In [ ]:

```
#Multi-variate Linear Regression
```

In [4]:

```
#Import the usual libraries for pandas and plotting and DATASET

import pandas as pd
import seaborn as sns
import numpy as np
#import sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
df1 = pd.read_csv('C:/Users/Deep/Desktop/USA_Housing.csv')
```

In [5]:

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

df1.head()
df1.info()
df1.describe()
```

```
<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
```

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%	61480.562388	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

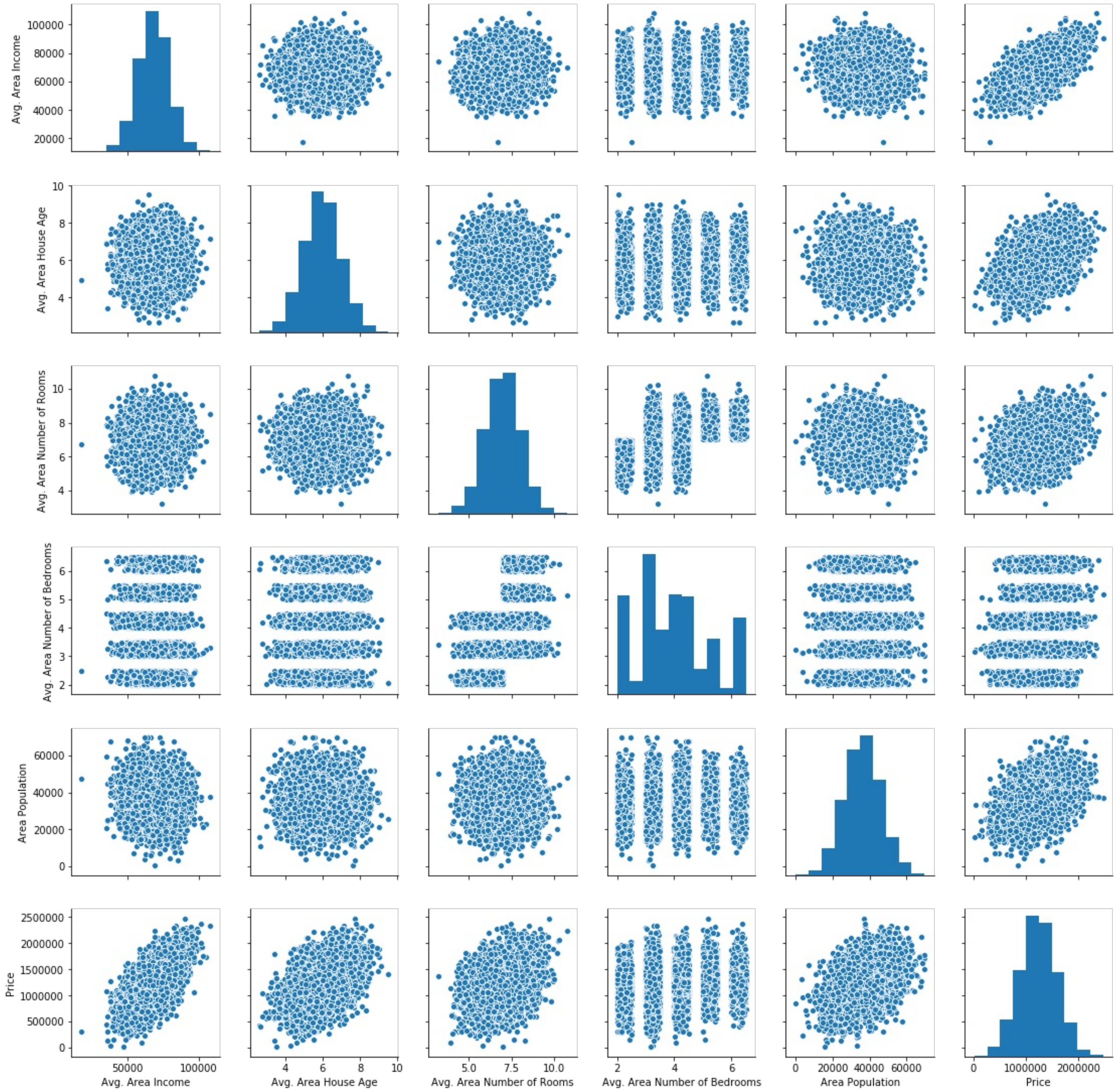
In [6]:

```
#Explore the types of relationships across the entire dataset using ‘pairplot’ method of seaborn and comment on t
hat.

sns.pairplot(df1)
```

Out[6]:

```
<seaborn.axisgrid.PairGrid at 0x23dfb6aac48>
```



In [8]:

```
#Set a variable X equal to the numerical features of the given dataset and a variable Y equal to the “Price” colu
mn

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

In [9]:

```
#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.

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

In [10]:

```
#Train the Linear Regression model based on the Training data after importing LinearRegression from sklearn.linea
r_model

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

Out[10]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [11]:

```
#Print the coefficients of the trained model. How can you interpret these coefficients?

print(lm1.score(x_test,y_test))
print(lm1.intercept_)
coeff_df = pd.DataFrame(lm1.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

```
0.9177363133484153
-2633616.177975607
```

Out[11]:

	Coefficient
Avg. Area Income	21.499022
Avg. Area House Age	164774.107750
Avg. Area Number of Rooms	121247.316541
Avg. Area Number of Bedrooms	1305.391428
Area Population	15.323144

In [12]:

```
#Interpretation of the Coefficients:
#Let Avg. Area Income’s Coefficient value is 21.709074
#Then it means ----- Holding all other features fixed, a 1 unit increase in Avg. Are Income is associated with an
increase of $21.709074
#Same explanation for all other features.
```

In [13]:

```
#Predict the house price of the Test Set data and display them

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

```
[1128475.60681551 1309602.83727064 1508597.81890157 ... 1285382.68325259
 1208236.63271802 1607320.11612524]
```

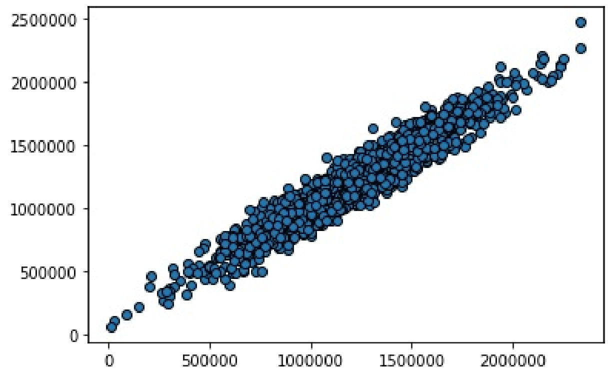
In [15]:

```
#Create a scatterplot of the real test values versus the predicted values

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

Out[15]:

```
<matplotlib.collections.PathCollection at 0x23dfcf95708>
```



In [18]:

```
#Calculate the Mean Absolute Error, Mean Squared Error, Root Mean Squared Error to evaluate our model performance
after importing metrics from sklearn.

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: 82726.79865953537
MSE: 10591541104.529352
RMSE: 102915.21318313124
```

In [24]:

```
#Plot a Histogram of the residuals. [Use either seaborn distplot, or just plt.hist()]

sns.distplot((y_test-prediction),bins=50,hist_kws=dict(edgecolor='black', linewidth=1))
```

Out[24]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x23dfe4f2c88>
```