LINEAR REGRESSION MODEL ---> USA HOUSING PRICES

In [2]:
```
#imports
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [4]:
```
#Reading file
df = pd.read_csv('USA_Housing.csv')
df.head(2)
```

Out[4]:

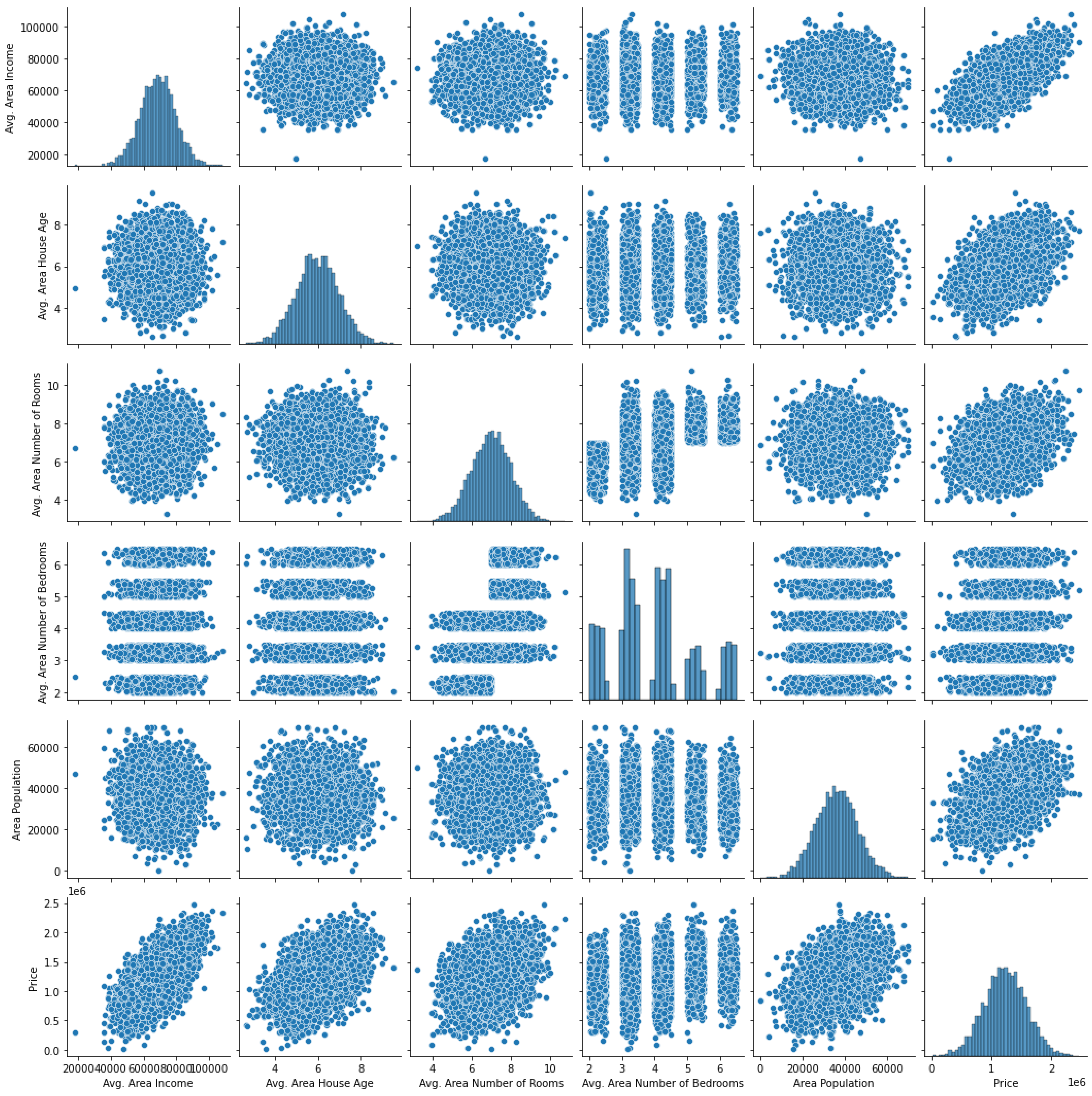
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1059033.558	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1505890.915	188 Johnson Views Suite 079\nLake Kathleen, CA...

In [5]:
```
#Determining the number of entries
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 [6]:
```
sns.pairplot(df)
```

Out[6]:
```
<seaborn.axisgrid.PairGrid at 0x29cca7d4a00>
```



In [7]:
```
df.columns
```

Out[7]:
```
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 [8]:
```
#Declaring the data into required feature and prediction feature
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 [9]:
```
#Splitting the data into training and testing set
from sklearn.model_selection import train_test_split

X_train, X_test , y_train , y_test = train_test_split(X,y,test_size=0.3)
```

In [10]:
```
#Importing LinearRegression model
from sklearn.linear_model import LinearRegression
```

In [11]:
```
#Initializing the model for use
lr = LinearRegression()
```

In [12]:
```
#Fitting the training data on our model
lr.fit(X_train,y_train)
```

Out[12]:
```
LinearRegression()
```

In [13]:
```
#Intercept ----> Detrmining the value of response if the predictor variables are zero
print(lr.intercept_)
```

```
-2630108.9848230053
```

In [17]:
```
#Coefficients ----> This determines that if all other features are held constant then 1 unit change in that feature will affect the price ogf targe
lr.coef_
cdf = pd.DataFrame(lr.coef_ , X.columns , columns = ['Coeff'])
cdf
```

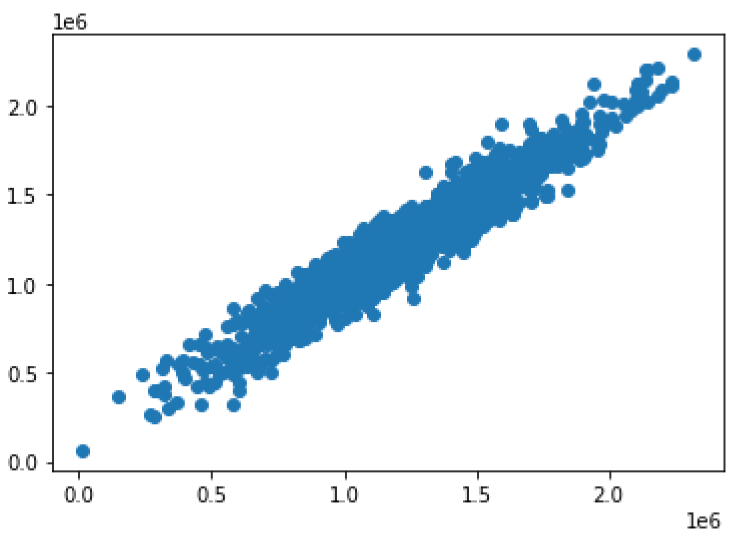
Out[17]:

	Coeff
Avg. Area Income	21.543042
Avg. Area House Age	164974.342435
Avg. Area Number of Rooms	119936.487959
Avg. Area Number of Bedrooms	2163.200874
Area Population	15.260811

In [19]:
```
#Predicting the prices
predictions = lr.predict(X_test)
```

In [20]:
```
#Plotting the graph
plt.scatter(y_test, predictions)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x29ccf53f8e0>
```



In [21]:
```
#Evaluation metrics for our model
from sklearn import metrics
```

In [28]:
```
#Since this is a regression model we will determine the performance of our model on the basis of mae , mse rmse
mae = metrics.mean_absolute_error(y_test,predictions)
mse = metrics.mean_squared_error(y_test,predictions)
rmse = np.sqrt(metrics.mean_squared_error(y_test,predictions))
```

In [27]:
```
print(mae)
```

```
81051.1958528423
```

In [29]:
```
print(rmse)
```

```
101463.76909377535
```

In [ ]:
```

```