problem statement t:

A real estate agent want help to preedict the house price for regions in USA . He gave us the dataset to work on to use Linear regression model. Create a model that helps hom to estimate of what the house would sell sell for

Data collection

In [1]:

```
# import libraries
import numpy as np # data cleaning and collection
import pandas as pd # ""  ""
import matplotlib.pyplot as py # visualization
import seaborn as sns #" "  ""
```

In [2]:

```
d=pd.read_csv(r"C:\Users\user\Downloads\USA_Housing.csv")
d
```

Out[2]:

	Avg. Area_Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.45857	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
...	...	...	...	...	...	...	...
4995	60567.94414	7.830362	6.137356	3.46	22837.36103	1.060194e+06	USNS Williams\nFPO AP 30153-7653
4996	78491.27543	6.999135	6.576763	4.02	25616.11549	1.482618e+06	PSC 9258, Box 8489\nAPO AA 42991-3352
4997	63390.68689	7.250591	4.805081	2.13	33266.14549	1.030730e+06	4215 Tracy Garden Suite 076\nJoshualand, VA 01...

	Avg. Area_Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
4998	68001.33124	5.534388	7.130144	5.44	42625.62016	1.198657e+06	USS Wallace\nFPO AE 73316
4999	65510.58180	5.992305	6.792336	4.07	46501.28380	1.298950e+06	37778 George Ridges Apt. 509\nEast Holly, NV 2...

5000 rows × 7 columns

In [3]:

```
d.head()
```

Out[3]:

	Avg. Area_Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.45857	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

In [4]:

```
d.info() #informa
```

```
<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 [5]:

```
#to display summary of statistics
d.describe()
```

Out[5]:

	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 [6]:

```
#to display column heading
d.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 [7]:

```
d.index
```

Out[7]:

```
RangeIndex(start=0, stop=5000, step=1)
```

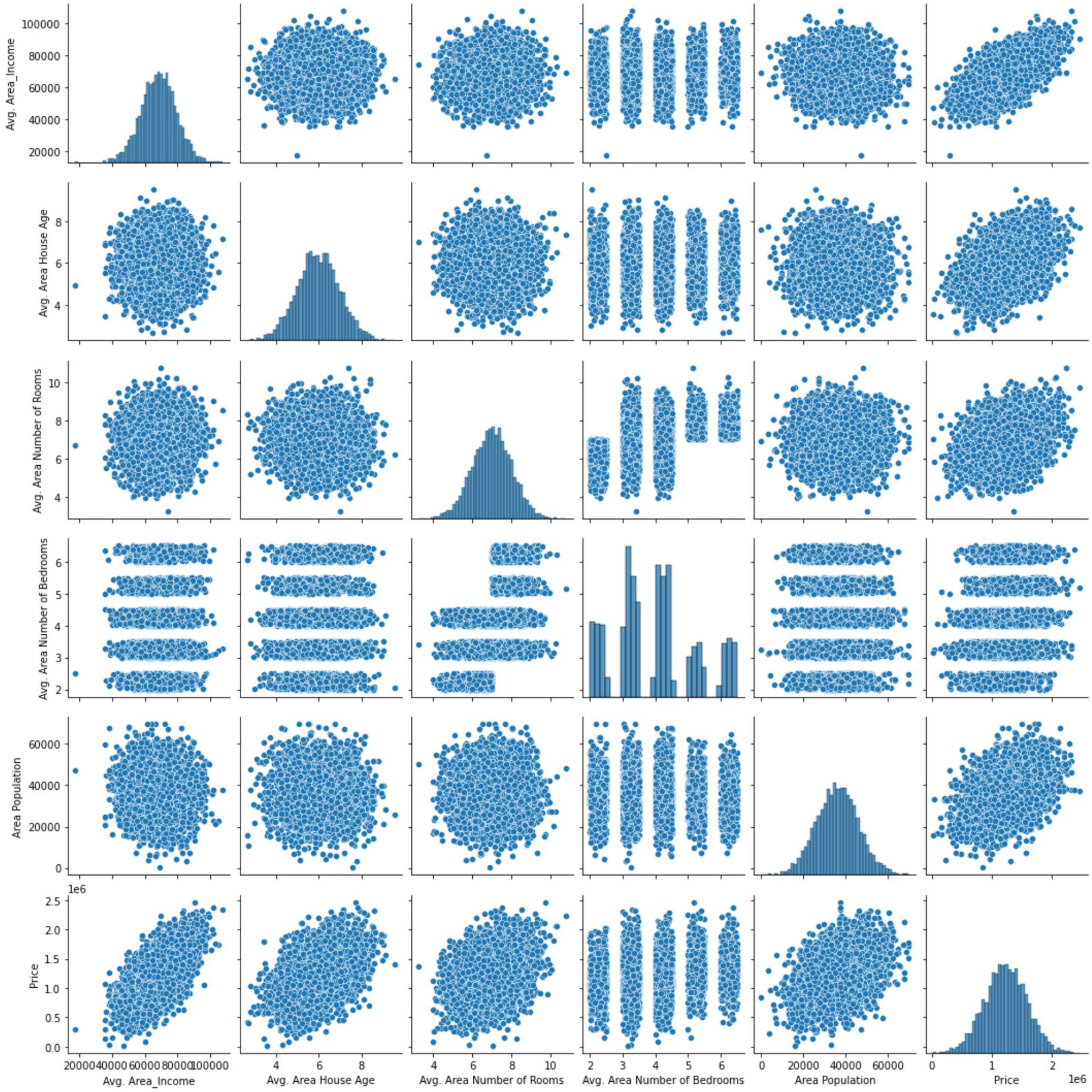
EDA and visualization

In [8]:

```
sns.pairplot(d)
```

Out[8]:

```
<seaborn.axisgrid.PairGrid at 0x17f28628bb0>
```



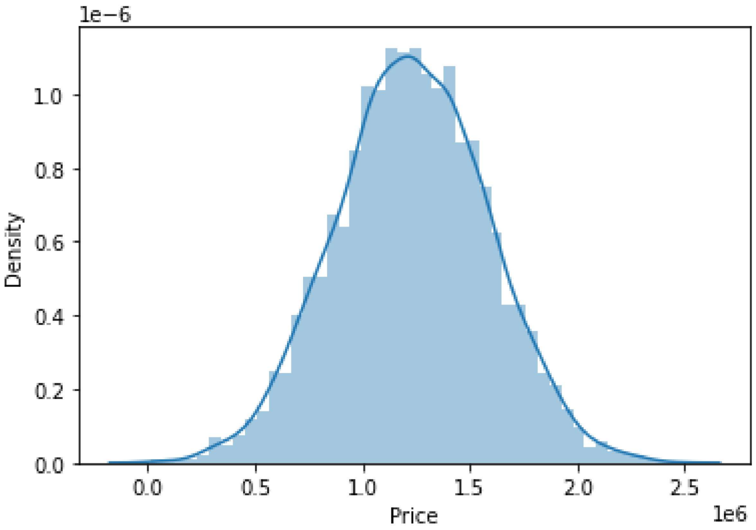
In [9]:

```
sns.distplot(d['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 adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

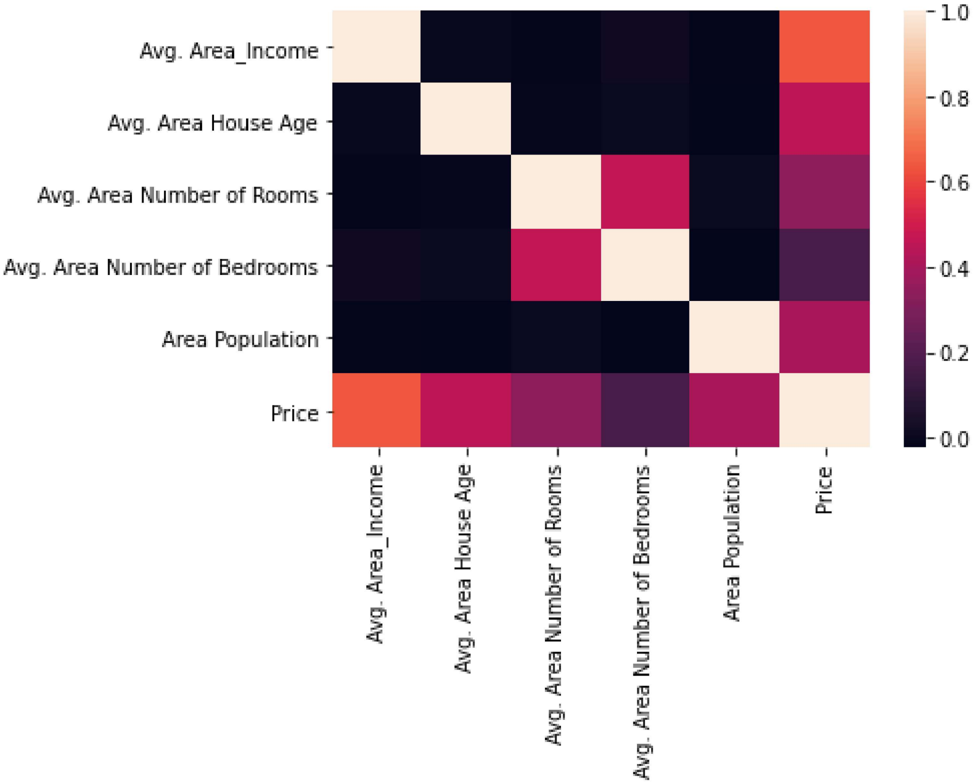
Out[9]:

```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



```
In [10]:
d1=d[['Avg. Area_Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address']]
sns.heatmap(d1.corr())
```

```
Out[10]: <AxesSubplot:>
```



to train the model - model building

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

In [11]:

```
#x-input,y-output
x=d1[['Avg. Area_Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
y=d1['Price']
```

In [12]:

```
#to split my dataset into training 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 [13]:

```
from sklearn.linear_model import LinearRegression
```

In [14]:

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

Out[14]:

```
LinearRegression()
```

In [15]:

```
print(lr.intercept_)
```

```
-2627404.1891406737
```

In [16]:

```
# y=mx+c , coeff=m
coeff =pd.DataFrame(lr.coef_,x.columns,columns=["Co-efficient"])
coeff
```

Out[16]:

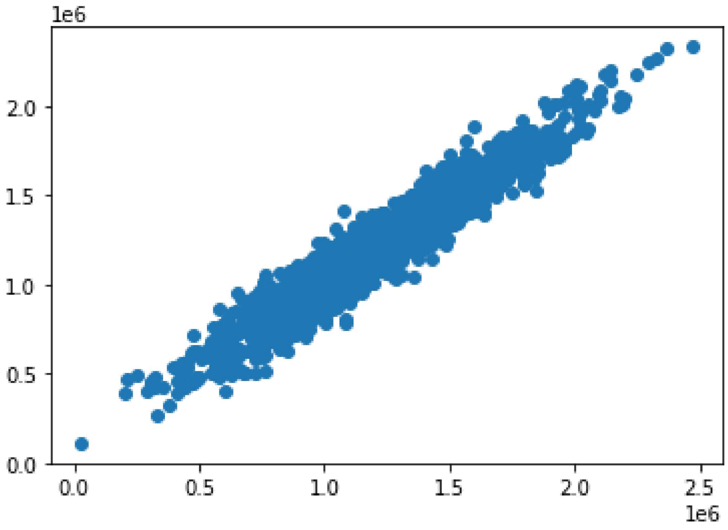
	Co-efficient
Avg. Area_Income	21.415542
Avg. Area House Age	165962.609867
Avg. Area Number of Rooms	119394.519811
Avg. Area Number of Bedrooms	2806.514707
Area Population	15.319576

In [17]:

```
prediction =lr.predict(x_test)
py.scatter(y_test,prediction)
```

Out[17]:

```
<matplotlib.collections.PathCollection at 0x17f2bf91fa0>
```



In [18]:

```
print(lr.score(x_test,y_test))
```

```
0.9176019291991886
```

In [19]:

```
print(lr.score(x_train,y_train))
```

```
0.9181168859388773
```

In [20]:

```
from sklearn.linear_model import Ridge,Lasso
```

In [21]:

```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[21]:

```
Ridge(alpha=10)
```

In [22]:

```
rr.score(x_test,y_test)
```

Out[22]:

```
0.9175841703573315
```

In [23]:

```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[23]:

```
Lasso(alpha=10)
```

In [24]:

```
la.score(x_test,y_test)
```

Out[24]:

```
0.9176020277657732
```

In [25]:

```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[25]:

```
ElasticNet()
```

In [26]:

```
print(en.coef_)
```

```
[2.12135675e+01 1.08975542e+05 7.51370003e+04 1.43564743e+04
 1.53595576e+01]
```

In [27]:

```
print(en.intercept_)
```

```
-2011619.4173096179
```

In [28]:

```
print(en.predict(x_test))
```

```
[1256720.3605405  1311611.68032796 1671766.35687042 ... 1497353.91061606
 1245011.71139054  939028.23742129]
```

In [29]:

```
print(en.score(x_test,y_test))
```

```
0.8782932216399257
```

In [30]:

```
from sklearn import metrics
```

In [31]:

```
print("Mean Absolute Error:",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error: 82375.96847878218
```

In [32]:

```
print("Mean Squared Error:",metrics.mean_squared_error(y_test,prediction))
```

```
Mean Squared Error: 10480055759.793833
```

In [33]:

```
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,prediction))
```

```
Root Mean Squared Error: 102372.14347562443
```

In [ ]: