Assignment 4: Regression

A. Simple Linear Regression

a) Import the dataset in Python environment.

In [1]:

```
import numpy as np
import pandas as pd
import seaborn as sb
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [2]:

```
df = pd.read_csv('USA_Housing.csv')
```

b) Make a new dataset considering only 'Avg. Area House Age'; and 'Price' attributes.

In [3]:

```
df_binary = df[['Avg. Area House Age', 'Price']].copy()
df_binary.columns = ['Avg. Area House Age', 'Price']
df_binary.head()
```

Out[3]:

	Avg. Area House Age	Price
0	5.682861	1.059034e+06
1	6.002900	1.505891e+06
2	5.865890	1.058988e+06
3	7.188236	1.260617e+06
4	5.040555	6.309435e+05

c) Set a variable X equal to the 'Avg. Area House Age'feature of the given dataset and a variable y equal to the "Price" column.

In [4]:

```
x=df['Avg. Area House Age']
y=df['Price']
x
```

Out[4]:

```
0       5.682861
1       6.002900
2       5.865890
3       7.188236
4       5.040555
          ...
4995    7.830362
4996    6.999135
4997    7.250591
4998    5.534388
4999    5.992305
Name: Avg. Area House Age, Length: 5000, dtype: float64
```

In [5]:

```
y
```

Out[5]:

```
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
```

d) Split the new dataset into the Training set and Test set such that Test set consists 1/3 of total records usingmodel_selection.train_test_split from sklearn.

In [6]:

```
X = np.array(df_binary['Avg. Area House Age']).reshape(-1, 1)
Y = np.array(df_binary['Price']).reshape(-1, 1)
df_binary.dropna(inplace = True)
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.33)
```

e) Train the Simple Linear Regression model based on the Training set after importing LinearRegression from sklearn.linear_model.

In [7]:

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

Out[7]:

```
LinearRegression()
```

f) Display the model's coefficients after train the model and hence write the final regression model.

In [8]:

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

```
0.19740199111771872
```

In [9]:

```
lr.coef_
```

Out[9]:

```
array([[161628.45007805]])
```

In [10]:

```
lr.intercept_
```

Out[10]:

```
array([264376.53968167])
```

In [11]:

```
m=lr.intercept_
b=lr.coef_
```

In [12]:

```
print("Y=",m,"*X+",b)
```

```
Y= [264376.53968167] *X+ [[161628.45007805]]
```

g) Predict the house price of the Test set data and display them.

In [13]:

```
pred=lr.predict(x_test)
```

In [14]:

```
print(pred)
```

```
[[1306425.03128295]
 [1256462.13592285]
 [1168569.88135383]
 ...
 [1175667.64798037]
 [1122970.9181027 ]
 [1085855.73974469]]
```

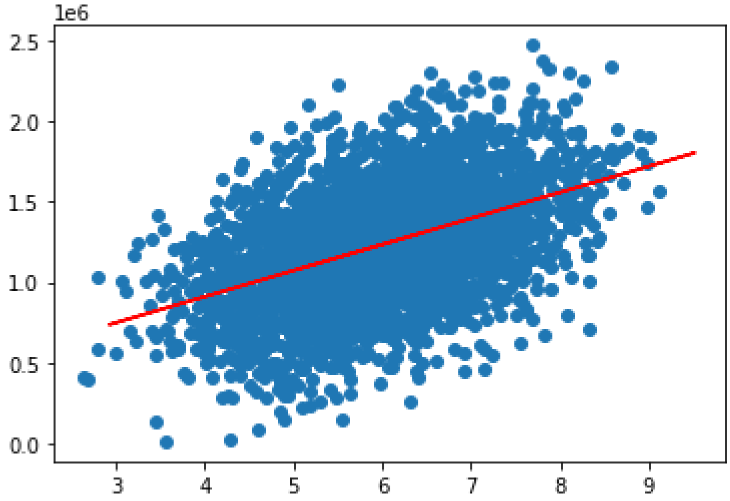
h) Visualize the Training set results using scatter plot along with regression line.

In [15]:

```
plt.scatter(x_train,y_train)
plt.plot(x_test,pred,color='red')
```

Out[15]:

```
[<matplotlib.lines.Line2D at 0x1b32b228670>]
```



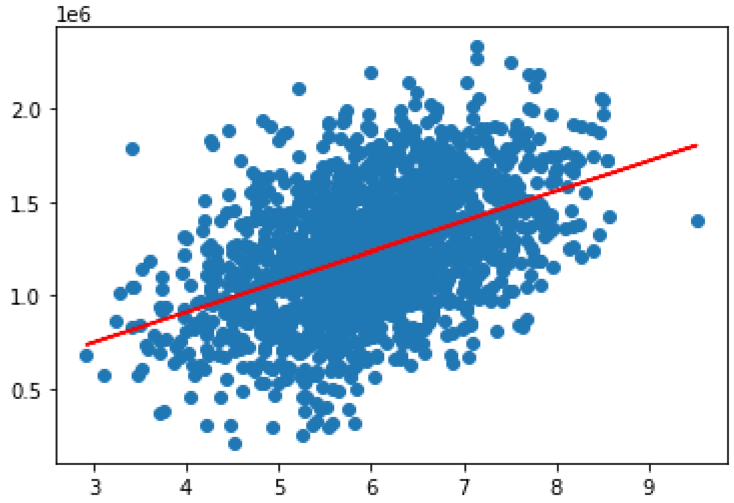
i) Visualize the Test set results using scatter plot along with regression line.

In [16]:

```
plt.scatter(x_test,y_test)
plt.plot(x_test,pred,color='red')
```

Out[16]:

```
[<matplotlib.lines.Line2D at 0x1b32d49c700>]
```



B. Multiple Linear Regression

a) Use the dataset which you have already imported in Python environment in Question a of Section A above.

In [17]:

```
df1=pd.read_csv("USA_Housing.csv")
```

b) Check the head of your Dataset, and also check out its info() and describe() methods over the dataset.

In [18]:

```
df1.head()
```

Out[18]:

	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 Elizabe Stravenue\nDanieltow 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

In [19]:

```
df1.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 [20]:

```
df1.describe()
```

Out[20]:

	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

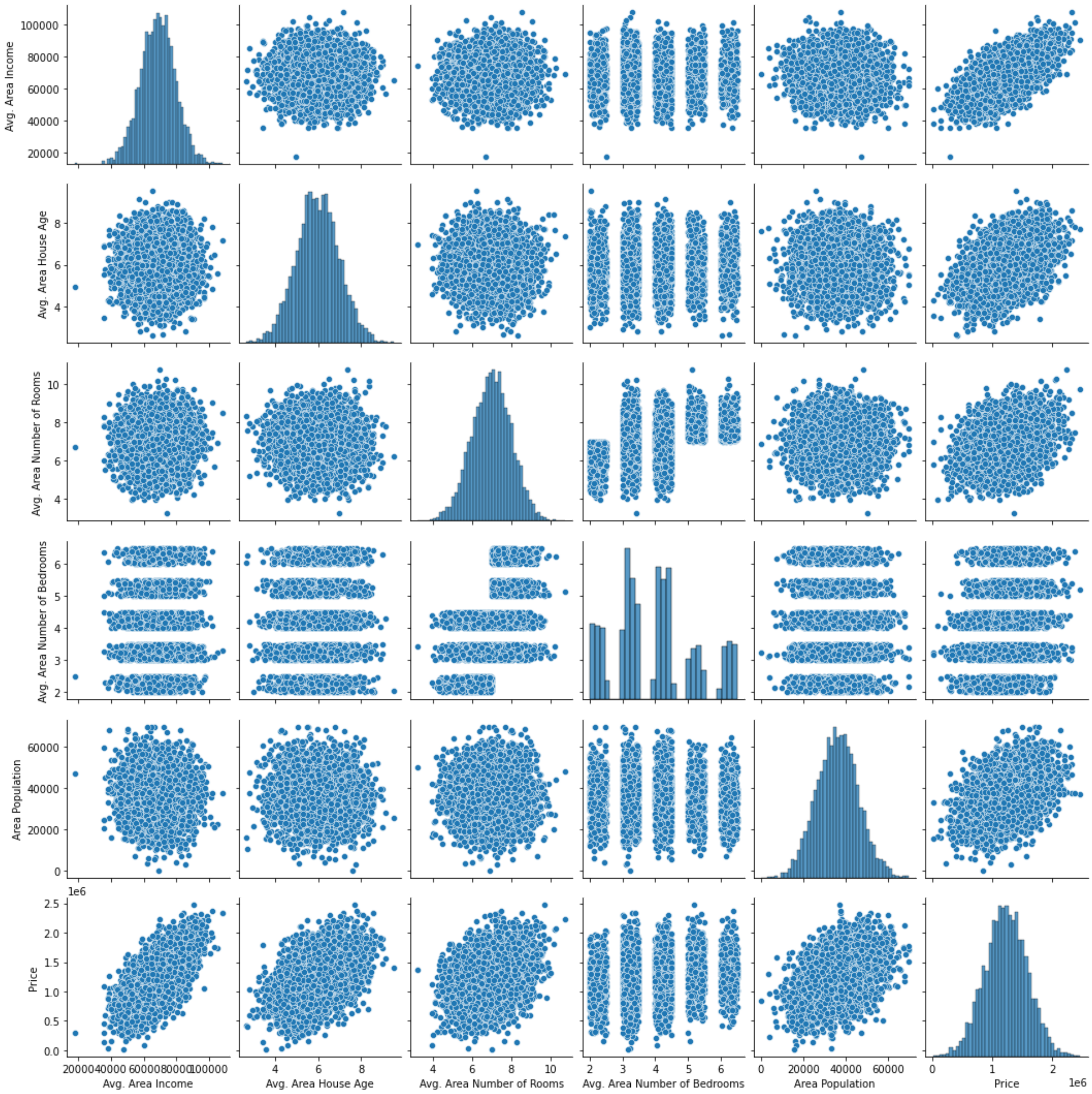
c) Explore the types of relationships across the entire data set using ‘pairplot’ method of seaborn and comment on that.

In [21]:

```
sb.pairplot(df1)
```

Out[21]:

```
<seaborn.axisgrid.PairGrid at 0x1b32d71d1f0>
```



d) Set a variable X equal to the numerical features of the given dataset and a variable y equal to the "Price"column.

In [22]:

```
X = df1[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
 'Avg. Area Number of Bedrooms', 'Area Population']]
Y=df1['Price']
```

e) Split the data into training and testing sets using model_selection.train_test_split from sklearn such that Test set consists 30% of total data.

In [23]:

```
x_train,x_test,y_train,y_test=train_test_split(X,Y,test_size=0.30)
```

f) Train the Linear Regression model based on the Training data after importing LinearRegression from sklearn.linear_model .

In [24]:

```
lm1=LinearRegression()
lm1.fit(x_train,y_train)
```

Out[24]:

```
LinearRegression()
```

g) Print the coefficients of the trained model. How can you interpret these coefficients?

In [25]:

```
print(lm1.score(x_test,y_test))
```

```
0.9204289208842987
```

In [26]:

```
print(lm1.intercept_)
```

```
-2631251.8071228913
```

In [27]:

```
coeff_df = pd.DataFrame(lm1.coef_, X.columns, columns=['Coefficient'])
coeff_df
```

Out[27]:

	Coefficient
Avg. Area Income	21.588120
Avg. Area House Age	165100.159433
Avg. Area Number of Rooms	119746.635081
Avg. Area Number of Bedrooms	2765.916748
Area Population	15.146601

h) Predict the house price of the Test set data and display them.

In [28]:

```
prediction=lm1.predict(x_test)
print(prediction)
```

```
[1225074.00144784 1349362.37857084  991051.61288649 ... 1098298.67538327
 1187409.46602538 1363950.36658695]
```

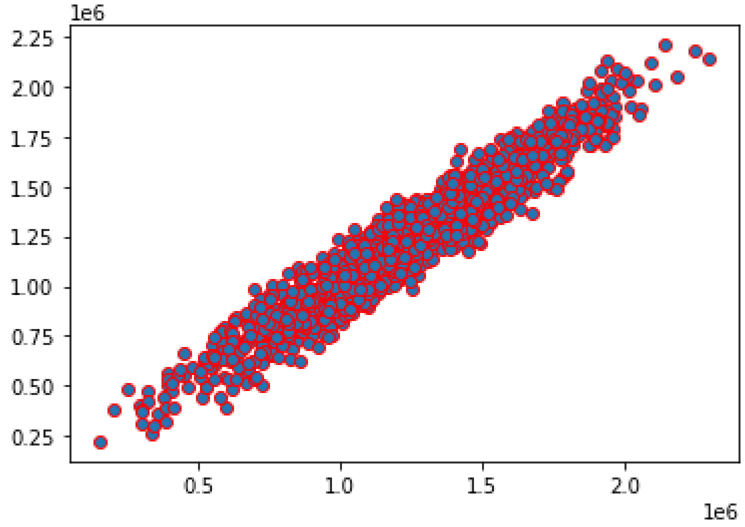
i) Create a scatterplot of the real test values versus the predicted values.

In [29]:

```
plt.scatter(y_test,prediction,edgecolor='red')
```

Out[29]:

```
<matplotlib.collections.PathCollection at 0x1b32fde31f0>
```



j) Calculate the Mean Absolute Error, Mean Squared Error, and the Root Mean Squared Error to evaluate ourmodel performance after importing metrics from sklearn.

In [30]:

```
from sklearn import metrics
print('MAE:', metrics.mean_absolute_error(y_test, prediction))
print('MSE:', metrics.mean_squared_error(y_test, prediction))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, prediction)))
```

```
MAE: 81491.64716425468
MSE: 10003687394.799738
RMSE: 100018.43527470193
```

k) Plot a histogram of the residuals. [Use either seaborn distplot, or just plt.hist()].

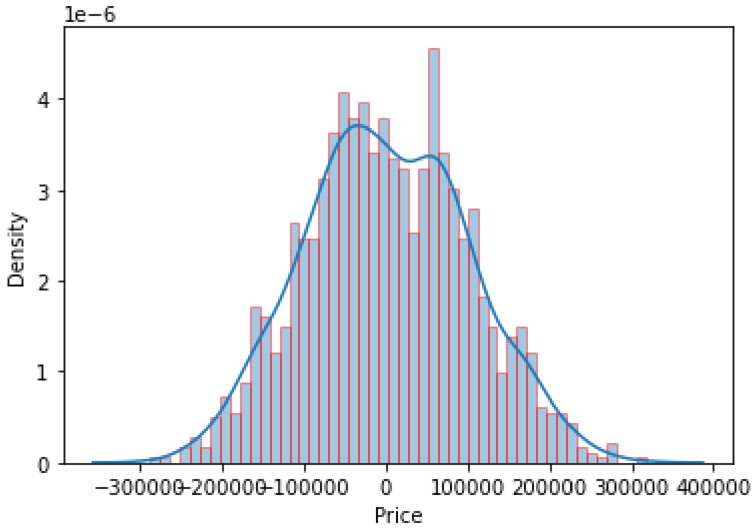
In [31]:

```
sb.distplot((y_test - prediction), bins = 50, hist_kws=dict(edgecolor="red", linewidth=1))
```

```
C:\Users\hp\anaconda3\lib\site-packages\seaborn\distributions.py:2551: Futur
eWarning: `distplot` is a deprecated function and will be removed in a futur
e version. Please adapt your code to use either `displot` (a figure-level fu
nction with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  warnings.warn(msg, FutureWarning)
```

Out[31]:

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



In [ ]: