In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df=pd.read_csv("Real_estates.csv")
df
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Ad
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Fer 674\nLaurabu 3
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Suite 079\ Kathleen
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Eliz Stravenue\nDanie WI 06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFF
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\ AE
...	...	...	...	...	...	...	
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06	USNS Williams\ AP 30153
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06	PSC 925 8489\nAF 42991
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06	4215 Tracy G Suite 076\nJoshu V.
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06	USS Wallace\nFF
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06	37778 George F Apt. 509\nEast N

5000 rows × 7 columns

drop unwanted columns

In [3]:
```
df.drop("Address",axis=1,inplace=True)
```

In [4]:
```
df
```

Out[4]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05
...	...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035	1.060194e+06
4996	78491.275435	6.999135	6.576763	4.02	25616.115489	1.482618e+06
4997	63390.686886	7.250591	4.805081	2.13	33266.145490	1.030730e+06
4998	68001.331235	5.534388	7.130144	5.44	42625.620156	1.198657e+06
4999	65510.581804	5.992305	6.792336	4.07	46501.283803	1.298950e+06

5000 rows × 6 columns

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 6 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
dtypes: float64(6)
memory usage: 234.5 KB
```

```
1) in the Real_estates dataset we have 6 columns and 5000 rows
2) price column in tagret column and rest of the columns are features
3)all columns have float datatype
```

Goal= the goal is to predict price of the house

In [6]:
```
df.describe()
```

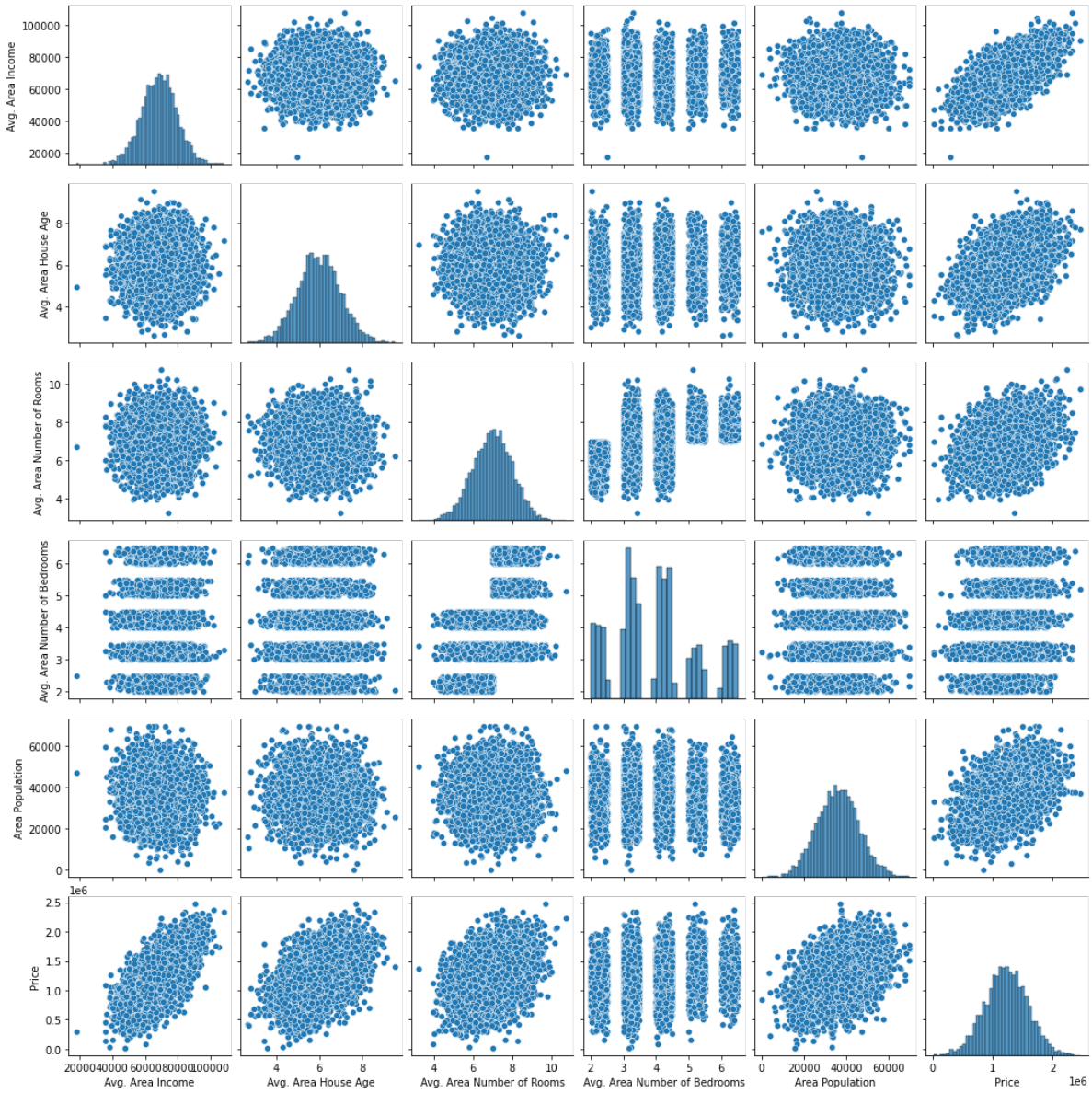
Out[6]:

	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

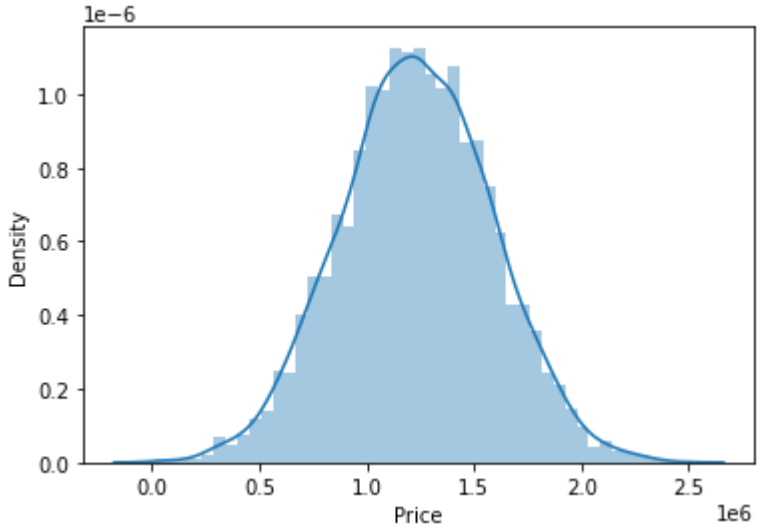
In [ ]:
```

```

In [7]:
```
sns.pairplot(df)
```

Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x15d6688a520>
```



In [8]:
```
sns.distplot(df["Price"])
plt.show()
```



In [9]:
```
df.corr().style.background_gradient()
```

Out[9]:

	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	0.171071
Area Population	-0.016234	-0.018743	0.002040	-0.022168	1.000000	0.408556
Price	0.639734	0.452543	0.335664	0.171071	0.408556	1.000000

OR

In [10]:
```
sns.heatmap(df.corr(),annot=True,cmap="rainbow")
```

Out[10]: <AxesSubplot:>

In [11]:
```
from scipy.stats import skew
```

In [12]:
```
skew(df["Avg. Area Income"])
```

Out[12]: -0.03370985856853668

In [13]:
```
skew(df["Avg. Area House Age"])
```

Out[13]: -0.007211708023735578

In [14]:
```
skew(df["Avg. Area Number of Rooms"])
```

Out[14]: -0.040983610381965664

In [15]:
```
skew(df["Avg. Area Number of Bedrooms"])
```

Out[15]: 0.37612751568905145

In [16]:
```
skew(df["Area Population"])
```

Out[16]: 0.05063448536127159

split the data into X and Y

In [17]:
```
x=df.iloc[:,:-1]
y=df.iloc[:,-1]
```

In [18]:
```
x
```

Out[18]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population
0	79545.458574	5.682861	7.009188	4.09	23086.800503
1	79248.642455	6.002900	6.730821	3.09	40173.072174
2	61287.067179	5.865890	8.512727	5.13	36882.159400
3	63345.240046	7.188236	5.586729	3.26	34310.242831
4	59982.197226	5.040555	7.839388	4.23	26354.109472
...	...	...	...	...	...
4995	60567.944140	7.830362	6.137356	3.46	22837.361035
4996	78491.275435	6.999135	6.576763	4.02	25616.115489
4997	63390.686886	7.250591	4.805081	2.13	33266.145490
4998	68001.331235	5.534388	7.130144	5.44	42625.620156
4999	65510.581804	5.992305	6.792336	4.07	46501.283803

5000 rows × 5 columns

In [19]:
```
y
```

Out[19]:
```
0       1.059034e+06
1       1.505891e+06
2       1.058988e+06
3       1.260617e+06
4       6.309435e+05
            ...
4995    1.060194e+06
4996    1.482618e+06
4997    1.030730e+06
4998    1.198657e+06
4999    1.298950e+06
Name: Price, Length: 5000, dtype: float64
```

Train test split

In [20]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_state=3)
```

Train model

In [21]:
```
from sklearn.linear_model import LinearRegression
linreg=LinearRegression()
linreg.fit(xtrain,ytrain)
ypred=linreg.predict(xtest)
```

In [22]:
```
linreg.coef_
```

Out[22]:
```
array([2.14670622e+01, 1.65516587e+05, 1.22191414e+05, 2.37449538e+03,
       1.50581938e+01])
```

In [23]:
```
pd.DataFrame(linreg.coef_,index=x.columns,columns=["Coeffcient"])
```

Out[23]:

	Coeffcient
Avg. Area Income	21.467062
Avg. Area House Age	165516.587350
Avg. Area Number of Rooms	122191.414467
Avg. Area Number of Bedrooms	2374.495377
Area Population	15.058194

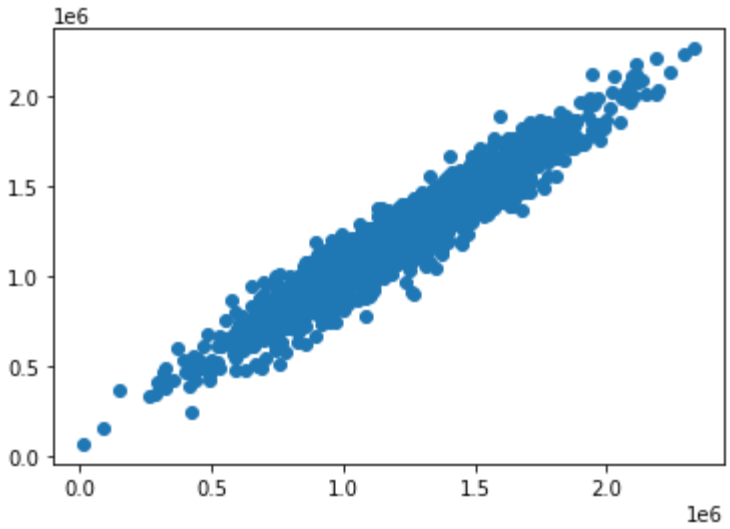
For every unit growth in X, we estimate that y will grow by M

Interpreting the Coefficient

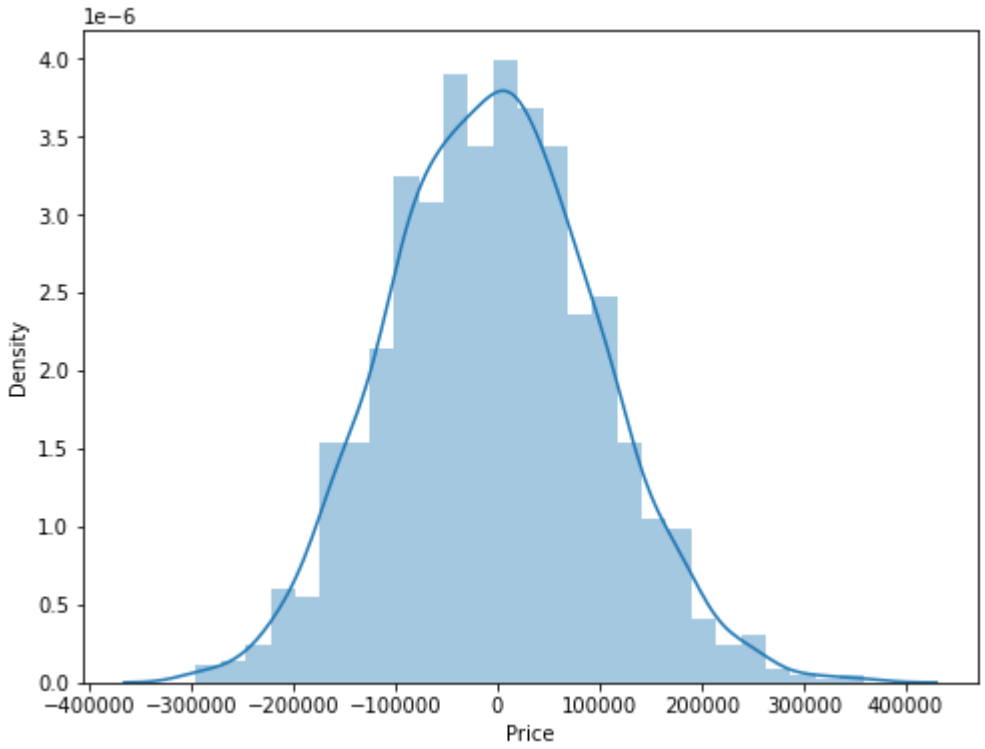
In [24]:
```
# Holding all other features fixed, a 1 unit incresed in Avg.Area Income is as
# Holding all other features fixed, a 1 unit incresed in Avg.Area House Age is
# Holding all other features fixed, a 1 unit incresed in Avg.Area Number of Ro
# Holding all other features fixed, a 1 unit incresed in Avg. Number of Bedroo
# Holding all other features fixed, a 1 unit incresed in Area Population is as
```

In [25]:
```
plt.scatter(ytest,ypred)
```

Out[25]:
```
<matplotlib.collections.PathCollection at 0x15d6a436af0>
```



In [26]:
```
plt.figure(figsize=(8,6))
sns.distplot((ytest-ypred))
plt.show()
```



Regression Evaluation Metrics

In [27]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score

mse=mean_squared_error(ytest,ypred)
mae=mean_absolute_error(ytest,ypred)
rmse=np.sqrt(mse)
r2=r2_score(ytest,ypred)

print(f"MAE={mae}\nMSE={mse}\nRMSE={rmse}\nAccuracy={r2}")
```

```
MAE=81088.0010534593
MSE=10277027976.08512
RMSE=101375.67743835362
Accuracy=0.9156495834619673
```

Testing our model on new observation

In [28]:
```
def predictprice(aai,aaha,aanr,aanb,ap):
    newob=[[aai,aaha,aanr,aanb,ap]]
    yp=linreg.predict(newob)[0]
    print(f"The price of your dream house is $={yp}")
    return yp
```

In [29]:
```
predictprice(68001.331235,5.534388,7.130144,5.44,42625.620156)
```

```
The price of your dream house is $=1265589.3520344673
```

Out[29]:
```
1265589.3520344673
```

Reguralization

In [30]:
```
train=linreg.score(xtrain,ytrain)
test=linreg.score(xtest,ytest)

print(f"Training Accuracy={train}")
print(f"Testing Accuracy={test}")
```

```
Training Accuracy=0.9188688514537523
Testing Accuracy=0.9156495834619673
```

In [31]:
```
from sklearn.linear_model import Ridge,Lasso
l2=Ridge(alpha=4)
l2.fit(xtrain,ytrain)
yp=l2.predict(xtest)

train=linreg.score(xtrain,ytrain)
test=linreg.score(xtest,ytest)

print(f"Training Accuracy={train}")
print(f"Testing Accuracy={test}")
```

```
Training Accuracy=0.9188688514537523
Testing Accuracy=0.9156495834619673
```

In [32]:
```
l1=Lasso(alpha=4)
l1.fit(xtrain,ytrain)
l1.predict(xtest)

train=linreg.score(xtrain,ytrain)
test=linreg.score(xtest,ytest)

print(f"Training Accuracy={train}")
print(f"Testing Accuracy={test}")
```

```
Training Accuracy=0.9188688514537523
Testing Accuracy=0.9156495834619673
```

In [ ]: