ML_LinearRegression

February 9, 2023

[1]:
```
import numpy as np
import pandas as pd
```

[2]:
```
import matplotlib.pyplot as plt
import seaborn as sns
```

[3]:
```
%matplotlib inline
```

[4]:
```
house=pd.read_csv('USA_Housing.csv')
```

[5]:
```
house.head(3)
```

[5]:
```
   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

   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

                                             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…
```

[6]:
```
house.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
```

[ ]:

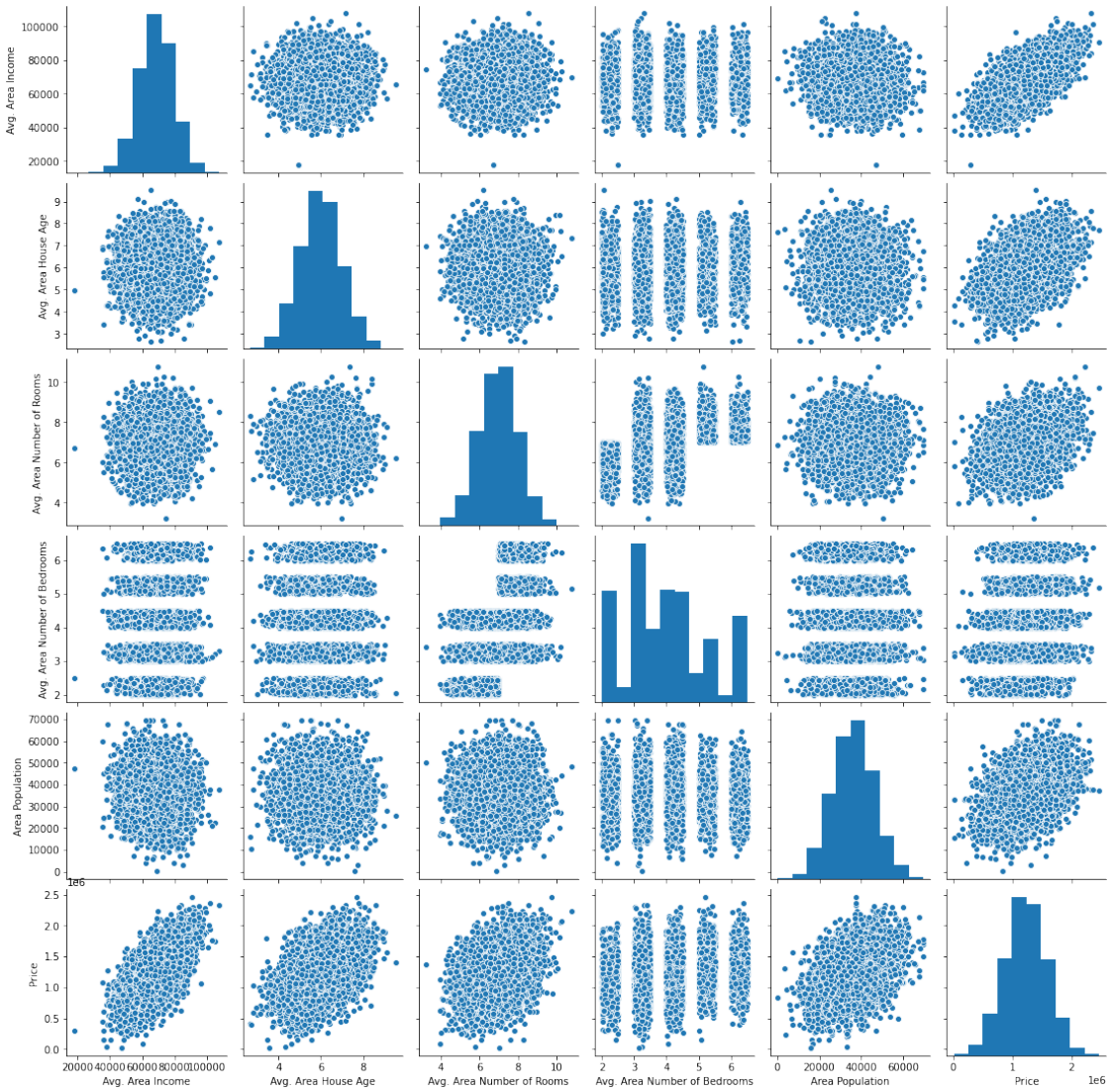
[7]:
```
house.describe()
```

[7]:

	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

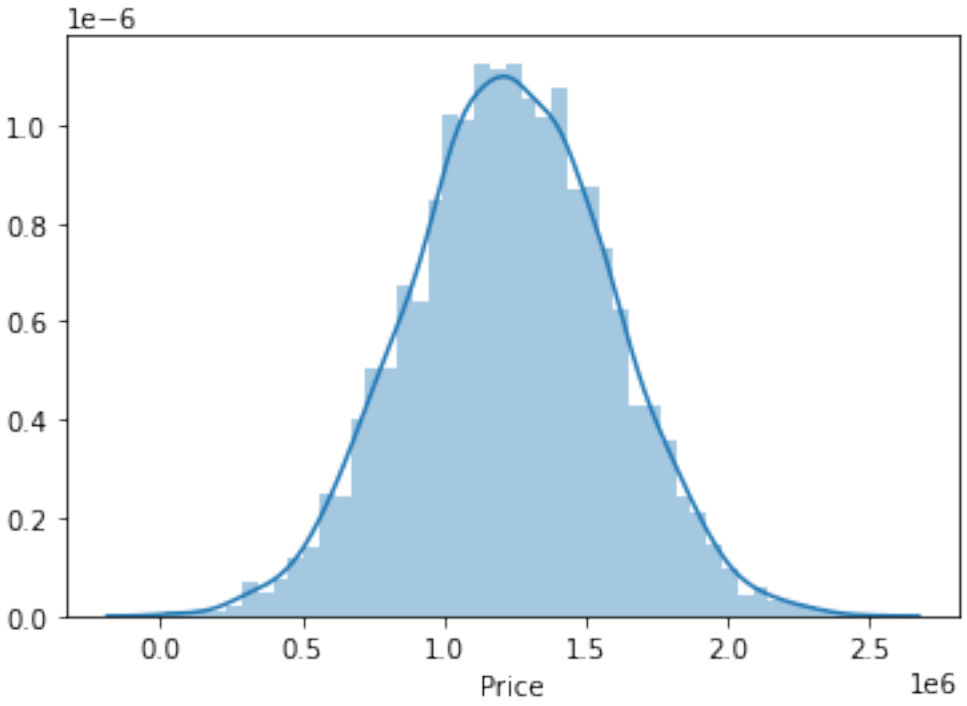
[8]:
```
sns.pairplot(house)
```

[8]:
```
<seaborn.axisgrid.PairGrid at 0x7f58bbebf8d0>
```



[9]:
```
sns.distplot(house['Price'])
```

[9]: <AxesSubplot:xlabel='Price'>



[10]:
```
house.corr()
```

[10]:

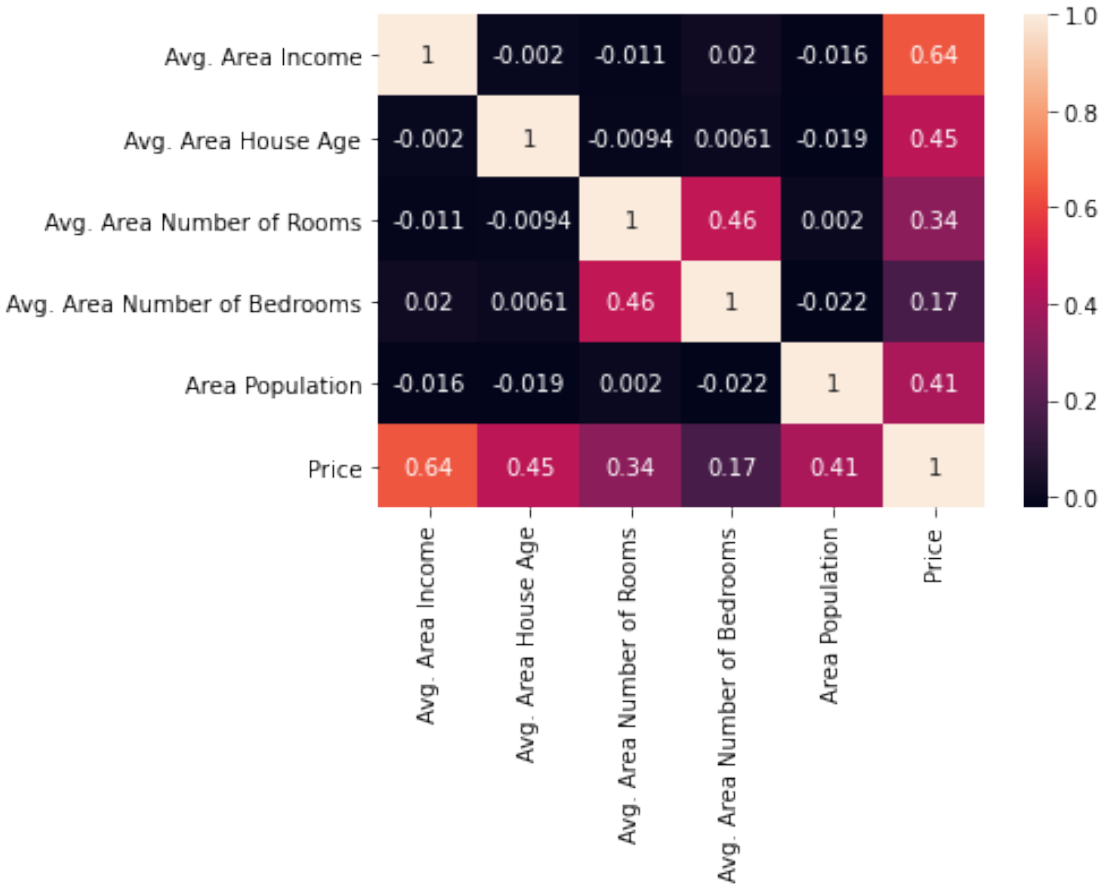
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-0.009428	1.000000	0.462695	0.002040	0.335664
Avg. Area Number of Bedrooms	0.019788	0.006149	0.462695	1.000000	-0.022168	
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	
Price	0.639734	0.452543	0.335664			

```
Price                                          0.171071          0.408556
```

```
                               Price
Avg. Area Income            0.639734
Avg. Area House Age         0.452543
Avg. Area Number of Rooms   0.335664
Avg. Area Number of Bedrooms  0.171071
Area Population             0.408556
Price                       1.000000
```

[11]:
```
sns.heatmap(house.corr(),annot=True)
```

[11]:
```
<AxesSubplot:>
```



[13]:
```
house.columns
```

[13]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

[ ]:

[14]:
```
X=house[['Avg. Area Income', 'Avg. Area House Age','Avg. Area Number of␣
↪Rooms','Avg. Area Number of Bedrooms', 'Area Population']]
```

[15]:
```
y=house[['Price']]
```

[16]:
```
from sklearn.model_selection import train_test_split
```

[17]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4,␣
↪random_state=101)
```

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

[19]:
```
lm=LinearRegression()
```

[20]:
```
lm.fit(X_train,y_train)
```

[20]:
```
LinearRegression()
```

[21]:
```
print(lm.intercept_)
```

```
[-2640159.79685191]
```

[22]:
```
lm.coef_
```

[22]:
```
array([[2.15282755e+01, 1.64883282e+05, 1.22368678e+05, 2.23380186e+03,
        1.51504200e+01]])
```

[23]:
```
lmcoeftransp = np.transpose(lm.coef_)
```

[24]:
```
lmcoeftransp
```

[24]:
```
array([[2.15282755e+01],
       [1.64883282e+05],
       [1.22368678e+05],
       [2.23380186e+03],
       [1.51504200e+01]])
```

[25]:
```
X.columns
```

[25]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
      dtype='object')
```

[26]:
```
X_train.columns
```

[26]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
      dtype='object')
```

[27]:
```
cdf = pd.DataFrame(lmcoeftransp,X.columns,columns=['Coeff'])
```

[28]:
```
cdf
```

[28]:

	Coeff
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

PREDICTIONS

[29]:
```
predictions = lm.predict(X_test)
```

[30]:
```
predictions
```

[30]:
```
array([[1260960.70567626],
       [ 827588.75560352],
       [1742421.24254328],
       …,
       [ 372191.40626952],
       [1365217.15140895],
       [1914519.54178824]])
```

[31]:
```
y_test
```

[31]:

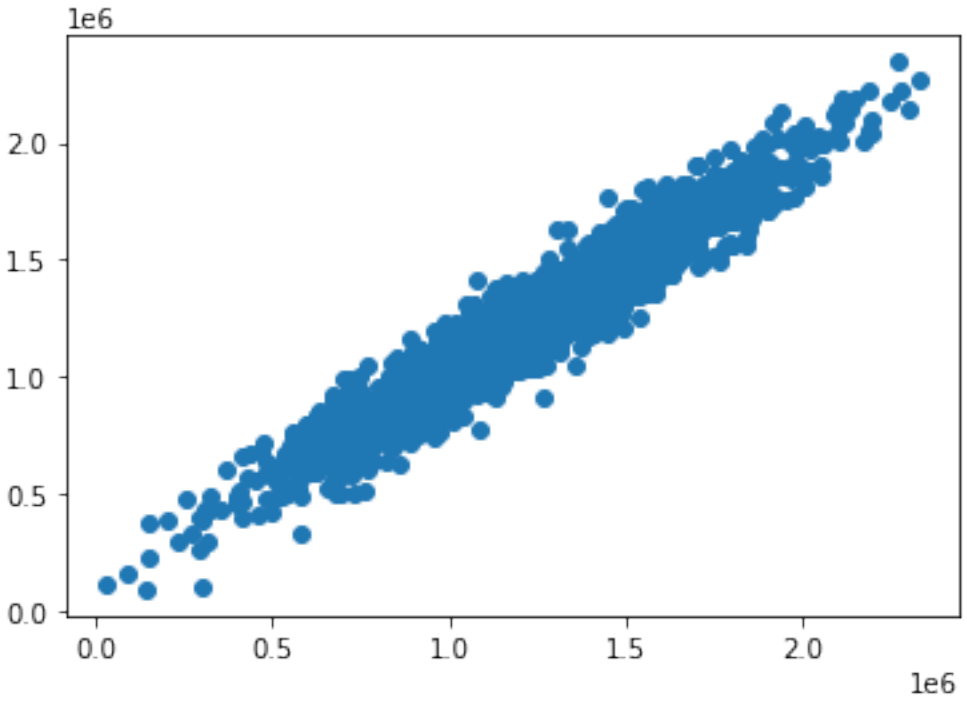
	Price
1718	1.251689e+06
2511	8.730483e+05
345	1.696978e+06
2521	1.063964e+06
54	9.487883e+05
…	…
1776	1.489520e+06
4269	7.777336e+05
1661	1.515271e+05
2410	1.343824e+06
2302	1.906025e+06

[2000 rows x 1 columns]

[32]:
```
#Now we wanna know how far off we are from real dataset
```

[33]:
```
plt.scatter(y_test,predictions)
```

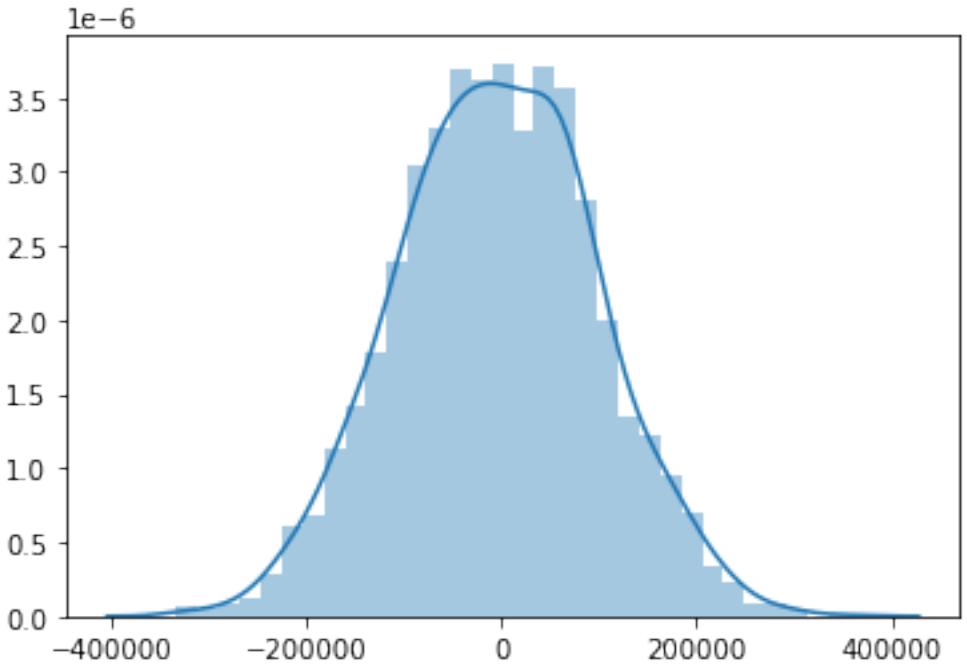
[33]:
```
<matplotlib.collections.PathCollection at 0x7f58b58b3d50>
```



[34]:
```
#Now we need to see through hist for a residue - what is a residue?
```

[35]:
```
sns.distplot((y_test - predictions))
```

[35]:
```
<AxesSubplot:>
```



[36]:
```
#Evaluating metrics
```

[37]:
```
from sklearn import metrics
```

[38]:
```
metrics.mean_absolute_error(y_test,predictions)
```

[38]: 82288.22251914957

[39]:
```
metrics.mean_squared_error(y_test,predictions)
```

[39]: 10460958907.209501

[40]:
```
#RMSE
```

[46]:
```
RresultMSE = np.sqrt(10460958907.209501)
```

[47]:
```
RresultMSE
```

[47]: 102278.82922291153

[48]:
```
r2_score = lm.score(X_test,y_test)
print(r2_score*100,'%')
```

91.76824009649201 %

[ ]: