kaviyadevi 20106064

DATA PROCESSING

In [6]:
```
#to import libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:
```
#to import dataset
data=pd.read_csv(r"C:\Users\user\Downloads\10_USA_Housing - 10_USA_Housing.csv")
data
```

Out[7]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addres
0	79545.45857	5.682861	7.009188	4.09	23086.80050	1.059034e+06	208 Michael Ferry Ap 674\nLaurabury, N 3701.
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06	188 Johnson View Suite 079\nLak Kathleen, CA.
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabet Stravenue\nDanieltowr WI 06482.
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USS Barnett\nFPO A 4482
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\nFP(AE 0938
...	...	...	...	...	...	...	.
4995	60567.94414	7.830362	6.137356	3.46	22837.36103	1.060194e+06	USNS Williams\nFP(AP 30153-765
4996	78491.27543	6.999135	6.576763	4.02	25616.11549	1.482618e+06	PSC 9258, Bo 8489\nAPO AA 42991 335
4997	63390.68689	7.250591	4.805081	2.13	33266.14549	1.030730e+06	4215 Tracy Garde Suite 076\nJoshualanc VA 01.
4998	68001.33124	5.534388	7.130144	5.44	42625.62016	1.198657e+06	USS Wallace\nFPO A 7331
4999	65510.58180	5.992305	6.792336	4.07	46501.28380	1.298950e+06	37778 George Ridge Apt. 509\nEast Holl NV 2.

5000 rows × 7 columns

In [8]:
```
#to display top 5 rows
data.head()
```

Out[8]:

	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	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.64245	6.002900	6.730821	3.09	40173.07217	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.06718	5.865890	8.512727	5.13	36882.15940	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.24005	7.188236	5.586729	3.26	34310.24283	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.19723	5.040555	7.839388	4.23	26354.10947	6.309435e+05	USNS Raymond\nFPO AE 09386

DATA CLEANING AND PREPROCESSING

In [9]:
```
#
data.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 [10]:
```
#to display summary of statistics(here to know min max value)
data.describe()
```

Out[10]:

	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%	61480.562390	5.322283	6.299250	3.140000	29403.928700	9.975771e+05
50%	68804.286405	5.970429	7.002902	4.050000	36199.406690	1.232669e+06
75%	75783.338665	6.650808	7.665871	4.490000	42861.290770	1.471210e+06
max	107701.748400	9.519088	10.759588	6.500000	69621.713380	2.469066e+06

In [11]:
```
#to display the column heading
data.columns
```

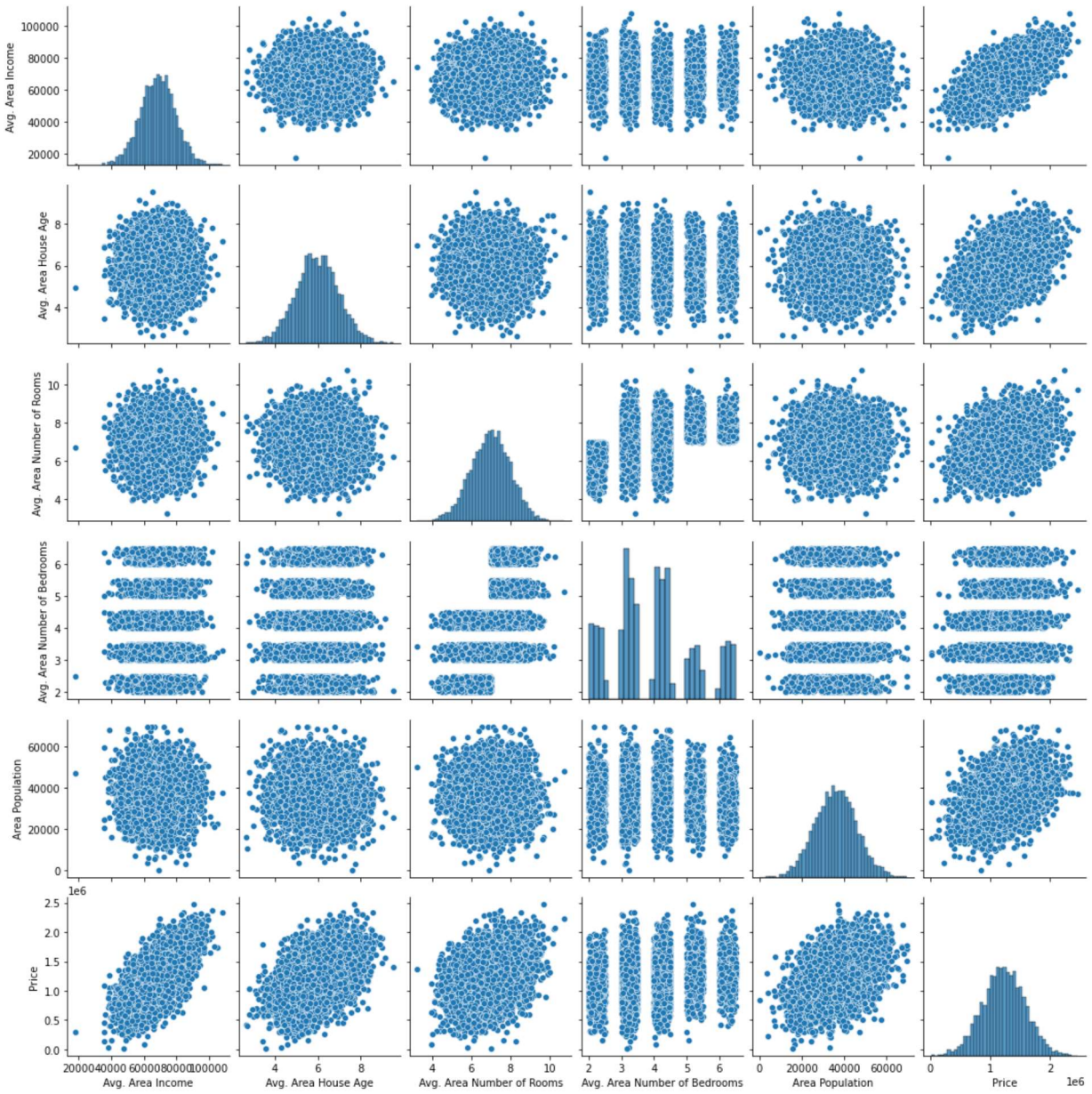
Out[11]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],
      dtype='object')
```

In [12]:
```
#here there is no missing values (identified through info() 5000 data are descrit
```

EDA and DATA VISUALIZATION

In [13]:
```
sns.pairplot(data)
```

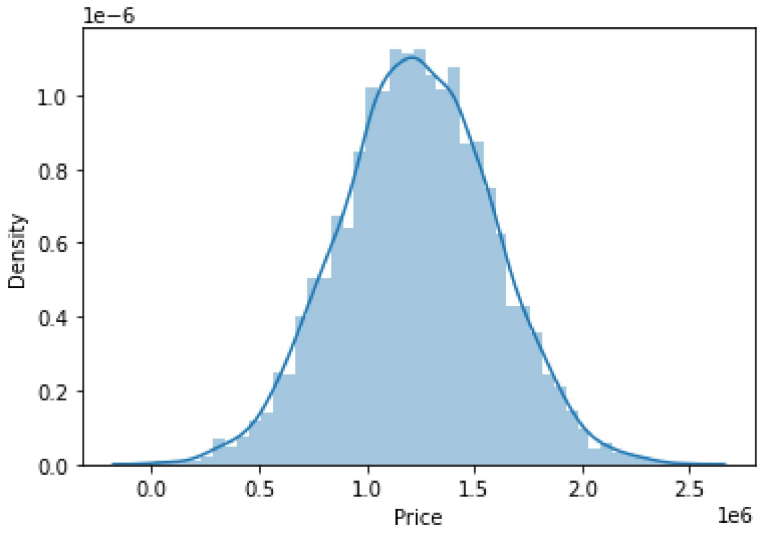
Out[13]:
```
<seaborn.axisgrid.PairGrid at 0x2391fdeb760>
```



In [14]:
```
sns.distplot(data["Price"])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

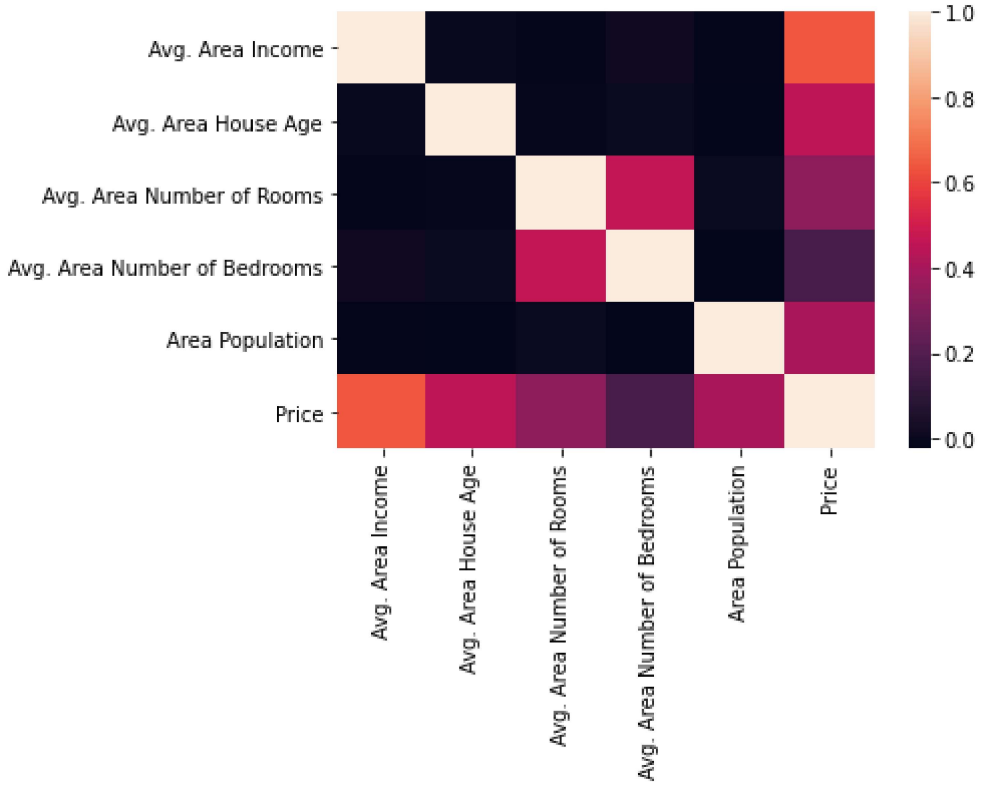
Out[14]: <AxesSubplot:xlabel='Price', ylabel='Density'>



In [15]:
```
df=data[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price']]
```

In [16]:
```
sns.heatmap(df.corr())
```

Out[16]: <AxesSubplot:>



TO TRAIN MODEL

MODEL BUILDING We are going to train linear regression model; we need to split out the data into two variables x and y where x is independent variables (input) and y is dependent on x(output) we could ignore address column as it is not required for our model

In [17]:
```
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 [18]:
```
#to split my dataset into trainning and test

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 [19]:
```
from sklearn.linear_model import LinearRegression

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

Out[19]:
```
LinearRegression()
```

In [20]:
```
#to find intercept
print(lr.intercept_)
```

```
-2631179.446847313
```

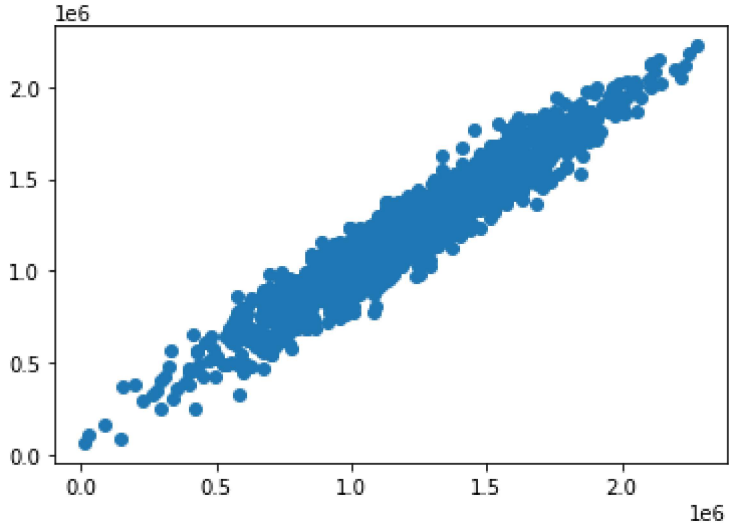
In [21]:
```
coeff = pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[21]:

	Co-efficient
Avg. Area Income	21.479593
Avg. Area House Age	165312.826052
Avg. Area Number of Rooms	121223.545008
Avg. Area Number of Bedrooms	2293.701818
Area Population	15.118977

In [22]:
```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[22]: <matplotlib.collections.PathCollection at 0x23922fedbb0>



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

```
0.9196122061704285
```

RIDGE AND LASSO REGRESSION

In [24]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [25]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[25]: Ridge(alpha=10)

In [26]:
```
rr.score(x_test,y_test)
```

Out[26]: 0.9196108618574063

In [27]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[27]:
```
Lasso(alpha=10)
```

In [28]:
```
la.score(x_test,y_test)
```

Out[28]:
```
0.9196126427939018
```

In [29]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]:
```
ElasticNet()
```

In [31]:
```
print(en.coef_)
```

```
[2.13485855e+01 1.08956510e+05 7.60150692e+04 1.48830865e+04
 1.49596622e+01]
```

In [33]:
```
print(en.predict(x_test))
```

```
[1233826.38851369 1039013.51432593 1449245.88937301 ... 1230827.74917279
 1120198.45040024 1159902.0622341 ]
```

In [34]:
```
print(en.score(x_test,y_test))
```

```
0.8832134954458345
```

Evaluation metrics

In [35]:
```
from sklearn import metrics
```

In [37]:
```
print("Mean Absolute error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute error 82122.1782559458
```

In [38]:
```
print("Mean Squared error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared error 10434724665.834866
```

In [41]:
```
print("Root Mean Absolute error",np.sqrt(metrics.mean_squared_error(y_test,predic
```

```
Root Mean Absolute error 102150.50007628385
```