Problem statement

predicting the house price in USA.To create a model to help him estimate of what the house would sell for.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df=pd.read_csv("USA")
```

To display top 10 rows

In [3]:
```
1 df.head(10)
```

Out[3]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	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
5	80175.754159	4.988408	6.104512	4.04	26748.428425	1.068138e+06	06039 Jennifer Islands Apt. 443\nTracyport, KS...
6	64698.463428	6.025336	8.147760	3.41	60828.249085	1.502056e+06	4759 Daniel Shoals Suite 442\nNguyenburgh, CO ...
7	78394.339278	6.989780	6.620478	2.42	36516.358972	1.573937e+06	972 Joyce Viaduct\nLake William, TN 17778-6483
8	59927.660813	5.362126	6.393121	2.30	29387.396003	7.988695e+05	USS Gilbert\nFPO AA 20957
9	81885.927184	4.423672	8.167688	6.10	40149.965749	1.545155e+06	Unit 9446 Box 0958\nDPO AE 97025

Data Cleaning And Pre-Processing

In [4]:
```
1 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 [5]:
```
1 # Display the statistical summary
2 df.describe()
```

Out[5]:

	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

In [6]:
```
1 # To display the col headings
2 df.columns
```

Out[6]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [7]:
```
cols=df.dropna(axis=1)
```

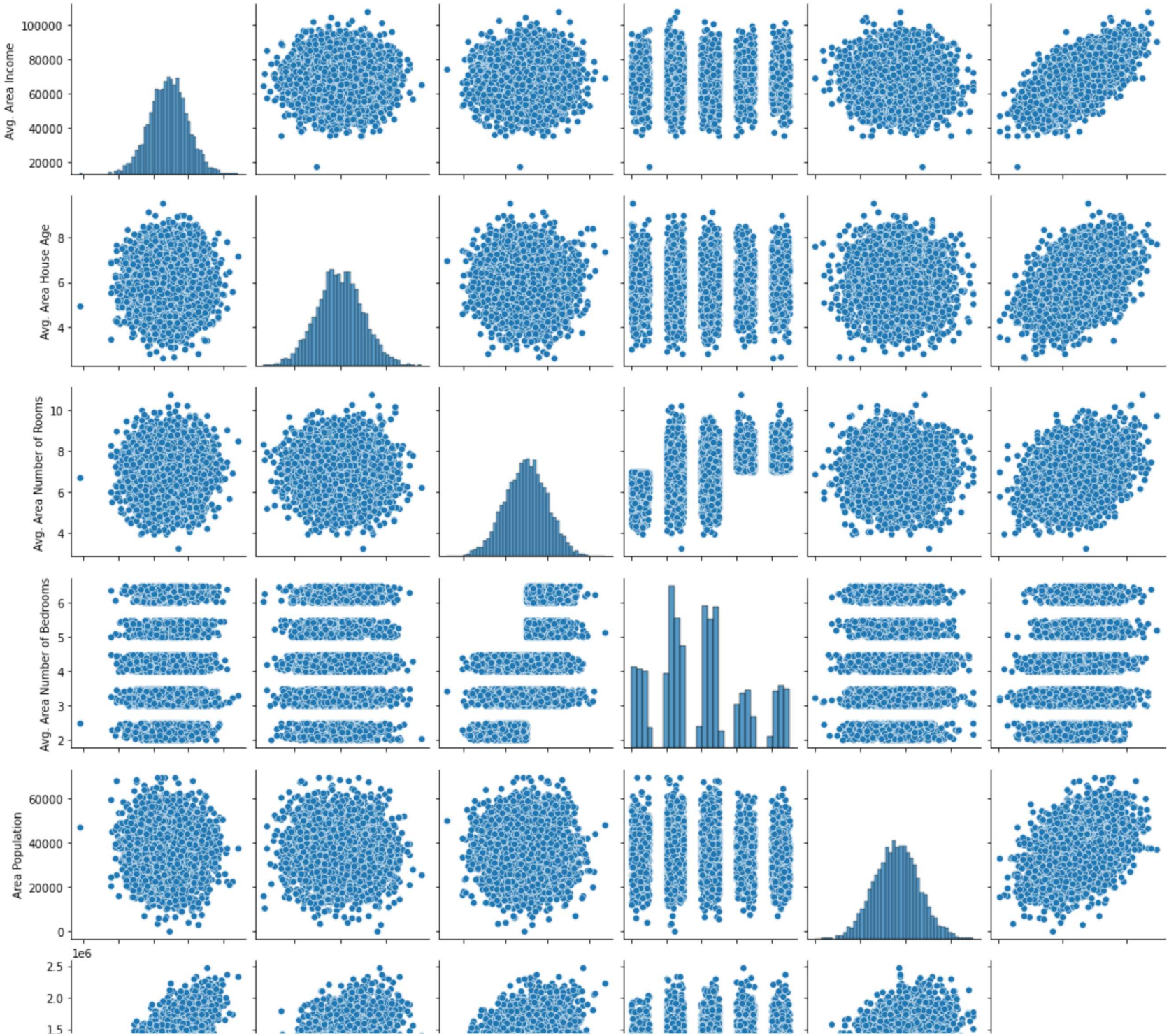
In [8]:
```
cols.columns
```

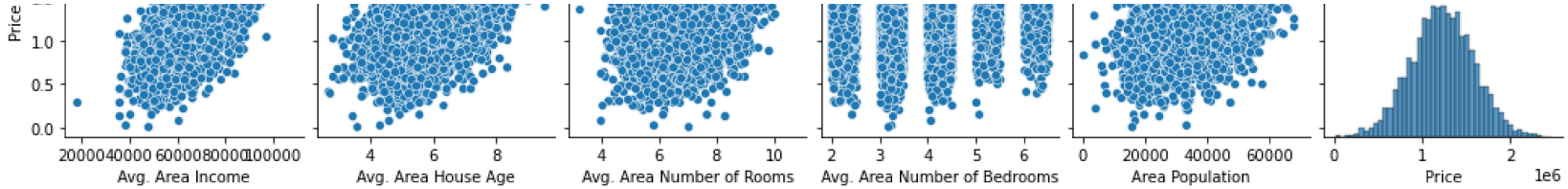
Out[8]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

EDA and Visualization

In [9]:
```
1 sns.pairplot(cols)
```

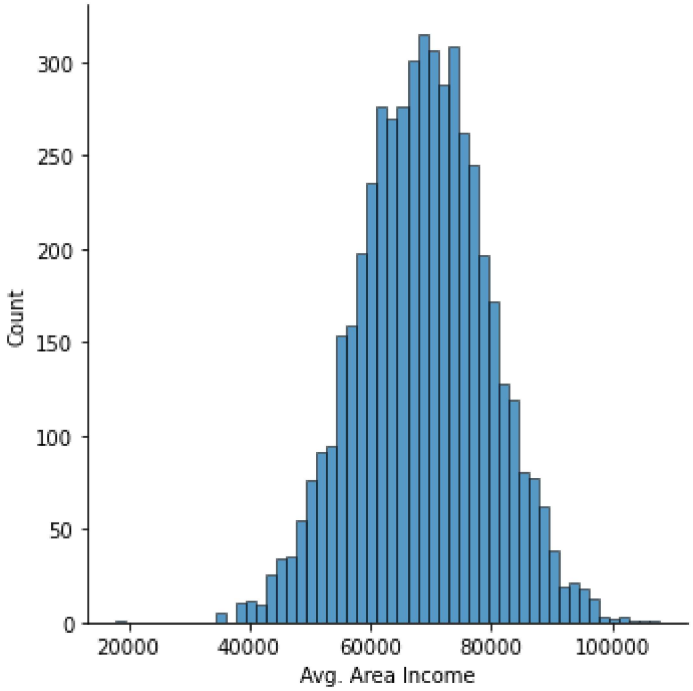
Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x1dddde41d60>
```


Avg. Area Income
Avg. Area House Age
Avg. Area Number of Rooms
Avg. Area Number of Bedrooms
Area Population
1e6



In [11]:
```
1 sns.displot(df['Avg. Area Income'])
```

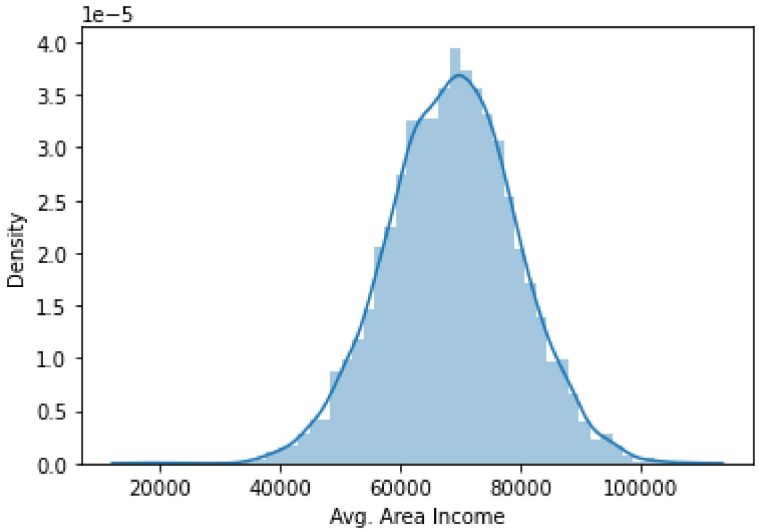
Out[11]: <seaborn.axisgrid.FacetGrid at 0x1dde091efd0>



In [12]:
```
# We use displot in older version we get distplot use displot
sns.distplot(df['Avg. Area Income'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a dep
recated function and will be removed in a future version. Please adapt your code to use either `displot` (a
figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[12]: <AxesSubplot:xlabel='Avg. Area Income', ylabel='Density'>



In [13]:

```
df1=cols[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms']]
df1
```

Out[13]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms
0	79545.458574	5.682861	7.009188	4.09
1	79248.642455	6.002900	6.730821	3.09
2	61287.067179	5.865890	8.512727	5.13
3	63345.240046	7.188236	5.586729	3.26
4	59982.197226	5.040555	7.839388	4.23
...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46
4996	78491.275435	6.999135	6.576763	4.02
4997	63390.686886	7.250591	4.805081	2.13
4998	68001.331235	5.534388	7.130144	5.44
4999	65510.581804	5.992305	6.792336	4.07

5000 rows × 4 columns

In [14]:
```
1 sns.heatmap(df1.corr())
```

Out[14]: <AxesSubplot:>

To train the model - MODEL BUILD

Going to train linear regression model;We split our data into 2 variables x and y where x is independent var(input) and y is dependent on x(output), we could ignore address col as it is not required for our model

In [15]:
```
1 x=df1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
2        'Avg. Area Number of Bedrooms']]
3 y=df1[['Avg. Area Income']]
```

To split the dataset into test data

In [16]:
```
# importing lib for splitting test data
from sklearn.model_selection import train_test_split
```

In [17]:
```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [18]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[18]:
```
LinearRegression()
```

In [19]:
```
print(lr.intercept_)
```

```
[-1.45519152e-11]
```

In [20]:
```
print(lr.score(x_test,y_test))
```

```
1.0
```

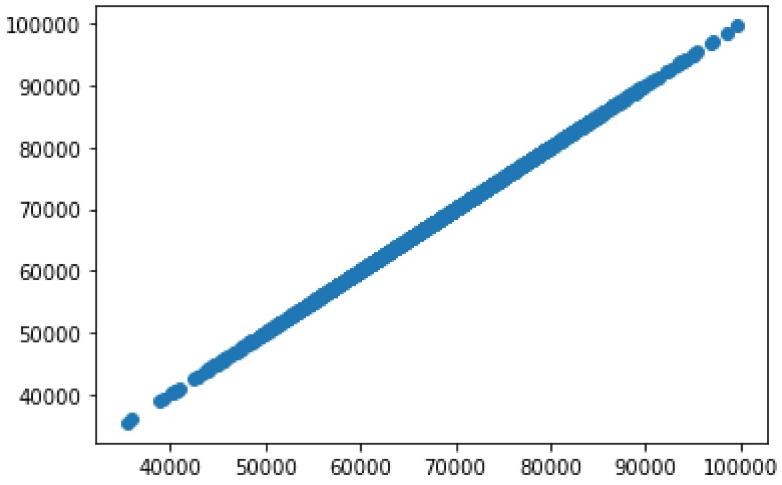
In [21]:
```
coeff=pd.DataFrame(lr.coef_)
coeff
```

Out[21]:

	0	1	2	3
0	1.0	-1.421563e-15	-2.056758e-15	1.100120e-14

In [22]:
```
pred = lr.predict(x_test)
plt.scatter(y_test,pred)
```

Out[22]: <matplotlib.collections.PathCollection at 0x1dde2aafb20>



In [23]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [24]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[24]: Ridge(alpha=10)

In [25]:
```
rr.score(x_test,y_test)
```

Out[25]: 1.0

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]: Lasso(alpha=10)

In [27]:
```
la.score(x_test,y_test)
```

Out[27]: 0.9999999999999923

In [ ]:
```
1
```