

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

```
In [2]: df=pd.read_csv("10_USA_Housing.csv")
df
```

Out[2]:

	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\nDanielstown, 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
...
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...

5000 rows × 7 columns

```
In [3]: 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
```

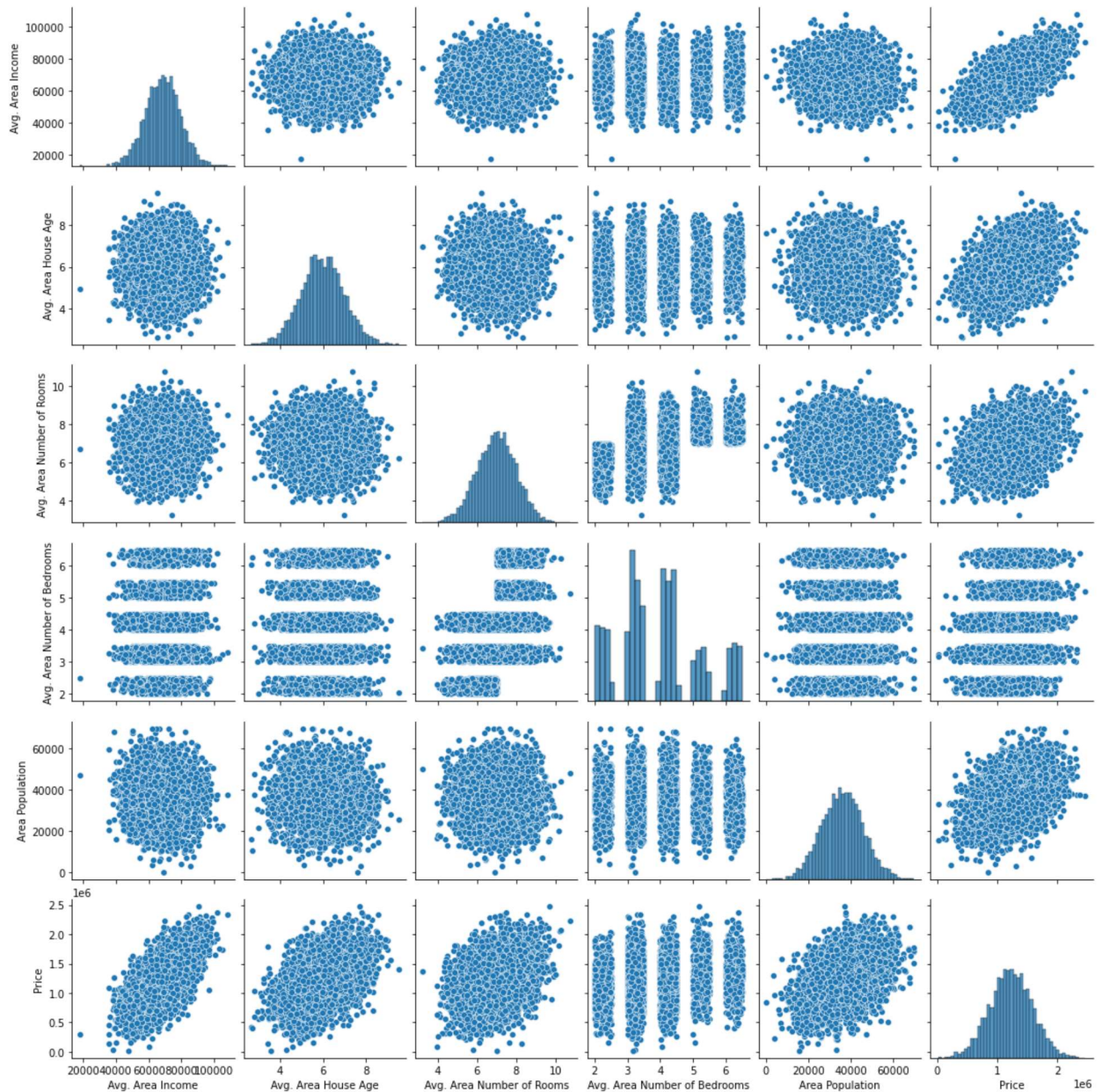
```
In [4]: df.describe()
```

Out[4]:

	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

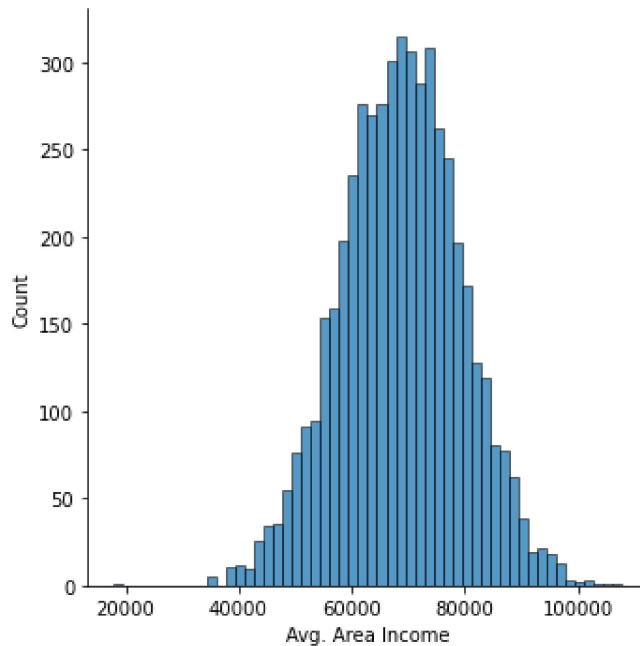
```
In [5]: sns.pairplot(df)
```

```
Out[5]: <seaborn.axisgrid.PairGrid at 0x18b48206760>
```



```
In [6]: sns.displot(df['Avg. Area Income'])
```

```
Out[6]: <seaborn.axisgrid.FacetGrid at 0x18b4ceca6a0>
```



```
In [7]: df1=df.drop(['Address'],axis=1)
df1
```

```
Out[7]:
```

	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

```
In [8]: sns.heatmap(df1.corr())
```

```
Out[8]: <AxesSubplot:>
```



```
In [9]: from sklearn.model_selection import train_test_split  
from sklearn.linear_model import LinearRegression
```

```
In [10]: y=df['Avg. Area Income']
x=df1.drop(['Avg. Area Income','Price'],axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3)
print(x_train)
```

	Avg. Area House Age	Avg. Area Number of Rooms \
1213	5.675774	7.217068
2186	5.689944	5.761823
2972	7.084664	6.584017
2299	5.830240	7.930393
4462	5.232204	5.458380
...
4379	6.313972	5.519717
494	5.930502	6.974340
496	6.987280	3.236194
888	7.827795	7.267250
3727	5.543498	6.172884

	Avg. Area Number of Bedrooms	Area Population
1213	5.11	30773.258989
2186	2.37	31879.323843
2972	3.13	42939.274240
2299	5.19	9579.071782
4462	2.01	54737.926636
...
4379	2.39	33579.913298
494	4.47	28851.601404
496	3.42	50233.790310
888	4.38	24199.052753
3727	2.25	32850.762037

[3500 rows x 4 columns]

```
In [11]: model=LinearRegression()
model.fit(x_train,y_train)
model.intercept_
```

Out[11]: 70501.15927059163

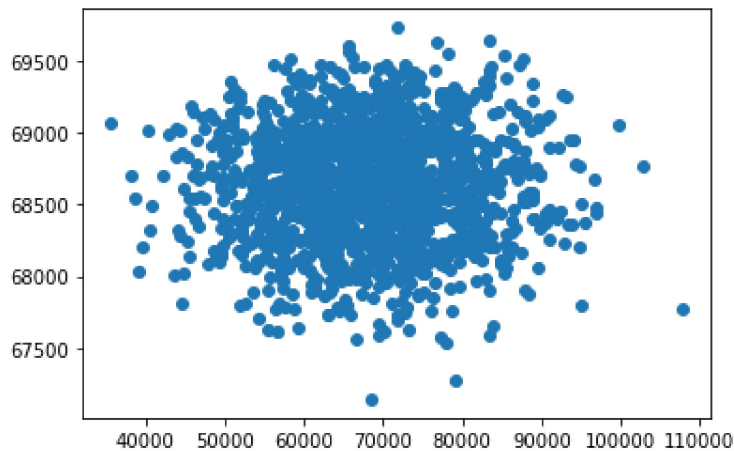
```
In [12]: coeff=pd.DataFrame(model.coef_,x.columns,columns=["Coefficient"])
coeff
```

Out[12]:

	Coefficient
Avg. Area House Age	-133.790199
Avg. Area Number of Rooms	-308.405210
Avg. Area Number of Bedrooms	316.128556
Area Population	-0.004836

```
In [13]: prediction=model.predict(x_test)
plt.scatter(y_test,prediction)
```

```
Out[13]: <matplotlib.collections.PathCollection at 0x18b50855b20>
```



```
In [14]: model.score(x_test,y_test)
```

```
Out[14]: -0.00031033545078851255
```

```
In [15]: from sklearn.linear_model import Ridge,Lasso
```

```
In [16]: rr = Ridge(alpha=10)
rr.fit(x_train,y_train)
```

```
Out[16]: Ridge(alpha=10)
```

```
In [17]: rr.score(x_test,y_test)
```

```
Out[17]: -0.00030355637971712923
```

```
In [18]: la = Lasso(alpha=10)
la.fit(x_train,y_train)
```

```
Out[18]: Lasso(alpha=10)
```

```
In [19]: la.score(x_test,y_test)
```

```
Out[19]: -0.00020112218876611188
```

```
In [20]: from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
print(en.coef_)
print(en.intercept_)
print(en.predict(x_test))
print(en.score(x_test,y_test))
from sklearn import metrics
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
[-8.47991004e+01 -1.58656620e+02  1.95218817e+02 -4.84553561e-03]
```

```
69644.25907581404
```

```
[68276.14125042 68772.41724707 68654.44749494 ... 68564.61550946
```

```
68440.31239906 69104.06136707]
```

```
0.00016501745274866142
```

```
Mean Absolute Error: 8463.243205483113
```

```
Mean Squared Error: 112651739.45437533
```

```
Root Mean Squared Error: 10613.752373895639
```