Importing Libraries

In [3]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading CSV file

In [4]:
```
df=pd.read_csv("USA_Housing.csv")
df.head()
```

Out[4]:

	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 Ferr 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

Information about data

In [5]:
```
df.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
```

overview of distribution and central tendencies of data

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 [7]:
```
df.isnull().sum()
```

Out[7]:
```
Avg. Area Income              0
Avg. Area House Age           0
Avg. Area Number of Rooms     0
Avg. Area Number of Bedrooms  0
Area Population               0
Price                         0
Address                       0
dtype: int64
```

In [8]:
```
df.columns
```

Out[8]:
```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Roo
ms',
       'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Addres
s'],
      dtype='object')
```

correlation between columns

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

Out[9]:

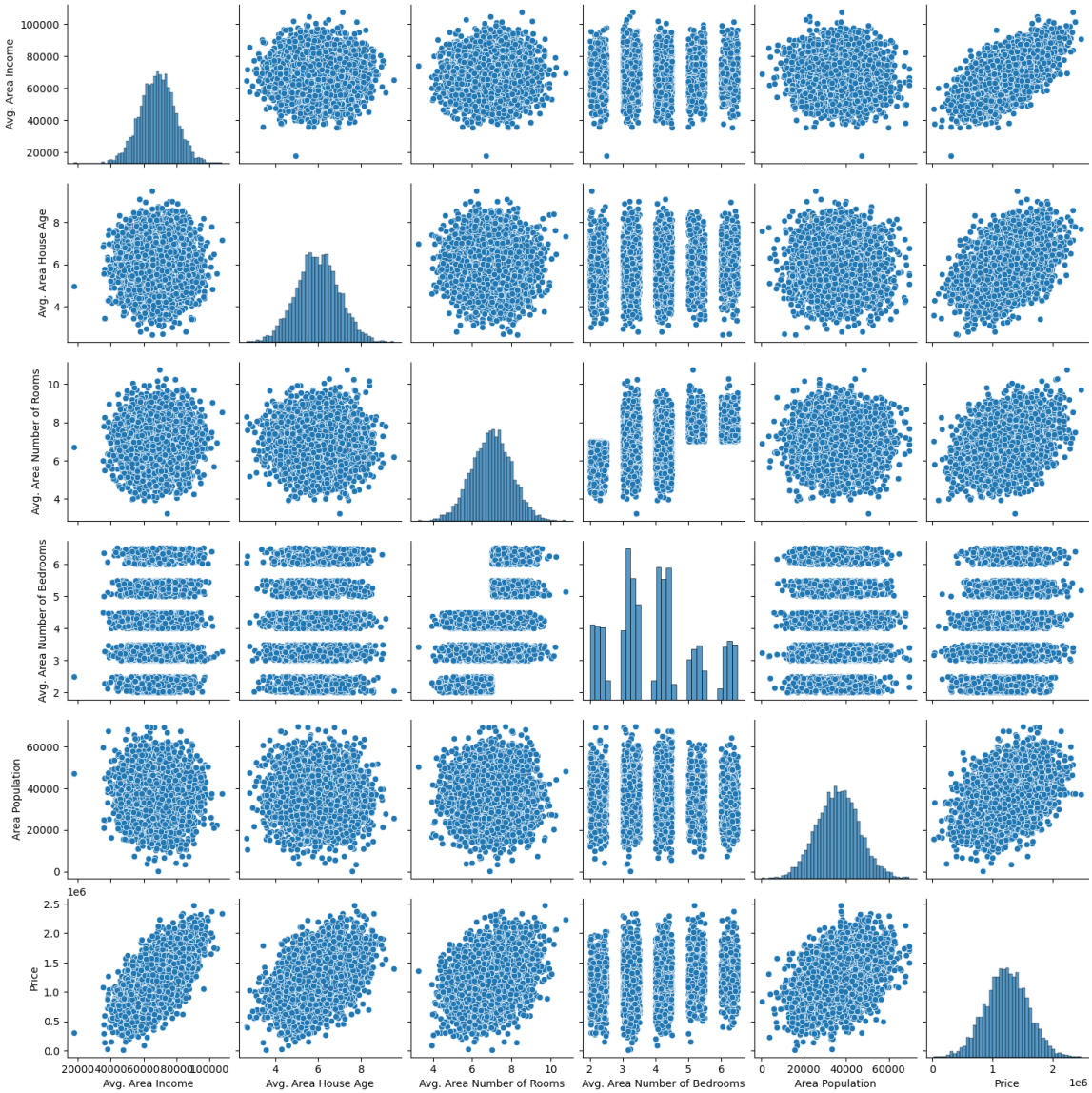
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
Avg. Area Income	1.000000	-0.002007	-0.011032	0.019788	-0.016234	0.639734
Avg. Area House Age	-0.002007	1.000000	-0.009428	0.006149	-0.018743	0.452543
Avg. Area Number of Rooms	-0.011032	-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

In [10]:
```
import warnings
warnings.simplefilter(action='ignore',category=FutureWarning)
```

Data Visualization

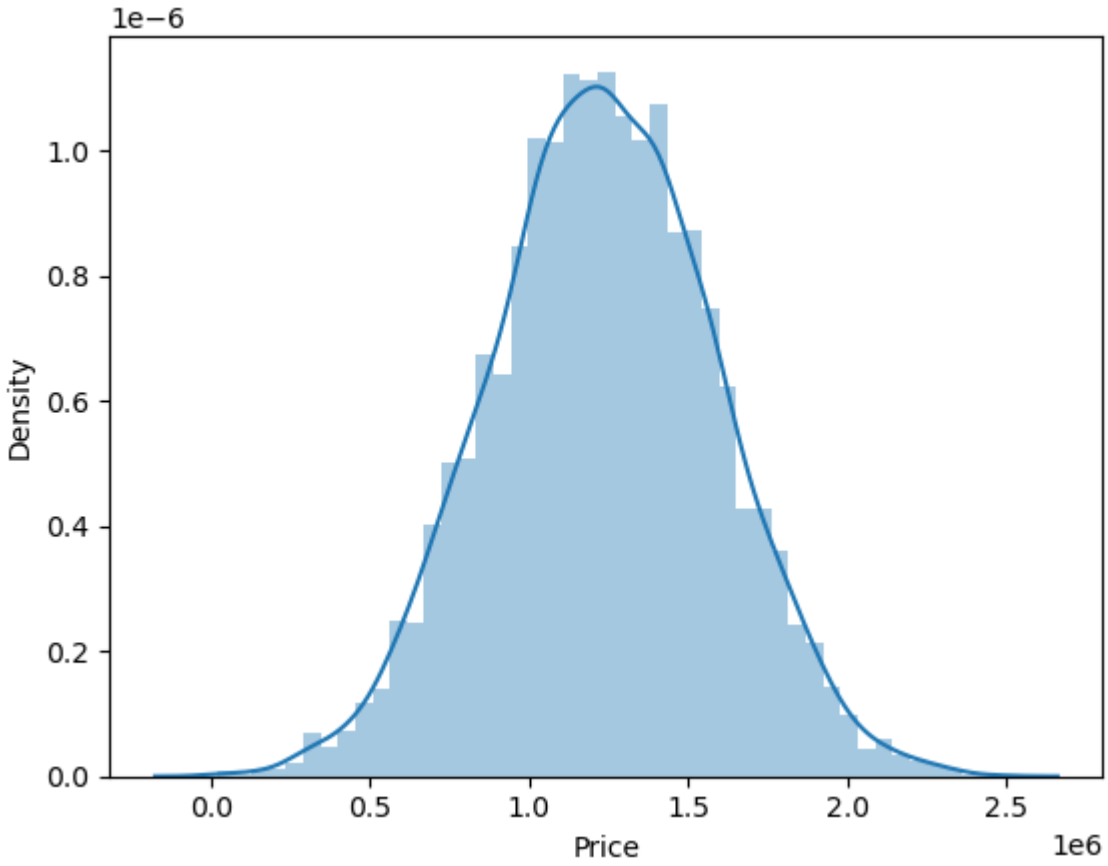
In [29]:
```
import seaborn as sns
sns.pairplot(df)
```

Out[29]:
```
<seaborn.axisgrid.PairGrid at 0x23b5eded8b0>
```



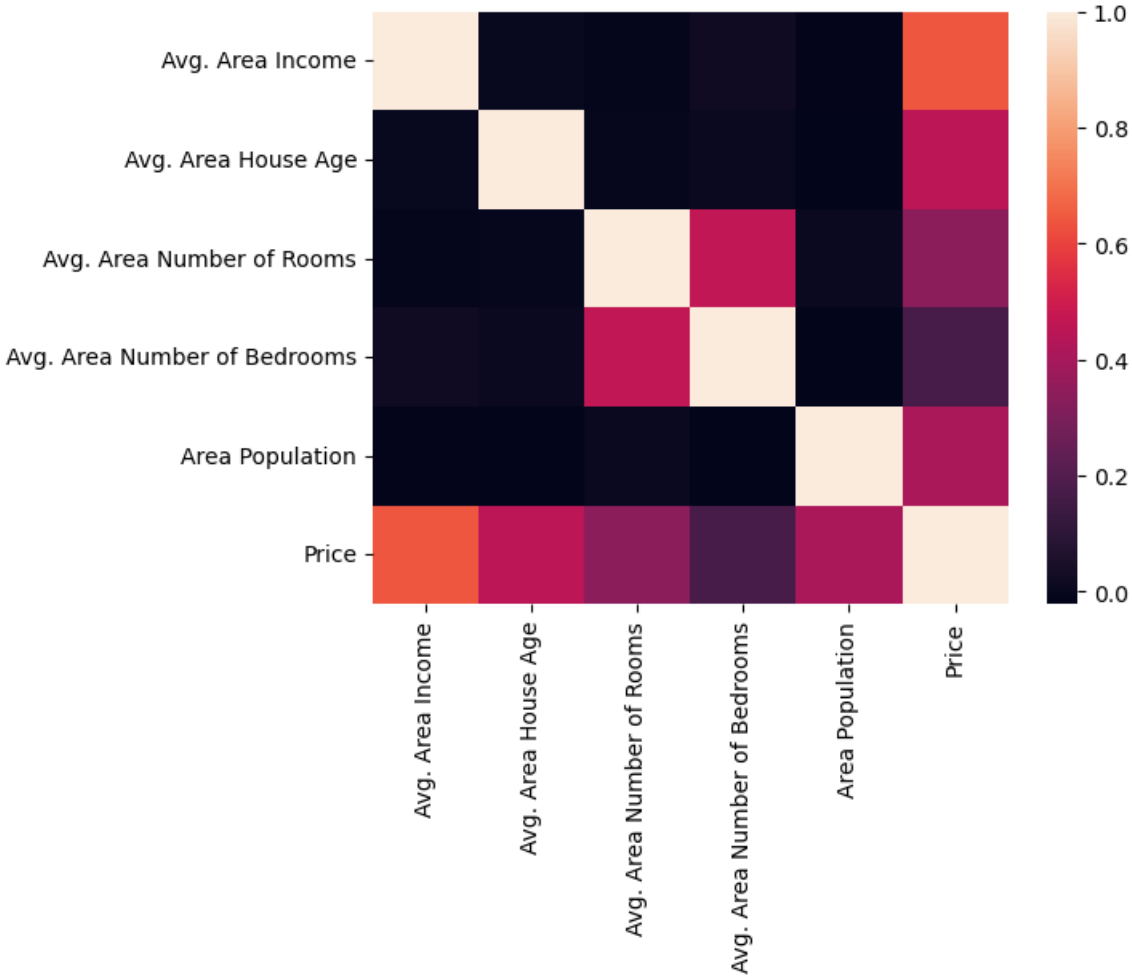
In [11]:
```
sns.distplot(df['Price'])
```

Out[11]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [12]:
```
sns.heatmap(df.corr())
```

Out[12]: <AxesSubplot:>



Training a linear regression model

Split data into features(x) and target(y)

In [12]:
```
x=df.drop(['Price','Address'],axis=1)
y=df['Price']
```

Train Test Split

In [13]:
```
from sklearn.model_selection import train_test_split
```

In [14]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.4,random_stat
```

Training model using Linear Regression

In [15]:
```
from sklearn.linear_model import LinearRegression
model=LinearRegression()
model.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

Make predictions

In [16]:
```
y_pred=model.predict(x_test)
```

Evaluate Model

In [17]:
```
from sklearn import metrics
```

In [18]:
```
print("Mean Squared error : ",metrics.mean_squared_error(y_test,y_pred))
print("r2_score : ",metrics.r2_score(y_test,y_pred))
```

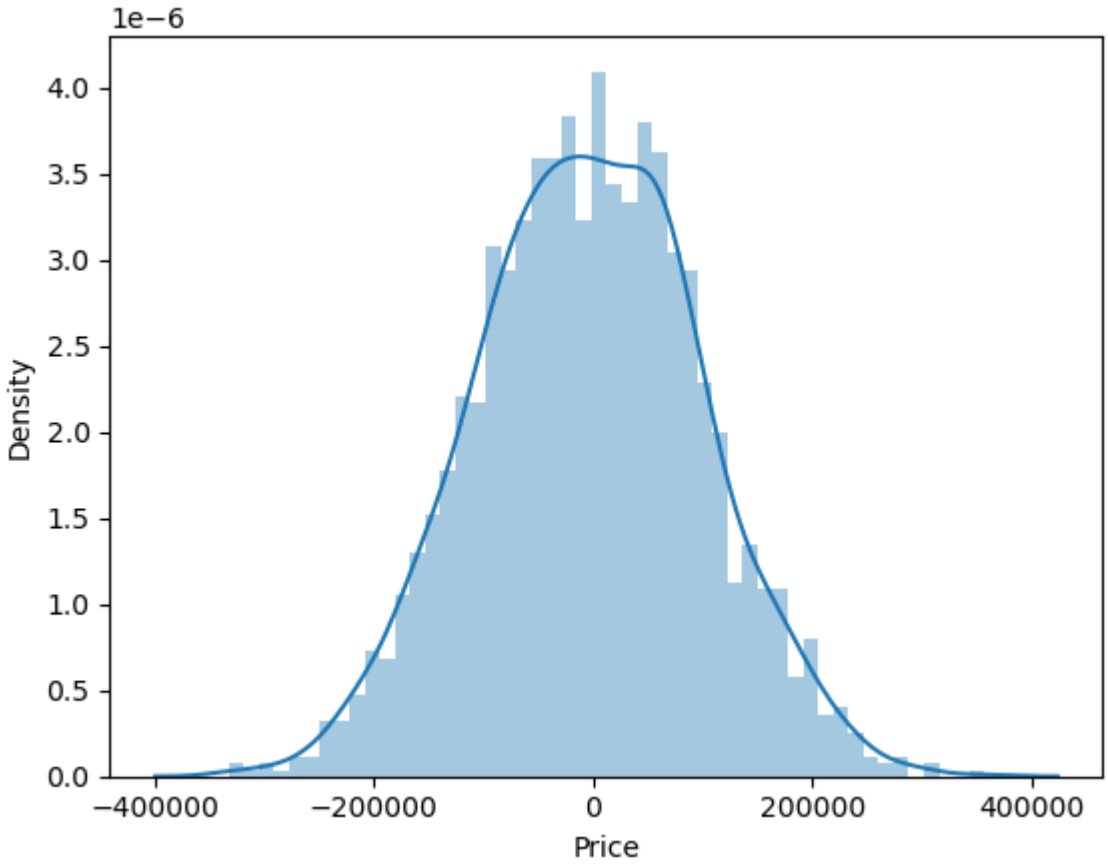
```
Mean Squared error :  10460958907.2095
r2_score :  0.91768240096492
```

An r2_score of 0.91768240096492 indicates approximately 92% of variance in price is explained by model predictions and it's predictions are reasonably close to actual values.

Visualizing the predictions from model

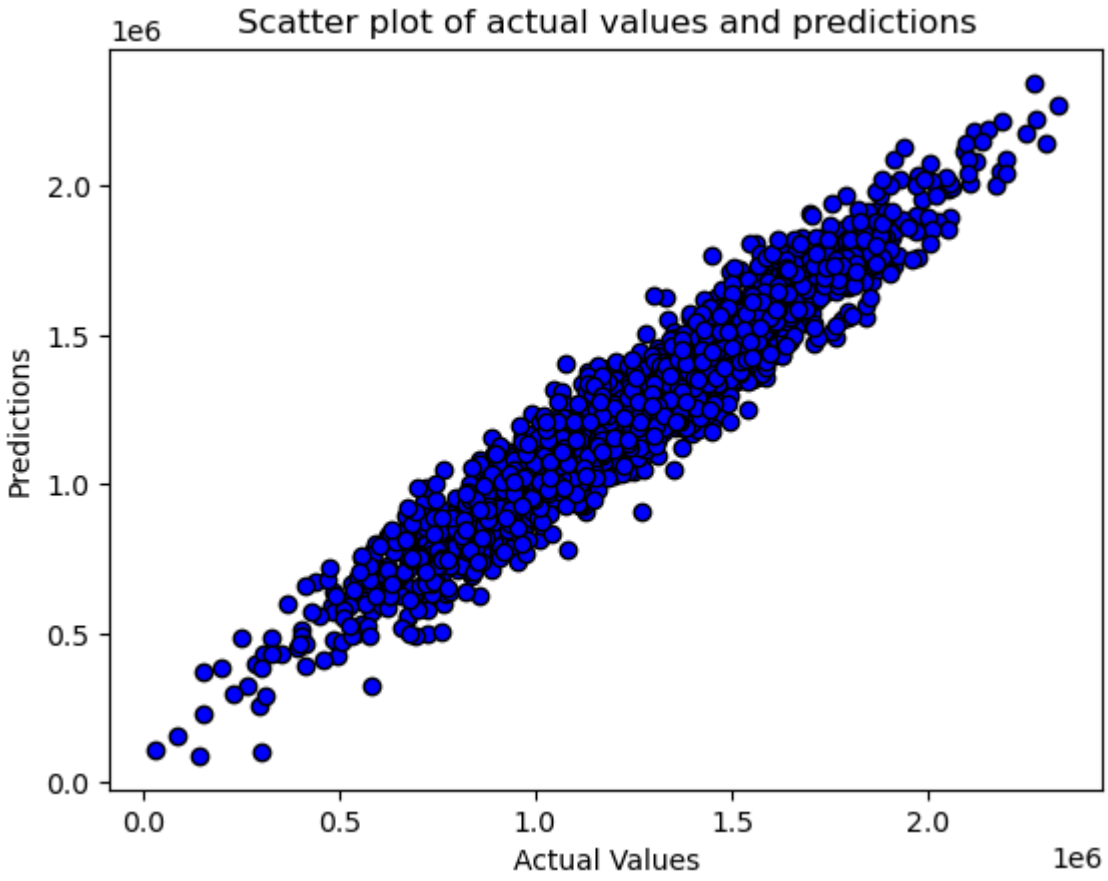
In [22]:
```
sns.distplot((y_test-y_pred),bins=50)
```

Out[22]:
```
<AxesSubplot:xlabel='Price', ylabel='Density'>
```



In [23]:
```
plt.scatter(y_test,y_pred,marker='o',edgecolors='black',c='blue')
plt.xlabel('Actual Values')
plt.ylabel('Predictions')
plt.title("Scatter plot of actual values and predictions")
```

Out[23]:
```
Text(0.5, 1.0, 'Scatter plot of actual values and predictions')
```



Prediction

Type *Markdown* and LaTeX: α^2

In [29]:
```
i=float(input("Enter Average area income : "))
a=float(input("Enter Average area house age : "))
r=float(input("Enter average area number of rooms : "))
b=float(input("Enter average area number of bedrooms : "))
p=float(input("Enter area population : "))
print("Predicted house price is ",model.predict([[i,a,r,b,p]])[0])
```

```
Enter Average area income : 70000
Enter Average area house age : 6
Enter average area number of rooms : 5
Enter average area number of bedrooms : 2
Enter area population : 20000
Predicted house price is  775438.5758781768

C:\Users\DELL\anaconda3\lib\site-packages\sklearn\base.py:450: UserWarnin
g: X does not have valid feature names, but LinearRegression was fitted wi
th feature names
  warnings.warn(
```