Housing Price Prediction

Your neighbor is a real estate agent and wants some help predicting housing prices for regions in the USA. It would be great if you could somehow create a model for her that allows her to put in a few features of a house and returns back an estimate of what the house would sell for.

She has asked you if you could help her out with your new data science skills. You say yes, and decide that Linear Regression might be a good path to solve this problem!

Your neighbor then gives you some information about a bunch of houses in regions of the United States,it is all in the data set: USA_Housing.csv.

The data contains the following columns:

- 'Avg. Area Income': Avg. Income of residents of the city house is located in.
- 'Avg. Area House Age': Avg Age of Houses in same city
- 'Avg. Area Number of Rooms': Avg Number of Rooms for Houses in same city
- 'Avg. Area Number of Bedrooms': Avg Number of Bedrooms for Houses in same city
- 'Area Population': Population of city house is located in
- 'Price': Price that the house sold at
- 'Address': Address for the house

Import the necessary libraries

In [85]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

Read 'USA_Housing.csv' dataset and store it in a variable

In [86]:
```
df = pd.read_csv('USA_Housing.csv')
```

View the top 5 rows of the data

In [87]:
```
df.head()
```

Out[87]:

	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\r\nLaurabury, NE 37...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\r\nLake Kathleen, ...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\r\nDanieltown, WI 064...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\r\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\r\nFPO AE 09386

View info about the dataset

In [88]:
```
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
```

View the basic statistical infomation about the dataset

In [89]:
```
df.describe()
```

Out[89]:

	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%	61480.562388	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

Find all the columns in the dataset

In [90]:
```
df.columns
```

Out[90]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

Drop address column from the dataset

In [91]:
```
df.drop(columns=["Address"],inplace=True)
df
```

Out[91]:

	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	1.059034e+06
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05
...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06

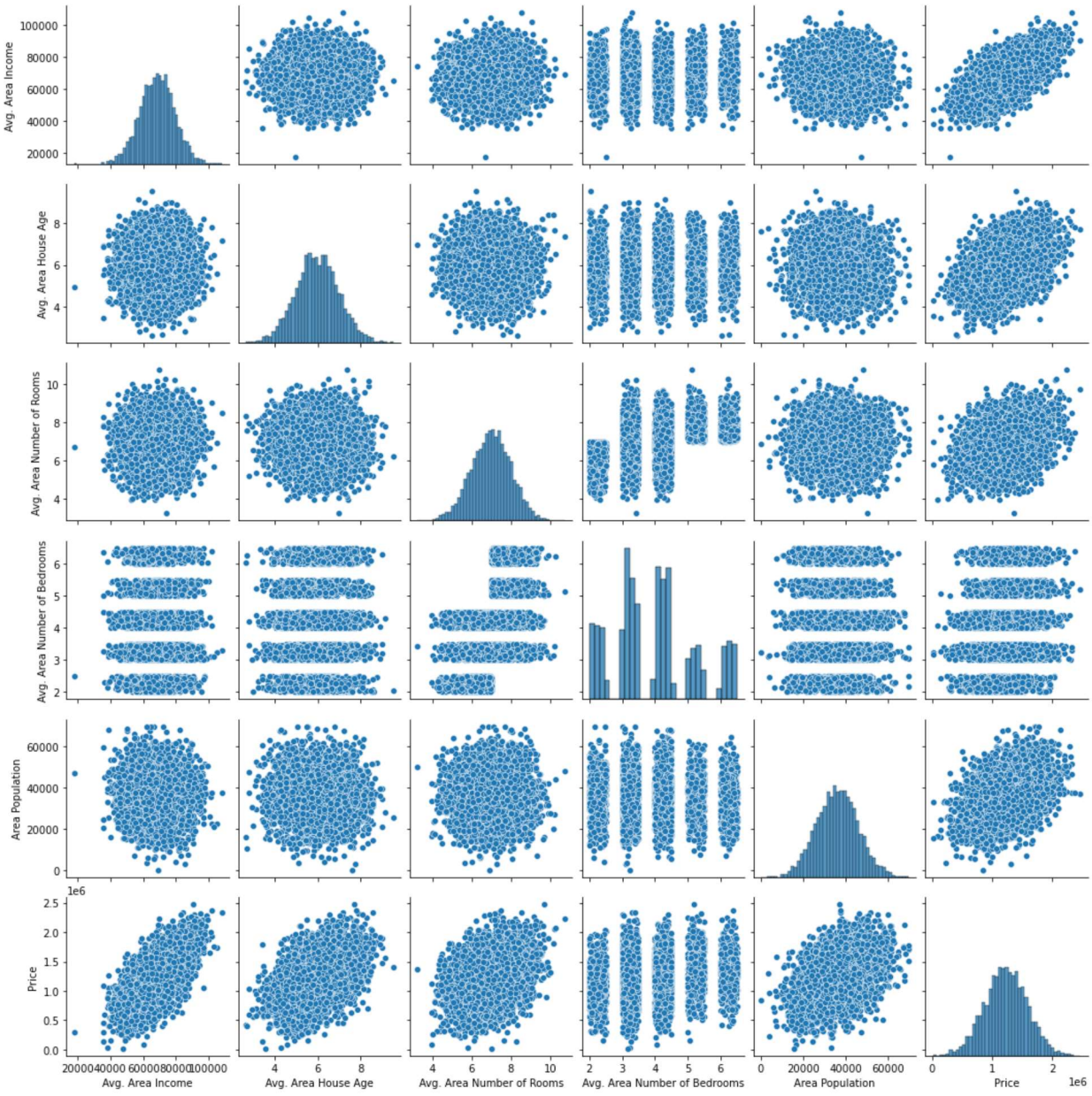
5000 rows × 6 columns

Visualization

Plot a pairplot to know the relationship between each column

In [92]:
```
sns.pairplot(df)
```

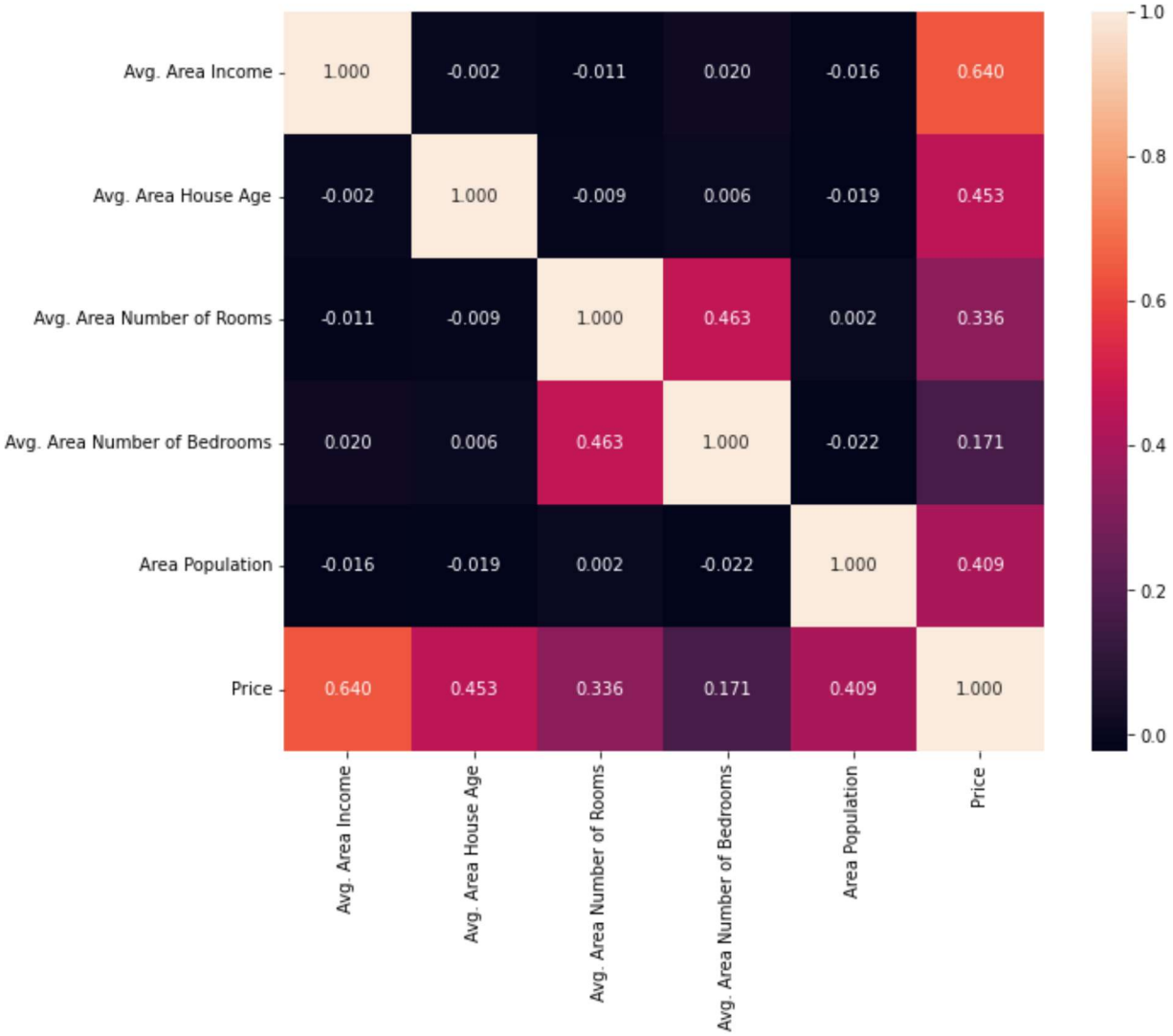
Out[92]:
```
<seaborn.axisgrid.PairGrid at 0x1746e5b7550>
```



Draw a heatmap to identify the correlation

In [93]:
```
plt.figure(figsize=(10,8))
sns.heatmap(df.corr(),annot=True,fmt=".3f")
```

Out[93]:
```
<AxesSubplot:>
```



Assign features and target on X and y variable

In [94]:
```
X=df.drop(columns=['Area Population'])
y=df["Area Population"]
```

Check the shape of X and y

In [95]:
```
X.shape
```

Out[95]:
```
(5000, 5)
```

In [96]:
```
y.shape
```

Out[96]:
```
(5000,)
```

In [72]:
```
# X must be a 2D array and y must be a 1D array
```

Standardise the data using StandardScaler

In [97]:
```
from sklearn.preprocessing import StandardScaler
```

In [98]:
```
st=StandardScaler()
```

In [99]:
```
xcolumns=X.columns
```

In [100…
```
X=st.fit_transform(X)
c=pd.DataFrame(columns=xcolumns)
```

In [119…
```
c.head()
```

Out[119]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Price

Split the data into training and testing set

In [104…
```
from sklearn.model_selection import train_test_split
```

In [105…
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.3,random_state=100)
```

Check the size of X_train and X_test

In [106…
```
X_train.shape
```

Out[106]:
```
(3500, 5)
```

In [107…
```
y_train.shape
```

Out[107]:
```
(3500,)
```

Create a Linear Regression model and train it

In [108…
```
from sklearn.linear_model import LinearRegression
```

In [109…
```
# Create the model
lr=LinearRegression()
```

In [110…
```
# Train the model
lr.fit(X_train,y_train)
```

Out[110]:
```
LinearRegression()
```

Check the score of our model

In [111…
```
lr.score(X_train,y_train)
```

Out[111]:
```
0.6886417112286181
```

Print the Intercept

In [112…
```
print(lr.intercept_)
```

```
36180.33319634007
```

Print the coefficents

In [113…
```
y_pred=lr.predict(X_test)
y_pred
```

Out[113]:
```
array([28160.57180674, 43848.40488343, 33612.82887896, ...,
       43517.8252681 , 34878.66170502, 19983.18942746])
```

Predict using the X_test and store the values in a variable

In [114…
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

Calculate the mean squared error and r2 score

In [115…
```
from sklearn import metrics
```

In [116…
```
#mean_squared_error
mean_squared_error(y_test,y_pred)
```

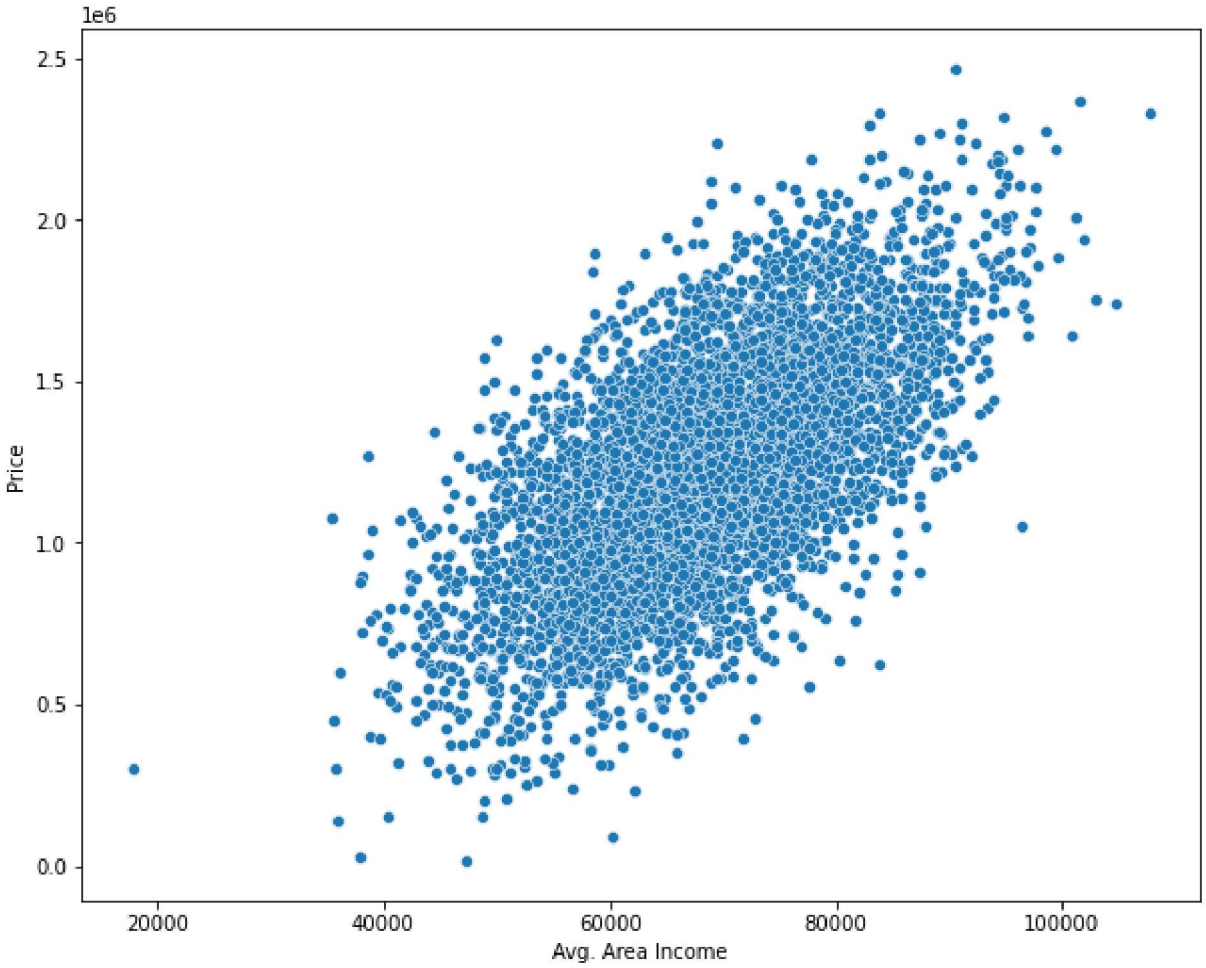
Out[116]:
```
31557023.257717926
```

In [117…
```
#r2 squared absolute_score(y_test,y_pred)
mean_absolute_error(y_test,y_pred)
```

Out[117]:
```
4508.171824695774
```

Create a scatterplot to visualize actual value vs predicted values

In [118…
```
plt.figure(figsize=(10,8))
sns.scatterplot(x='Avg. Area Income',y='Price',data=df)
plt.show()
```


1e6
2.5
2.0
1.5
1.0
0.5
0.0
Price
20000
40000
60000
80000
100000
Avg. Area Income

Great Job!