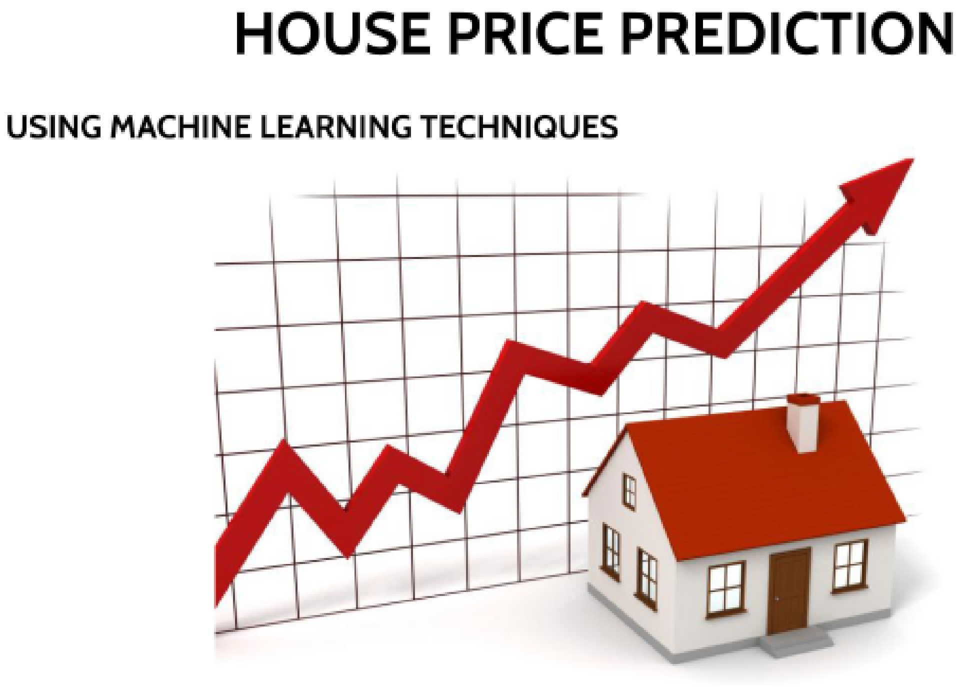


Importing All the necessary Libraries

In [1]:

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

import seaborn as sns
sns.set_style('darkgrid')

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import r2_score, mean_absolute_error,mean_squared_error

from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.ensemble import RandomForestRegressor
from sklearn.svm import SVR
import xgboost as xg

import warnings
warnings.filterwarnings("ignore")
```

Loading the Dataset

In [2]:
```
dataset = pd.read_csv('USA_Housing.csv')
dataset.head()
```

Out[2]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Address
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry Apt. 674\nLaurabury, NE 3701...
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Views Suite 079\nLake Kathleen, CA...
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizabeth Stravenue\nDanieltown, WI 06482...
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPO AP 44820
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nFPO AE 09386

Data Exploration

In [3]:
```
# Shape:
dataset.shape
```

Out[3]:
```
(5000, 7)
```

In [4]:
```
# Columns:
dataset.columns
```

Out[4]:
```
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 [5]:
```
dataset.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 [6]:
```
dataset.describe()
```

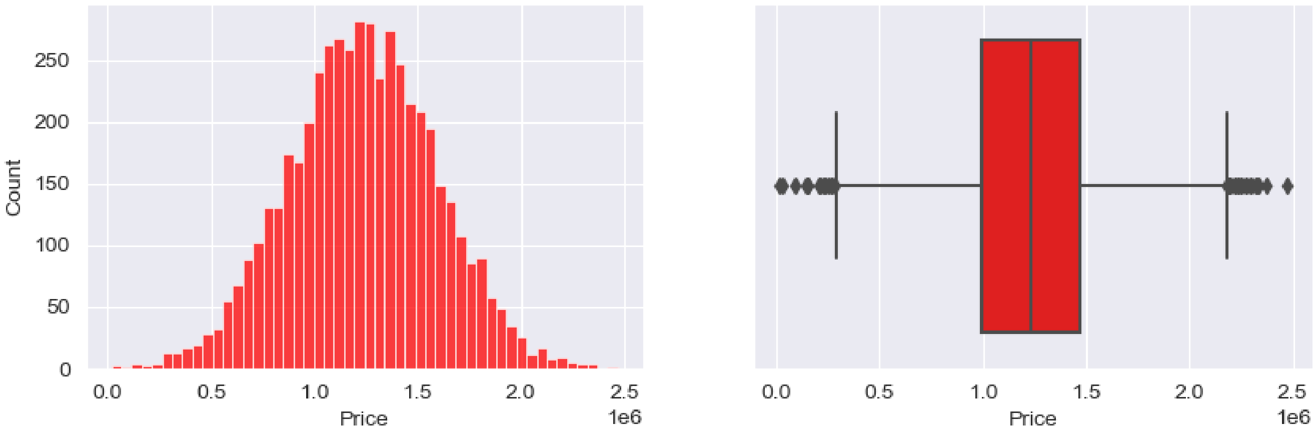
Out[6]:

	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

EDA and Pre-Processing of Data

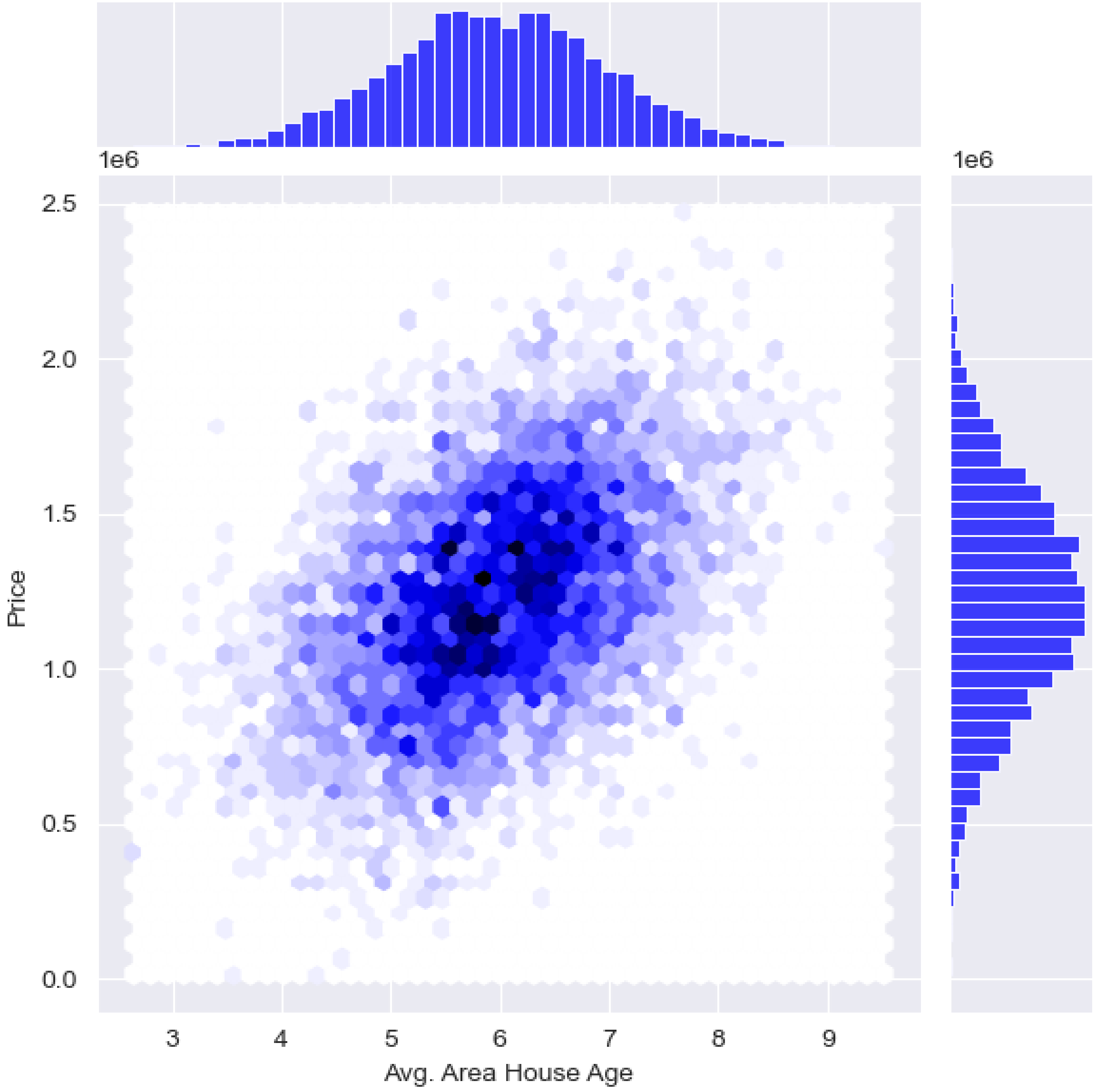
Distribution of Price column

In [7]:
```
plt.figure(figsize=(10,3))

plt.subplot(121)
sns.histplot(dataset, x='Price', bins=50, color='r')

plt.subplot(122)
sns.boxplot(dataset, x='Price',  color='r');
```



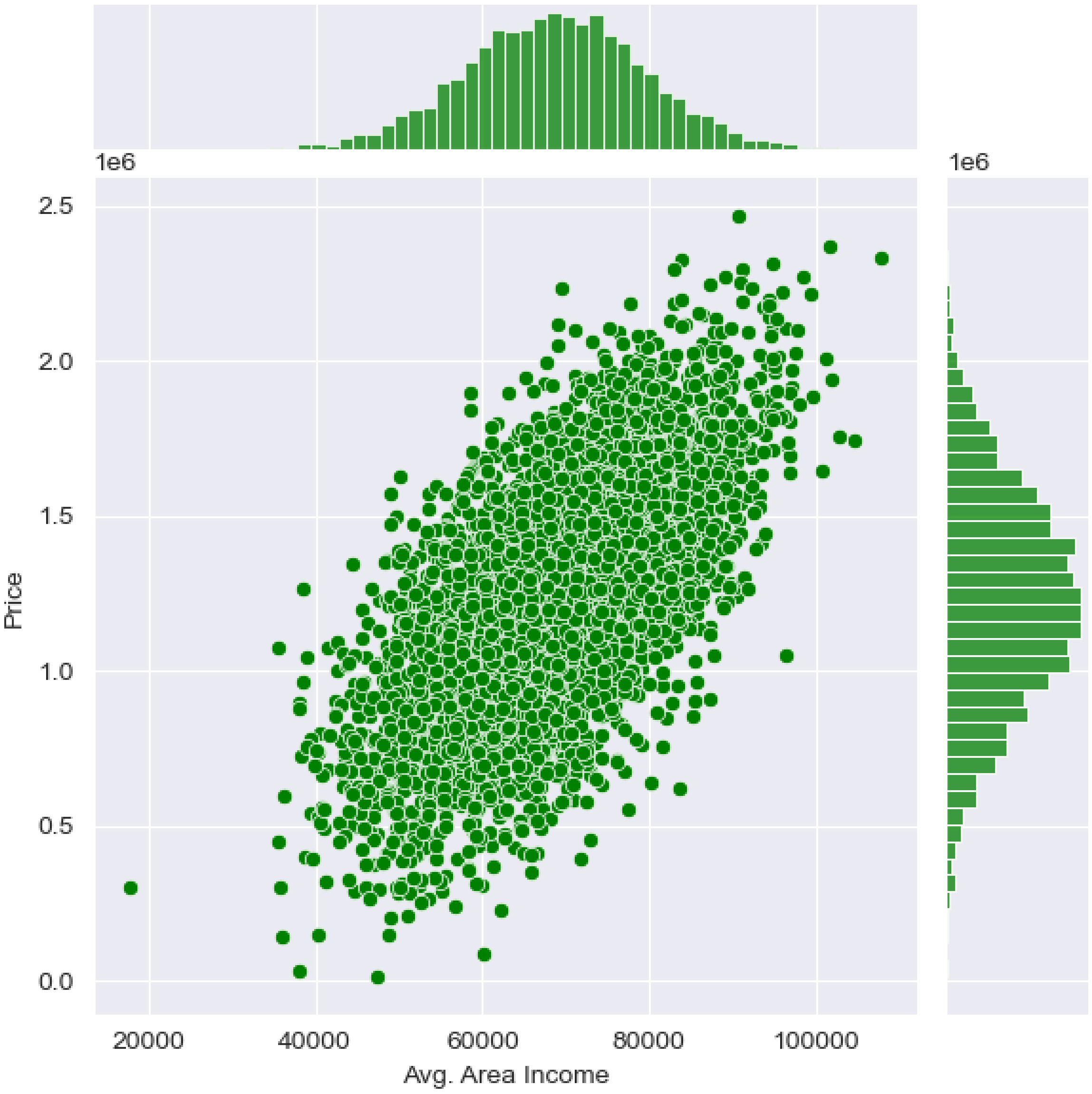
Avg. Area House Age Vs Price

In [8]:
```
sns.jointplot(dataset, x='Avg. Area House Age', y='Price', kind='hex', color='b');
```



Avg. Area Income Vs Price

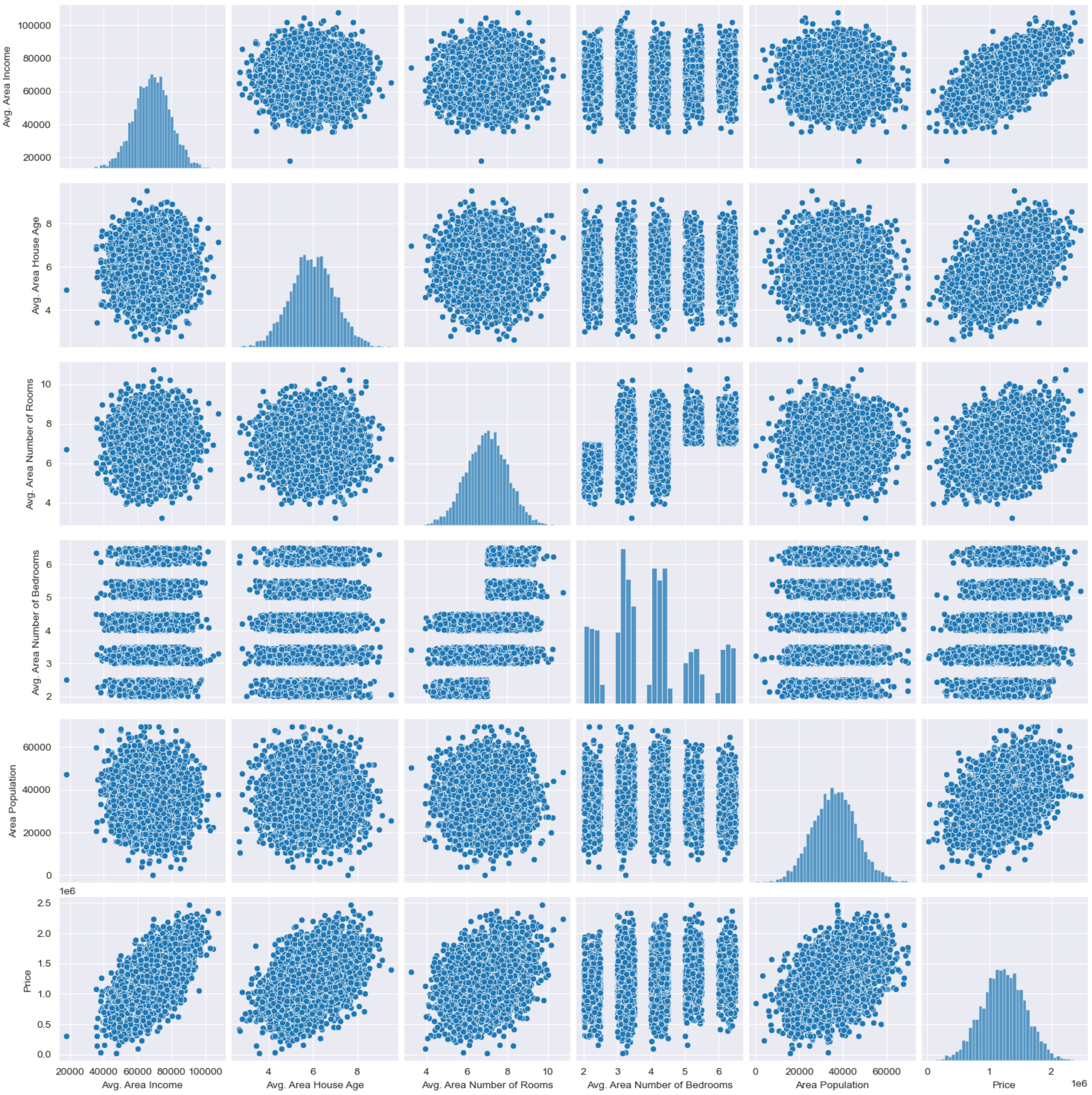
```
In [9]: sns.jointplot(dataset, x='Avg. Area Income', y='Price', color='g');
```



Correlation among all the columns

```
In [10]: plt.figure(figsize=(12,8))
         sns.pairplot(dataset);
```

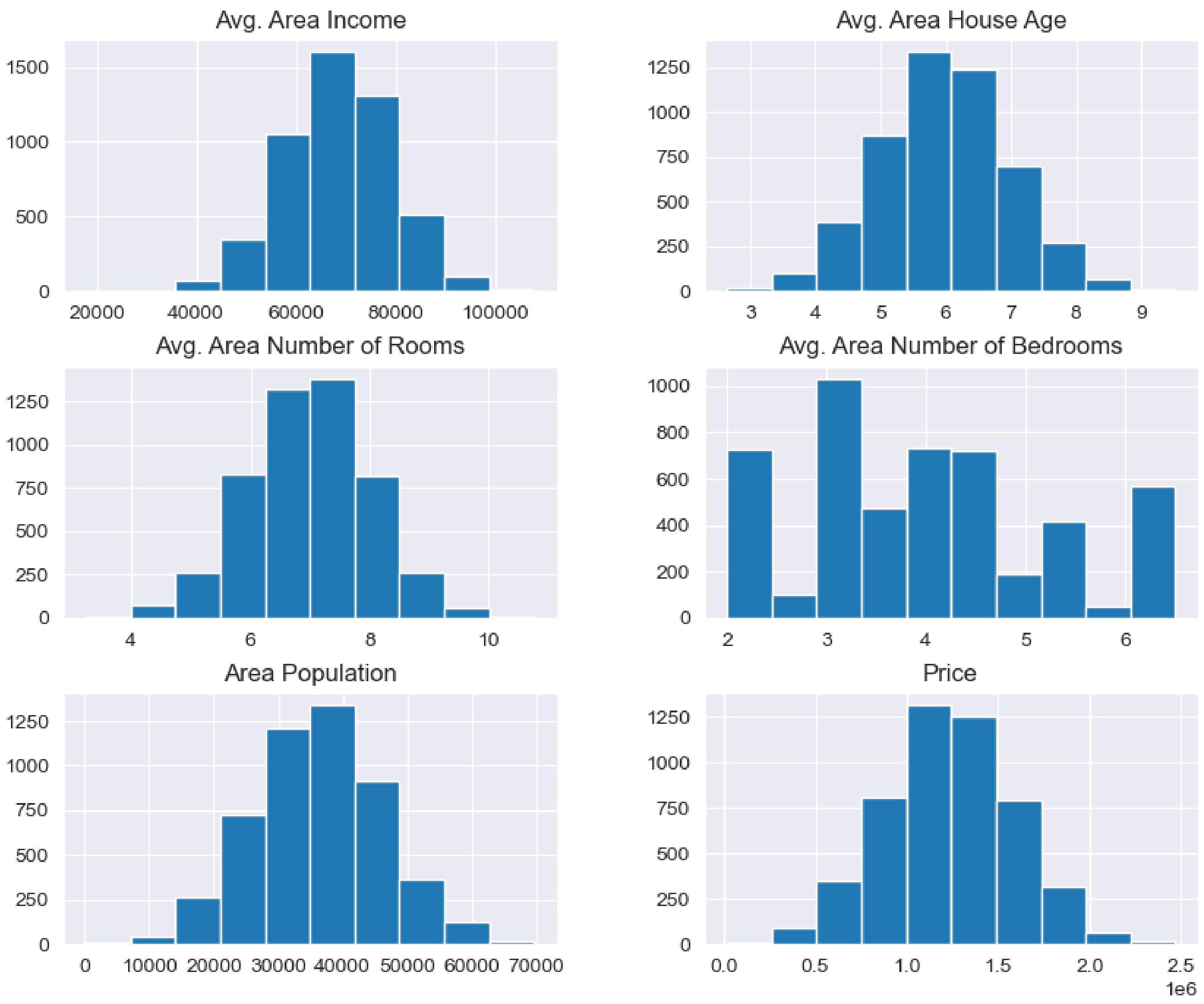
```
<Figure size 1200x800 with 0 Axes>
```



Distribution of all the columns

In [11]:

```
dataset.hist(figsize=(10,8));
```



Visualising Correlation

In [12]:

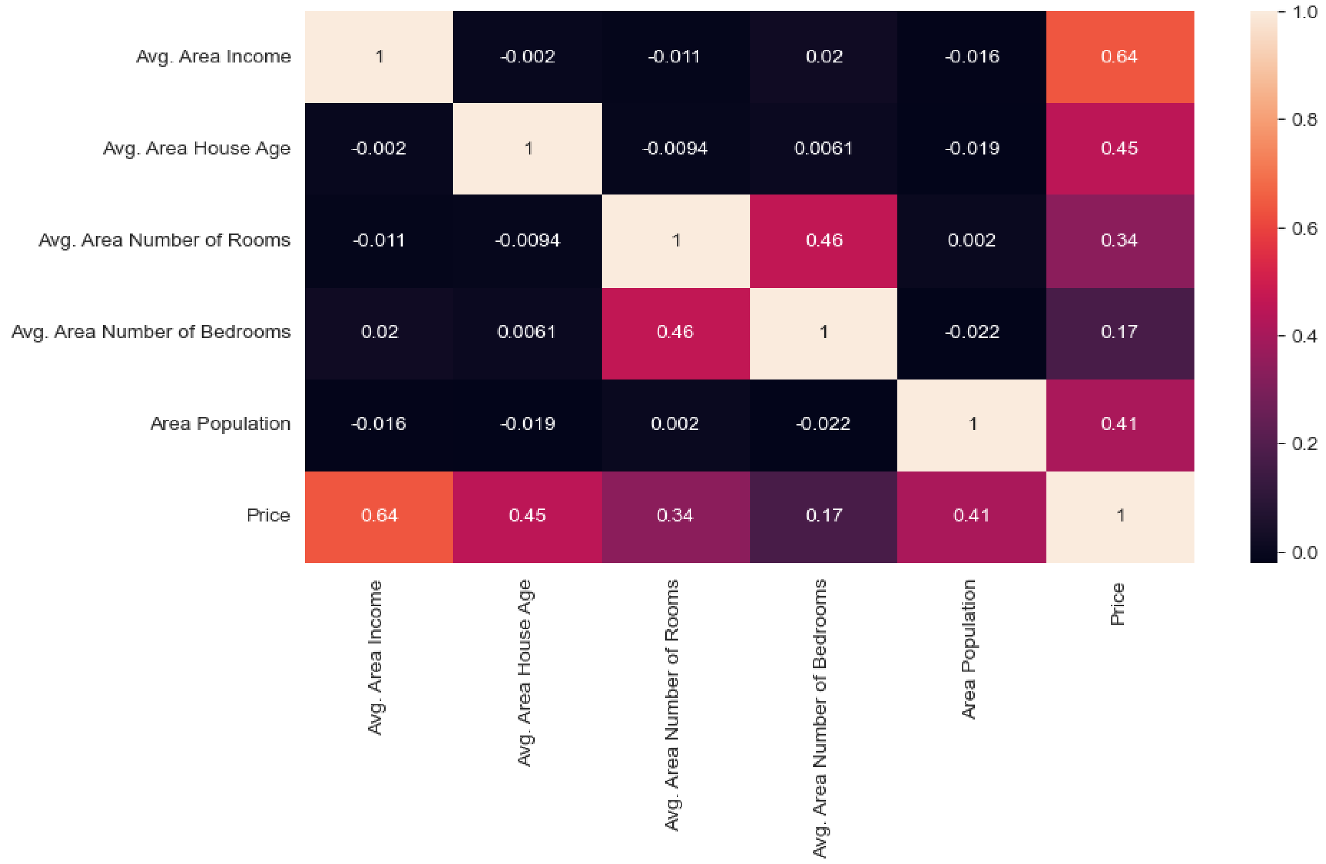
```
dataset.corr(numeric_only=True)
```

Out[12]:

	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

In [13]:

```
plt.figure(figsize=(10,5))
sns.heatmap(dataset.corr(numeric_only = True), annot=True);
```



Dividing Dataset in to features and target variable

In [14]:

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

Split the dataset into train and test

In [15]:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=10
```

In [16]:

```
Y_train.head()
```

Out[16]:

```
3413    1.305210e+06
1610    1.400961e+06
3459    1.048640e+06
4293    1.231157e+06
1039    1.391233e+06
Name: Price, dtype: float64
```

In [17]:

```
Y_train.shape
```

Out[17]:

```
(4000,)
```

In [18]:

```
Y_test.head()
```

Out[18]:

```
1718    1.251689e+06
2511    8.730483e+05
345     1.696978e+06
2521    1.063964e+06
54      9.487883e+05
Name: Price, dtype: float64
```

In [19]:

```
Y_test.shape
```

Out[19]:

```
(1000,)
```

Standardizing the data

```
In [20]: sc = StandardScaler()
         X_train_scal = sc.fit_transform(X_train)
         X_test_scal = sc.fit_transform(X_test)
```

```
In [21]: X_train_scal
```

```
Out[21]: array([[ 0.05569623,  0.65886183, -0.86300913,  0.29911519,  0.06391981],
                [-0.05545523, -0.58559522,  2.37598858,  1.2000951 ,  0.69883088],
                [-1.11165023, -0.48032202,  0.13621855,  1.73581289,  1.14379364],
                ...,
                [-1.20704442, -2.26895761, -0.11765963, -1.34862286,  2.71900465],
                [-0.50898477, -0.03604344, -1.07361484,  0.11242565, -0.34813857],
                [ 0.62279188,  1.69958661,  1.456617  ,  0.29911519,  2.01048875]])
```

```
In [22]: X_test_scal
```

```
Out[22]: array([[-0.21555096, -0.281372  ,  0.77408385,  0.20931536,  0.07088885],
                [-0.64310056, -1.06017806,  0.41824478,  1.74293863, -1.01812409],
                [ 0.42413203,  0.75165241,  1.2987239 , -0.58171138,  0.71366541],
                ...,
                [ 0.25042527, -0.08048033,  0.40212954, -0.04897908,  0.15625295],
                [ 1.69352407, -0.96166121,  0.18304551,  1.07298741, -2.88549085],
                [ 0.11007237, -0.14440016, -0.20680012, -1.2920211 , -0.80666516]])
```

Till now we have completed all the Data Pre-Processing steps. Now the data is ready for model building

```
In [ ]:
```