House_Price_Prediction

March 12, 2024

0.0.1 Building a machine learning model for predictin house prices

[ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
```

0.0.2 1. Input

[ ]:
```
USAhousing = pd.read_csv('USA_Housing.csv')
```

[ ]:
```
USAhousing.head()
```

[ ]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	\
0	79545.458574	5.682861	7.009188	
1	79248.642455	6.002900	6.730821	
2	61287.067179	5.865890	8.512727	
3	63345.240046	7.188236	5.586729	
4	59982.197226	5.040555	7.839388	

	Avg. Area Number of Bedrooms	Area Population	Price	\
0	4.09	23086.800503	1.059034e+06	
1	3.09	40173.072174	1.505891e+06	
2	5.13	36882.159400	1.058988e+06	
3	3.26	34310.242831	1.260617e+06	
4	4.23	26354.109472	6.309435e+05	

	Address
0	208 Michael Ferry Apt. 674\nLaurabury, NE 3701…
1	188 Johnson Views Suite 079\nLake Kathleen, CA…
2	9127 Elizabeth Stravenue\nDanieltown, WI 06482…
3	USS Barnett\nFPO AP 44820
4	USNS Raymond\nFPO AE 09386

[ ]:
```
USAhousing.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
```

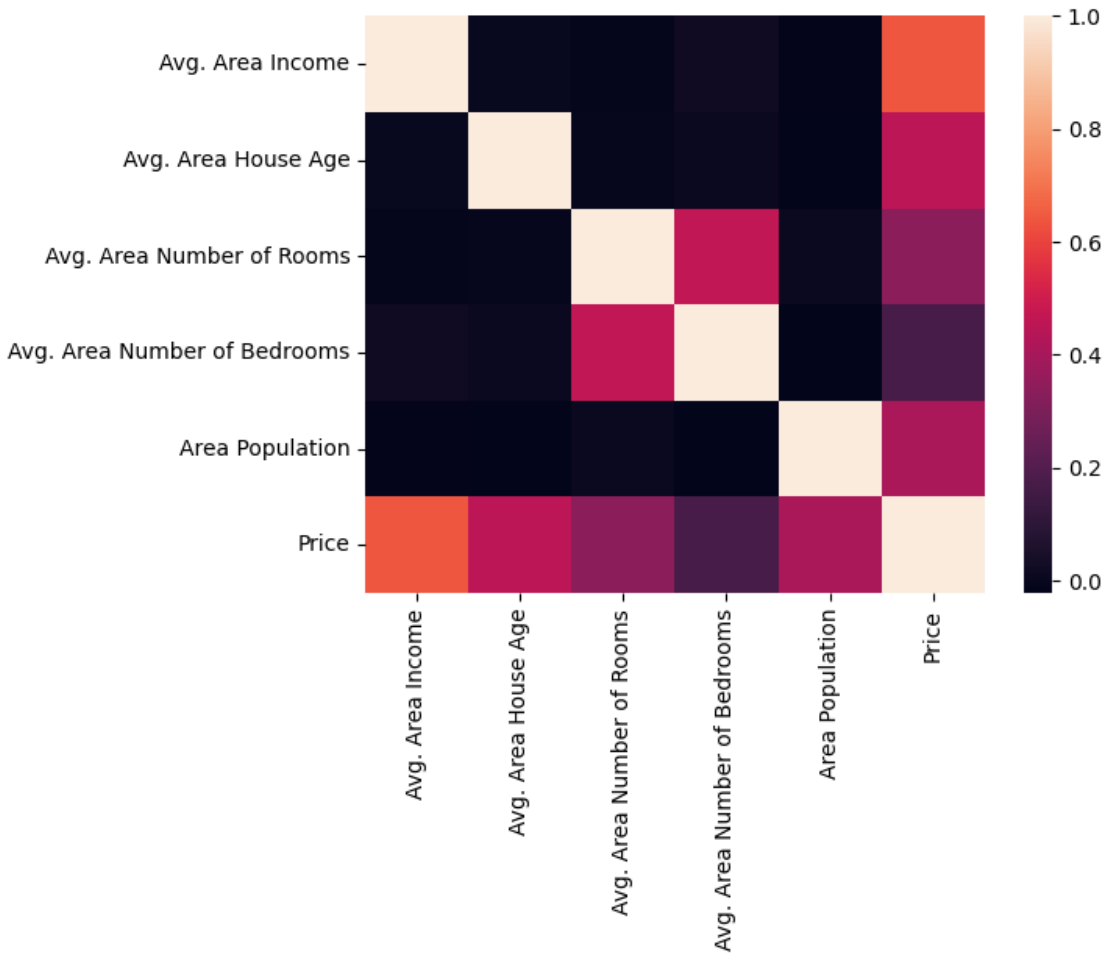
[ ]:
```
USAhousing.describe()
```

[ ]:
```
       Avg. Area Income  Avg. Area House Age  Avg. Area Number of Rooms  \
count       5000.000000          5000.000000                5000.000000   
mean       68583.108984             5.977222                   6.987792   
std        10657.991214             0.991456                   1.005833   
min        17796.631190             2.644304                   3.236194   
25%        61480.562388             5.322283                   6.299250   
50%        68804.286404             5.970429                   7.002902   
75%        75783.338666             6.650808                   7.665871   
max       107701.748378             9.519088                  10.759588   

       Avg. Area Number of Bedrooms  Area Population         Price  
count                   5000.000000      5000.000000  5.000000e+03  
mean                       3.981330     36163.516039  1.232073e+06  
std                        1.234137      9925.650114  3.531176e+05  
min                        2.000000       172.610686  1.593866e+04  
25%                        3.140000     29403.928702  9.975771e+05  
50%                        4.050000     36199.406689  1.232669e+06  
75%                        4.490000     42861.290769  1.471210e+06  
max                        6.500000     69621.713378  2.469066e+06  
```

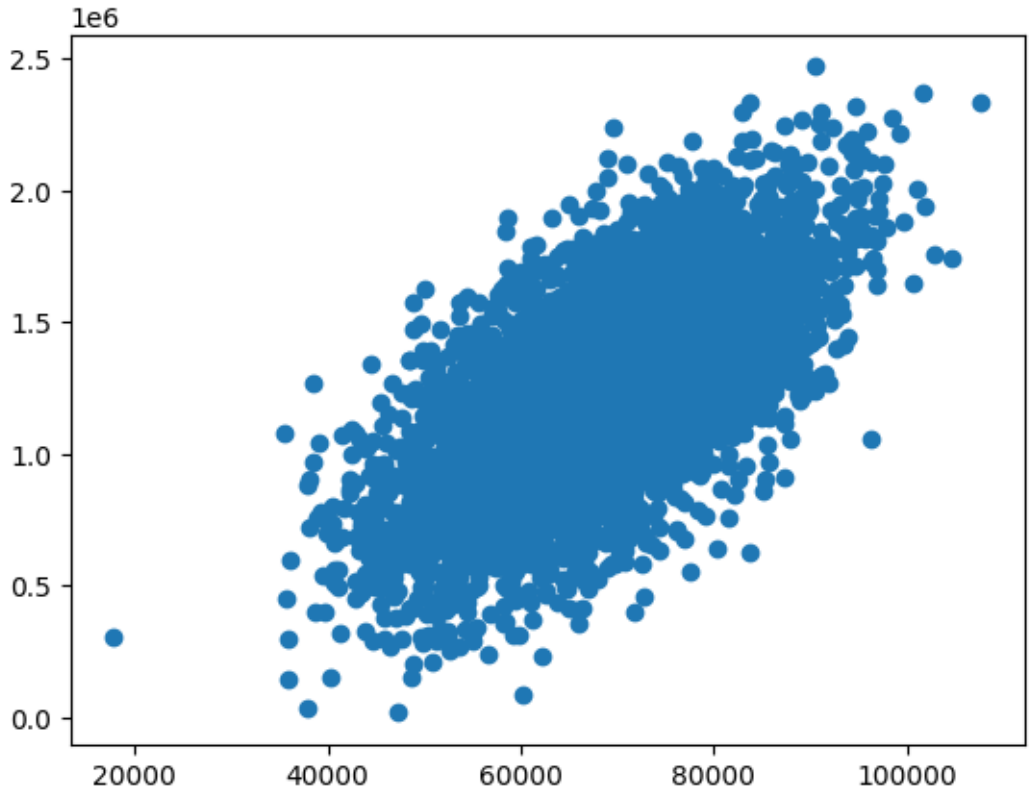
[ ]:
```
sns.heatmap(USAhousing.iloc[:,0:6].corr())
```

[ ]:
```
<Axes: >
```



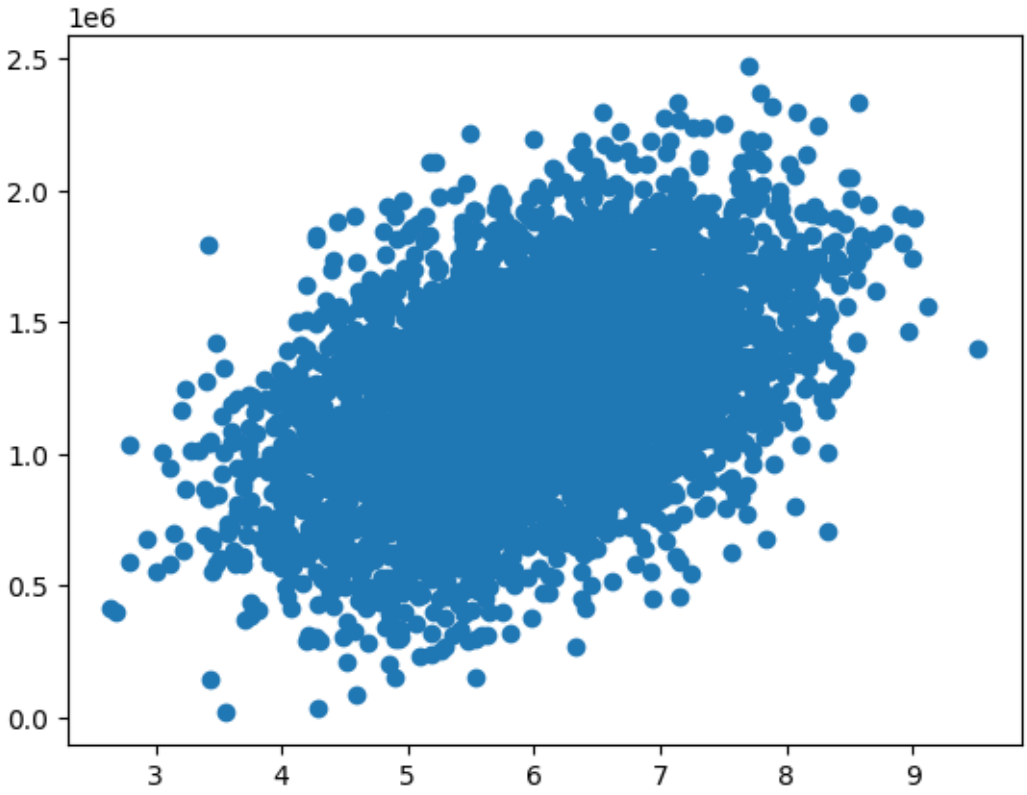
[ ]:
```
plt.scatter(USAhousing['Avg. Area Income'],USAhousing['Price'])
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x21da5135d90>
```



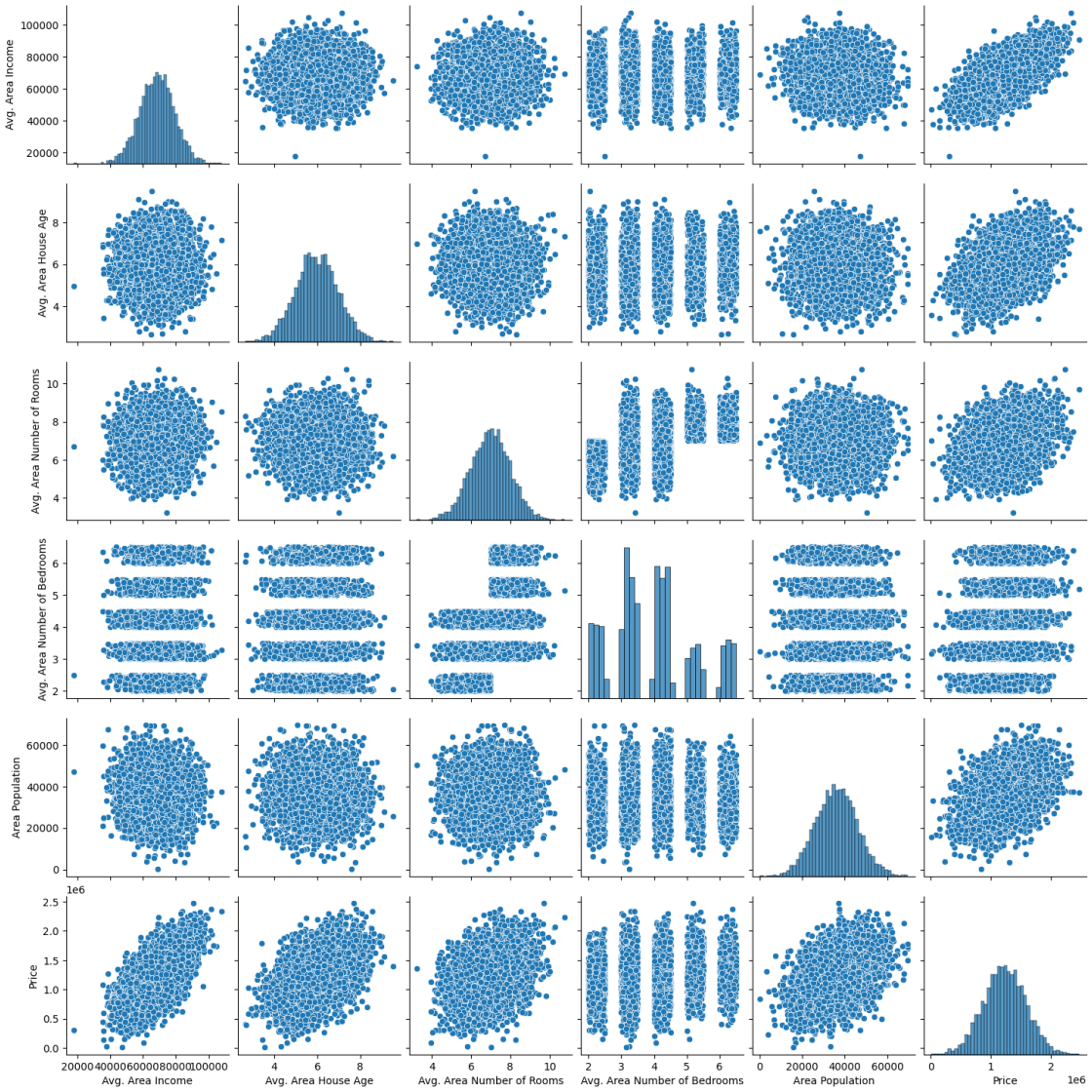
[ ]:
```
plt.scatter(USAhousing['Avg. Area House Age'],USAhousing['Price'])
```

[ ]:
```
<matplotlib.collections.PathCollection at 0x21da51b1610>
```



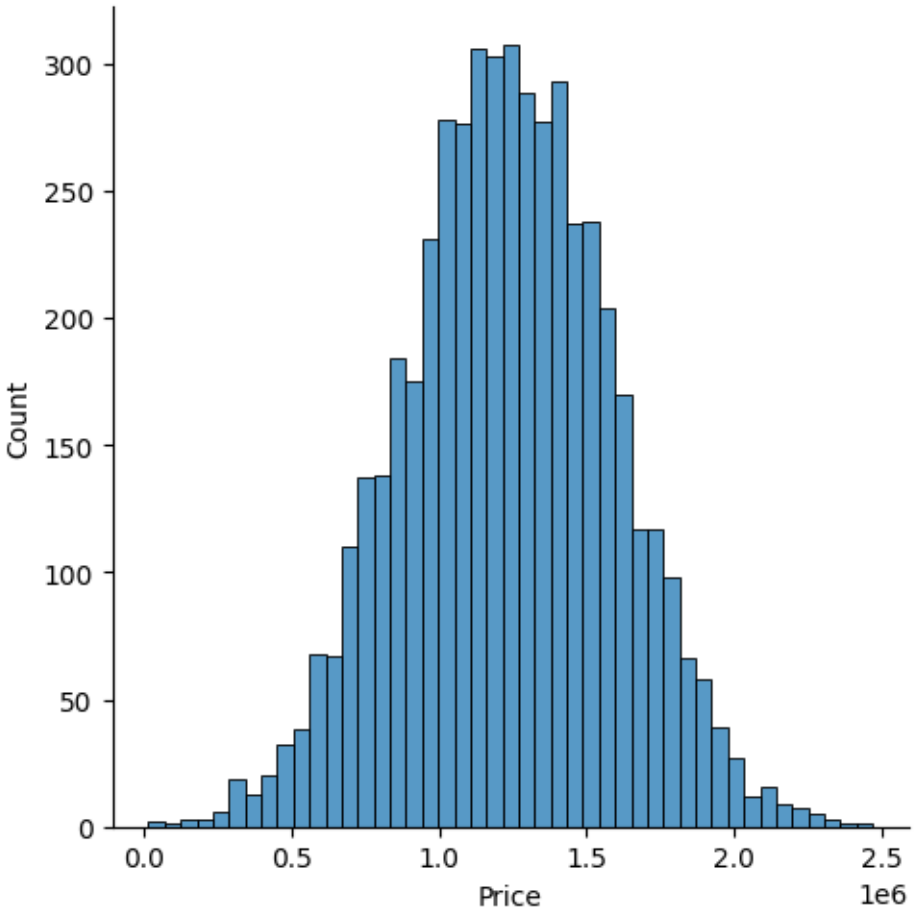
[ ]:
```
sns.pairplot(USAhousing)
```

[ ]:
```
<seaborn.axisgrid.PairGrid at 0x21da51b3380>
```



[ ]:
```
sns.displot(USAhousing['Price'])
```

[ ]: <seaborn.axisgrid.FacetGrid at 0x21da7bcb530>



[ ]:
```
X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of␣
↪Rooms',
               'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,␣
↪random_state=101)
```

0.0.3 2. Process

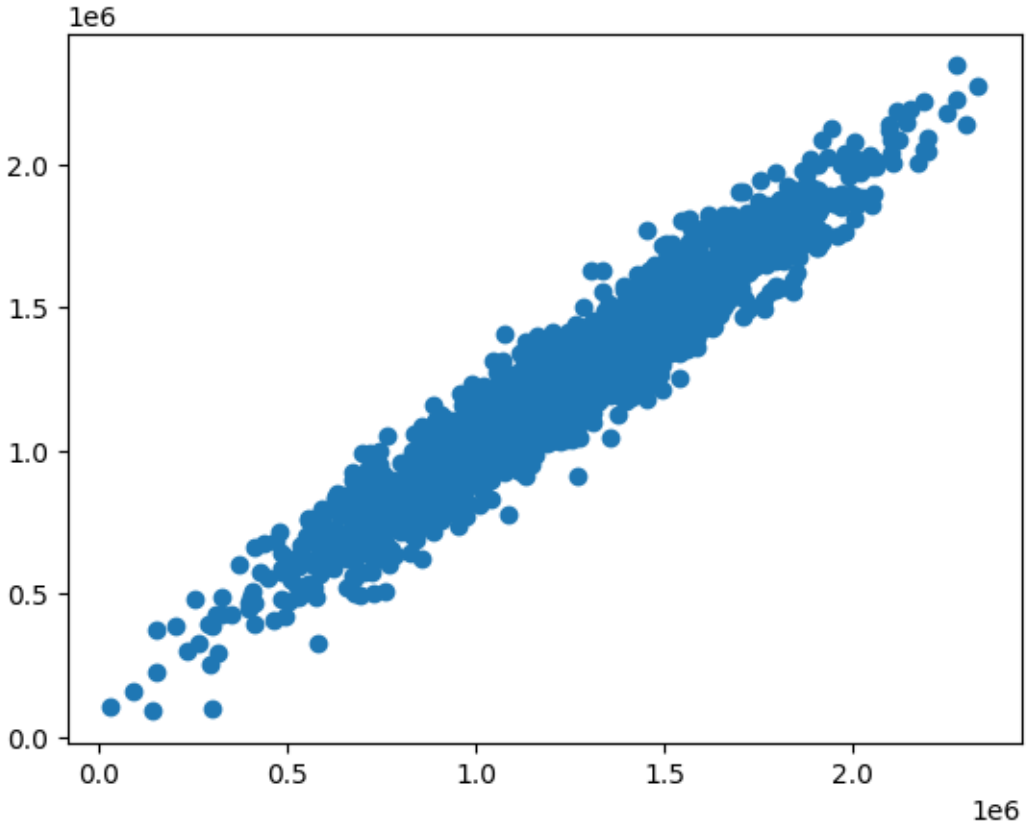
[ ]:
```
lm = LinearRegression()
lm.fit(X_train,y_train)
```

[ ]:
```
LinearRegression()
```

[ ]:
```
predictions = lm.predict(X_test)
```

[ ]:
```
plt.scatter(y_test,predictions)
```

[ ]: <matplotlib.collections.PathCollection at 0x21dac72e750>



0.0.4 3. Outout

[ ]:
```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 82288.22251914942
MSE: 10460958907.20898
RMSE: 102278.82922290899
```