

# 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 [58]: 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 [59]: df = pd.read_csv('USA_Housing.csv')
```

## View the top 5 rows of the data

```
In [60]: df.head()
```

Out[60]:

	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 A 674\r\nLaurabury, I 3'
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Su 079\r\nLake Kathleen,
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\r\nDanielton WI 06-
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\r\nFPO , 448
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\r\nFF AE 093



## View info about the dataset

In [61]: `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 [62]: `df.describe()`

Out[62]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
<b>count</b>	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
<b>mean</b>	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
<b>std</b>	10657.991214	0.991456	1.005833	1.234137	9925.650114	3.531176e+05
<b>min</b>	17796.631190	2.644304	3.236194	2.000000	172.610686	1.593866e+04
<b>25%</b>	61480.562388	5.322283	6.299250	3.140000	29403.928702	9.975771e+05
<b>50%</b>	68804.286404	5.970429	7.002902	4.050000	36199.406689	1.232669e+06
<b>75%</b>	75783.338666	6.650808	7.665871	4.490000	42861.290769	1.471210e+06
<b>max</b>	107701.748378	9.519088	10.759588	6.500000	69621.713378	2.469066e+06

## Find all the columns in the dataset

In [63]: `df.columns`Out[63]: `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 [64]: `df.drop(columns=["Address"], inplace=True)`  
`df`

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
<b>0</b>	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06
<b>1</b>	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
<b>2</b>	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
<b>3</b>	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
<b>4</b>	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05
...	...	...	...	...	...	...
<b>4995</b>	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06
<b>4996</b>	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06
<b>4997</b>	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06
<b>4998</b>	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06
<b>4999</b>	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06

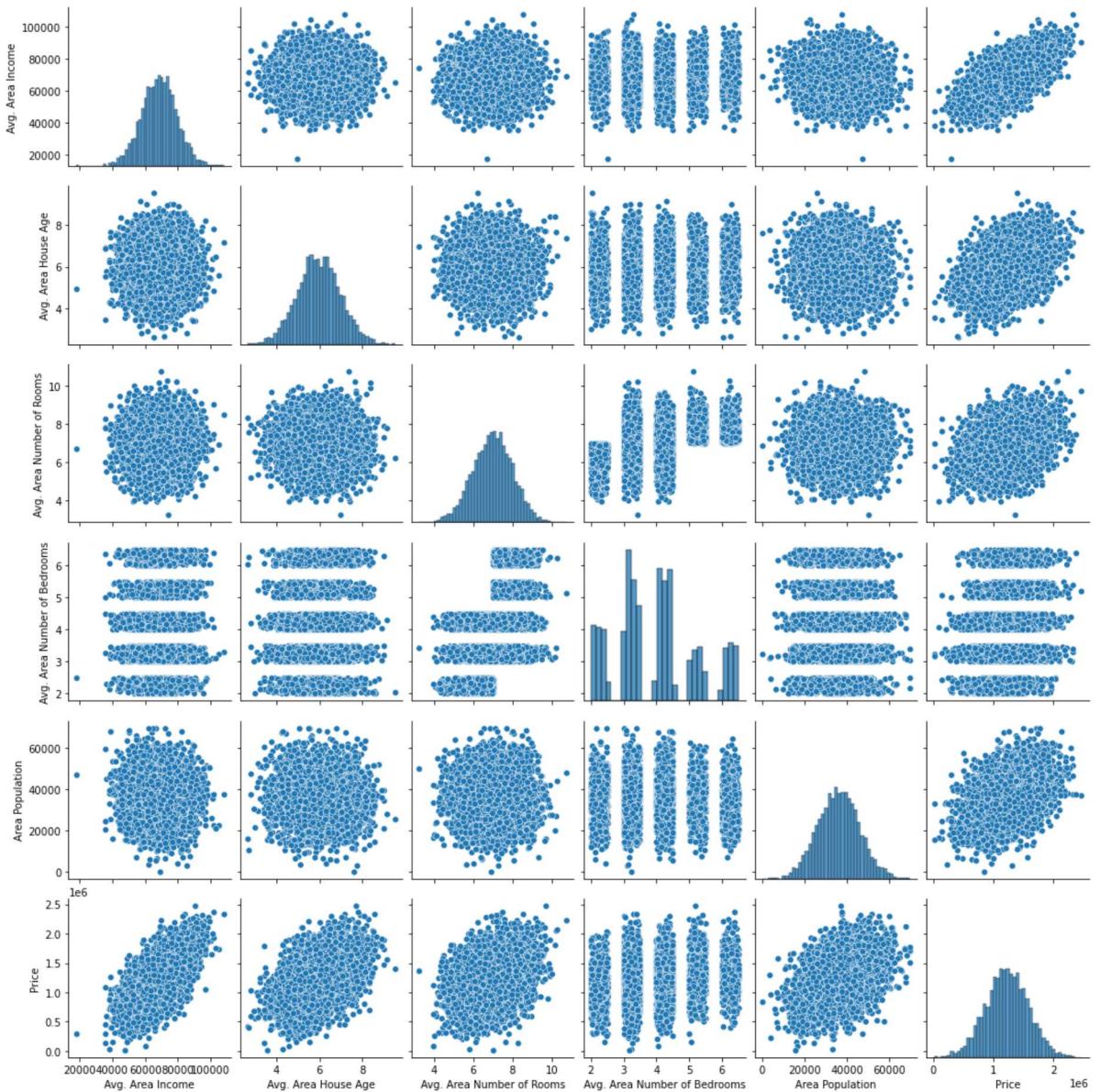
5000 rows × 6 columns

## Visualization

## Plot a pairplot to know the relationship between each column

In [65]: `sns.pairplot(df)`

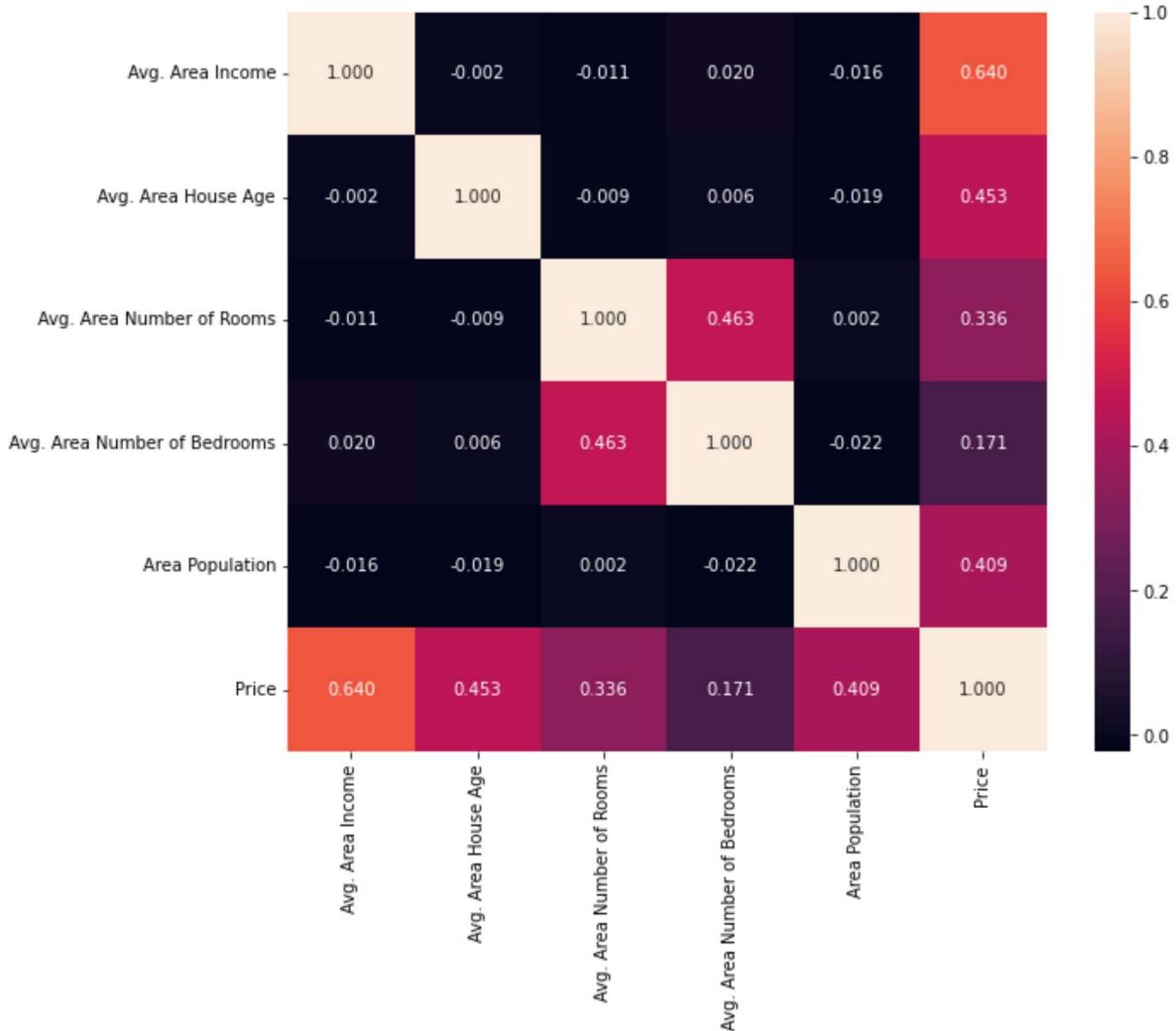
Out[65]: <seaborn.axisgrid.PairGrid at 0x1fad6ab1220>



## Draw a heatmap to identify the correlation

In [66]: `plt.figure(figsize=(10,8))  
sns.heatmap(df.corr(), annot=True, fmt=".3f")`

Out[66]: <AxesSubplot:>



## Assign features and target on X and y variable

```
In [67]: X=df.drop(columns=['Area Population'])
y=df["Area Population"]
```

## Check the shape of X and y

```
In [68]: X.shape
```

```
Out[68]: (5000, 5)
```

```
In [69]: y.shape
```

```
Out[69]: (5000, )
```

```
In [70]: # X must be a 2D array and y must be a 1D array
```

## Standardise the data using StandardScaler

```
In [71]: from sklearn.preprocessing import StandardScaler
```

```
In [72]: st=StandardScaler()
```

```
In [73]: xcolumns=X.columns
```

```
In [74]: X=st.fit_transform(X)
c=pd.DataFrame(columns=xcolumns)
```

```
In [75]: c.head()
```

```
Out[75]:      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 [76]: from sklearn.model_selection import train_test_split
```

```
In [77]: 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 [78]: X_train.shape
```

```
Out[78]: (3500, 5)
```

```
In [79]: y_train.shape
```

```
Out[79]: (3500,)
```

## Create a Linear Regression model and train it

```
In [80]: from sklearn.linear_model import LinearRegression
```

```
In [81]: # Create the model
lr=LinearRegression()
```

```
In [82]: # Train the model
lr.fit(X_train,y_train)
```

```
Out[82]: LinearRegression()
```

## Check the score of our model

```
In [83]: lr.score(X_train,y_train)
```

```
Out[83]: 0.6886417112286182
```

## Print the Intercept

```
In [84]: print(lr.intercept_)
```

```
36180.33319634007
```

## Print the coefficients

```
In [85]: print(lr.coef_)
```

```
[ -10432.36998667 -7445.07633326 -5451.79785884 -83.44592497  
 15930.60058887]
```

## Predict using the X\_test and store the values in a variable

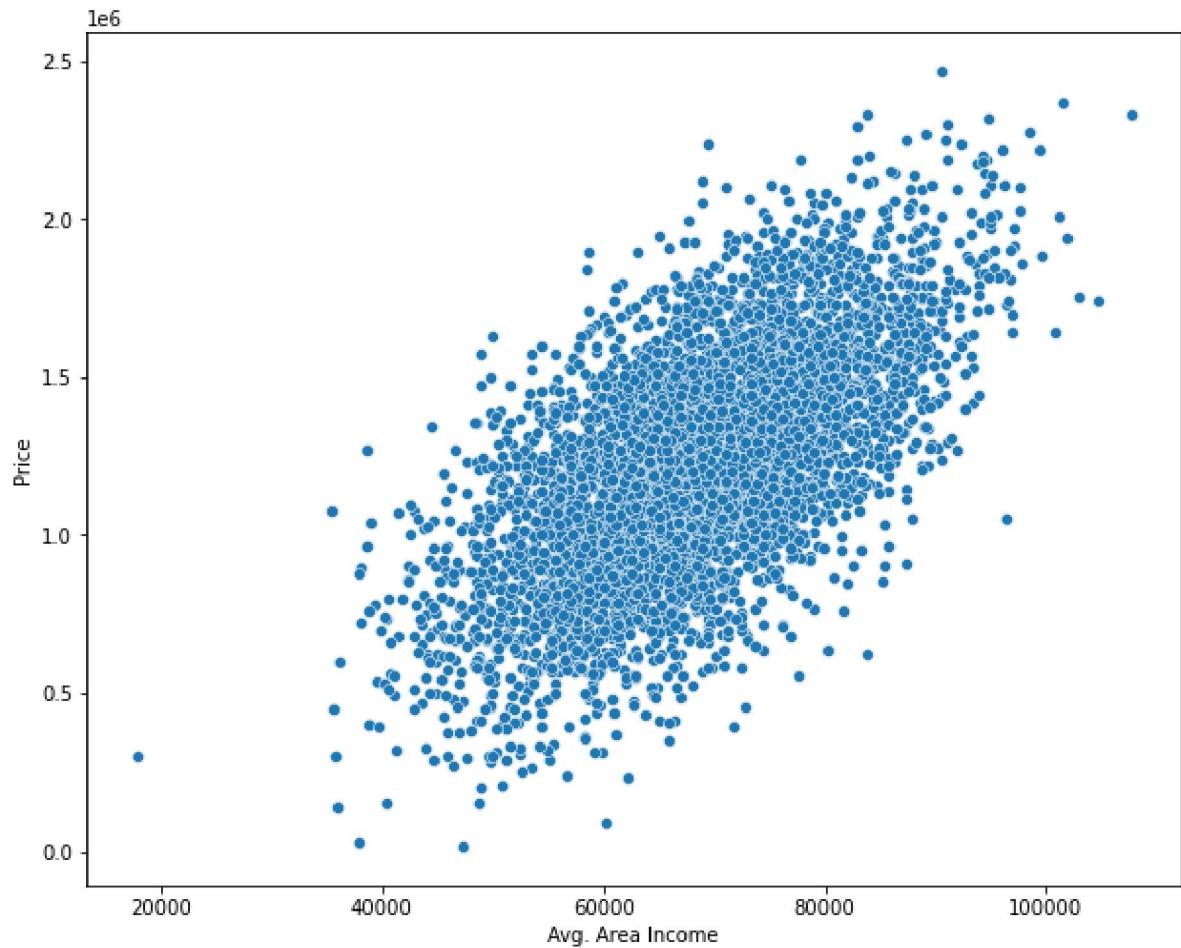
```
In [86]: y_pred=lr.predict(X_test)  
y_pred  
  
Out[86]: array([28160.57180674, 43848.40488343, 33612.82887896, ...,  
 43517.8252681 , 34878.66170502, 19983.18942746])
```

## Calculate the mean squared error and r2 score

```
In [87]: from sklearn.metrics import mean_squared_error,mean_absolute_error  
  
In [88]: #mean_squared_error  
  
mean_squared_error(y_test,y_pred)  
  
Out[88]: 31557023.257717926  
  
In [89]: #r2 scquared absolute_score(y_test,y_pred)  
mean_absolute_error(y_test,y_pred)  
  
Out[89]: 4508.171824695774
```

## Create a scatterplot to visualize actual value vs predicted values

```
In [100...]: plt.figure(figsize=(10,8))  
sns.scatterplot(x='Avg. Area Income',y='Price',data=df)  
plt.show()
```



---

# Great Job!