DAY-10

usa_house

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(r"C:\Users\user\Downloads\usa_house.csv")[0:500]
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\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
...	...	...	...	...	...	...	...
495	68738.365215	4.186458	5.987896	3.44	47445.359167	1.084945e+06	71654 Brown Summit Suite 194\nBonillaberg, VT ...
496	74277.719901	6.987280	3.236194	3.42	50233.790310	1.365081e+06	9835 Kimberly Street Suite 318\nMurphyview, ND...
497	81280.910557	5.875972	5.227988	4.12	34933.196294	1.239460e+06	9117 Abigail Island\nWest Thomas, ID 72774-3622
498	86305.365141	8.064453	7.203916	4.17	37854.373451	2.056693e+06	272 King Mountain Suite 538\nNew Jennifer, NC ...
499	59302.229576	6.790822	6.431430	3.40	25296.735833	1.045395e+06	723 Kimberly Common Suite 520\nEast Nicholasha...

500 rows × 7 columns

In [3]:
```
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

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	500.000000	500.000000	500.000000	500.000000	500.000000	5.000000e+02
mean	68115.407048	6.047905	6.970575	3.991820	35697.384306	1.226058e+06
std	10346.604982	0.994378	1.047597	1.231015	9902.877557	3.464790e+05
min	17796.631190	3.690891	3.236194	2.000000	172.610686	1.520719e+05
25%	61026.998724	5.360325	6.282778	3.157500	29389.511953	1.011227e+06
50%	68491.556863	6.030302	7.036324	4.065000	35759.645663	1.225956e+06
75%	75212.992120	6.774867	7.659252	4.490000	42199.318047	1.449365e+06
max	97112.361252	8.562611	9.710217	6.500000	69575.449464	2.469066e+06

In [5]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 7 columns):
 #   Column                       Non-Null Count  Dtype  
---  ------                       --------------  -----  
 0   Avg. Area Income             500 non-null    float64
 1   Avg. Area House Age          500 non-null    float64
 2   Avg. Area Number of Rooms    500 non-null    float64
 3   Avg. Area Number of Bedrooms 500 non-null    float64
 4   Area Population              500 non-null    float64
 5   Price                        500 non-null    float64
 6   Address                      500 non-null    object 
dtypes: float64(6), object(1)
memory usage: 27.5+ KB
```

In [6]:
```
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 [9]:
```
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 [10]:
```
#to split my dataset into traning and test data

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 [11]:
```
from sklearn.linear_model import LinearRegression

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

Out[11]:
```
LinearRegression()
```

In [12]:
```
print(lr.intercept_)
```

```
-2653212.4280222673
```

In [13]:
```
print(lr.score(x_test,y_test))
```

```
0.9021848929718427
```

In [14]:
```
lr.score(x_train,y_train)
```

Out[14]:
```
0.9193350208850604
```

Ridge Regression

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]:
```
0.901404497255129
```

Lasso Regression

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]:
```
0.9021831728692513
```

In [20]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[20]:
```
ElasticNet()
```

In [21]:
```
print(en.intercept_)
```

```
-2036824.9205533208
```

In [22]:
```
predict=(en.predict(x_test))
```

In [23]:
```
print(en.score(x_test,y_test))
```

```
0.8561754618720676
```

Evaluation Matrics

In [24]:
```
from sklearn import metrics
```

In [25]:
```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,predict))
```

```
Mean Absolute Error: 112551.26411409823
```

In [26]:
```
print("Mean Square Error:",metrics.mean_squared_error(y_test,predict))
```

```
Mean Square Error: 18702156449.51369
```

In [27]:
```
print("Root Mean Square Error:",np.sqrt(metrics.mean_squared_error(y_test,predict)))
```

```
Root Mean Square Error: 136755.82784478946
```

In [ ]:
```

```