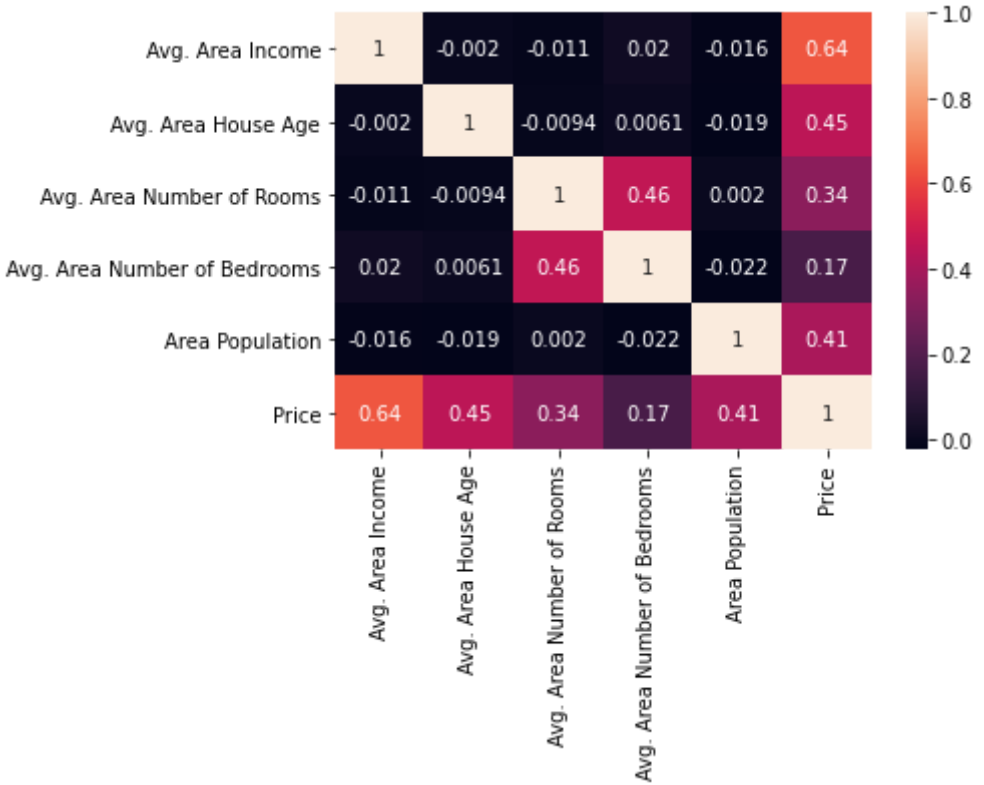
In [274…
```
sns.distplot(df['Price']);
```



In [275…
```
##Showing correlation between variables using heatmap function in Seaborn library
```

In [276…
```
sns.heatmap(df.corr(),annot=True)
```

Out[276… <AxesSubplot:>



In [277…
```
##Splitting dependent and independent variables
```

In [278…
```
X = df[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms','Avg. Area Number of Bedrooms','Area Population']]
y = df[['Price']]
```

In [279…
```
X
```

Out[279…

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	79545.458574	5.682861	7.009188	4.09	23086.800503
1	79248.642455	6.002900	6.730821	3.09	40173.072174
2	61287.067179	5.865890	8.512727	5.13	36882.159400
3	63345.240046	7.188236	5.586729	3.26	34310.242831
4	59982.197226	5.040555	7.839388	4.23	26354.109472

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035
4996	78491.275435	6.999135	6.576763	4.02	25616.115489
4997	63390.686886	7.250591	4.805081	2.13	33266.145490
4998	68001.331235	5.534388	7.130144	5.44	42625.620156
4999	65510.581804	5.992305	6.792336	4.07	46501.283803

5000 rows × 5 columns

In [280…
```
y
```

Out[280…

	Price
0	1059033
1	1505890
2	1058987
3	1260616
4	630943
...	...
4995	1060193
4996	1482617
4997	1030729
4998	1198656
4999	1298950

5000 rows × 1 columns

In [281…
```
##Analyzing multivariables outliers
```

In [282…
```
clf = LocalOutlierFactor(n_neighbors=20,contamination=0.1)
```

In [283…
```
clf.fit_predict(df)
```

Out[283…
```
array([1, 1, 1, ..., 1, 1, 1])
```

In [284…
```
df_scores=clf.negative_outlier_factor_
```

In [285…
```
df_scores[0:20]
```

Out[285…
```
array([-1.11536026, -1.01800941, -0.98971074, -1.00706385, -0.97757888,
       -1.12519572, -1.18037624, -0.97523747, -0.96700226, -1.05412456,
       -0.97721548, -1.00843405, -1.52143484, -0.98264698, -1.00082539,
       -1.0577659 , -0.96084366, -1.07970473, -0.99013277, -1.01408535])
```

In [286…
```
np.sort(df_scores)[0:20]
```

Out[286…
```
array([-3.77197884, -3.60372727, -2.95515785, -2.49487712, -2.49277646,
       -2.40679441, -2.31983953, -2.31976406, -2.26358443, -2.2436647 ,
       -2.23568524, -1.93856087, -1.8899767 , -1.86968521, -1.82042111,
       -1.81529495, -1.76302405, -1.75388591, -1.74290668, -1.732427  ])
```

In [287…
```
##Finding threshold variable
```

In [288…
```
threshold_variable = np.sort(df_scores)[15]
threshold_variable
```

Out[288…
```
-1.8152949456838037
```

In [290…
```
new_df = df[df_scores > threshold_variable ]
new_df
```

Out[290…

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1059033
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1505890
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1058987
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1260616

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
4	59982.197226	5.040555	7.839388	4.23	26354.109472	630943
...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1060193
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1482617
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1030729
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1198656
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1298950

4984 rows × 6 columns

In [291…
```
##After analyzing 15 entries removed
```

In [292…
```
threshold_variable_=df[df_scores < threshold_variable]
threshold_variable_
```

Out[292…

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
90	48904.983269	4.844973	5.448956	3.38	32960.753070	201898
263	40366.616291	4.902940	7.617118	5.07	16349.365394	152071
465	90592.469609	7.700132	9.708803	5.19	37223.876167	2469065
1271	37971.207566	4.291224	5.807510	3.24	33267.767728	31140
1459	35963.330809	3.438547	8.264122	3.28	24435.777302	143027
1530	85175.200626	7.750852	7.271163	3.11	3285.450538	1305972
1661	48735.924512	5.543730	6.091906	2.43	19682.347295	151527
1799	60167.672607	4.590613	3.950973	4.06	16811.303292	88591
2173	50143.644854	4.230051	7.979250	4.04	67601.223558	1168588
2451	63421.903955	7.594954	8.777735	3.09	11511.387050	1432318
3212	47320.657205	3.558054	7.006987	3.16	15776.618595	15938

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
3442	48879.259763	8.147518	7.149646	4.42	63620.011963	1575492
3991	50041.125224	5.981267	8.699555	3.08	68311.695822	1626676
4158	96397.582684	4.451224	6.200118	2.40	22681.929476	1053966
4716	38530.124478	4.265906	8.026969	4.47	67727.229051	1267986

In [293…
```
threshold_variable_.index
```

Out[293…
```
Int64Index([  90,  263,  465, 1271, 1459, 1530, 1661, 1799, 2173, 2451, 3212,
            3442, 3991, 4158, 4716],
           dtype='int64')
```

In [294…
```
## Splitting dependent and indepedent variables as train and test data
```

In [295…
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

In [296…
```
reg=LinearRegression()
```

In [297…
```
model = reg.fit(X_train,y_train)
```

In [298…
```
##Finding the model's intercept
```

In [299…
```
model.intercept_
```

Out[299…
```
array([-2638673.85642023])
```

In [300…
```
##Finding the model's coefficient
```

In [301…
```
model.coef_
```

Out[301…
```
array([[2.16257986e+01, 1.65590393e+05, 1.19827784e+05, 2.36108915e+03,
        1.52165806e+01]])
```

In [311…
```
##Converting the model's decimal coefficients to integer again
```

In [313…
```
(model.coef_).astype('int32')
```

Out[313…
```
array([[    21, 165590, 119827,   2361,     15]])
```

In [ ]:
```
# Increasing 1 unit in Average Area Income is associated with an increase of $21
# Increasing 1 unit in Average Area House Age is associated with an increase of $165590
# Increasing 1 unit in Average Area Number of Rooms is associated with an increase of $119827
# Increasing 1 unit in Average Area Number of Bedrooms is associated with an increase of $2361
# Increasing 1 unit in Average Area Population associated with an increase of $15
```

In [304…
```
## Rsquared calculations
model.score(X,y)
```

Out[304…
```
0.9180100008471123
```

The model's significance increases as it approaches 1

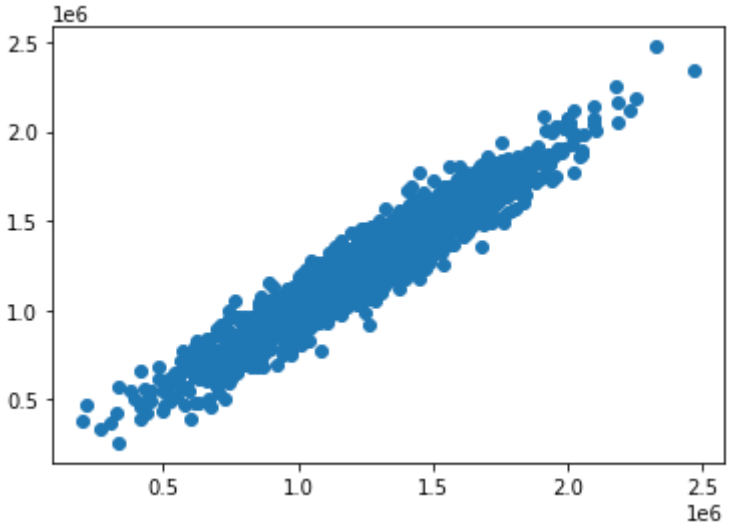
In [305…
```
##Predicting
```

In [306…
```
y_pred = reg.predict(X_test)
y_pred[:10]
```

Out[306…
```
array([[1308535.63219539],
       [1237122.22663937],
       [1243835.14007721],
       [1229241.20495096],
       [1059352.00975644],
       [1542358.20208215],
       [1095210.11913117],
       [ 832675.82175491],
       [ 788905.83572664],
       [1468527.22280102]])
```

In [307…
```
plt.scatter(y_test,y_pred)
```

Out[307…
```
<matplotlib.collections.PathCollection at 0x18a0fa601f0>
```



In [ ]:
```
#-MODEL SUCCESS EVALUATION METHODS-
#Mean Absolute Error is the mean of the absolute value of the errors
#Mean Squared Error is the mean of the squared errors
#Root Mean Squared Error is the square root of the mean of the squared errors
```

In [308…
```
##Calculating mean absolute error, mean squared error and root mean squared error
```

In [309…
```
print('MAE:', metrics.mean_absolute_error(y_test, y_pred))
print('MSE:', metrics.mean_squared_error(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
MAE: 81135.56947992502
MSE: 10068423085.143562
RMSE: 100341.5322044843
```