LINEAR REGRESSION MACHINE LEARNING MODEL

This programme is to visualise the USA_HOUSE_PRICING data set and run a linear regression model for predicting the house prices. Before we start Data visualisation is important.

In [2]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

%matplotlib inline
```

In [3]:

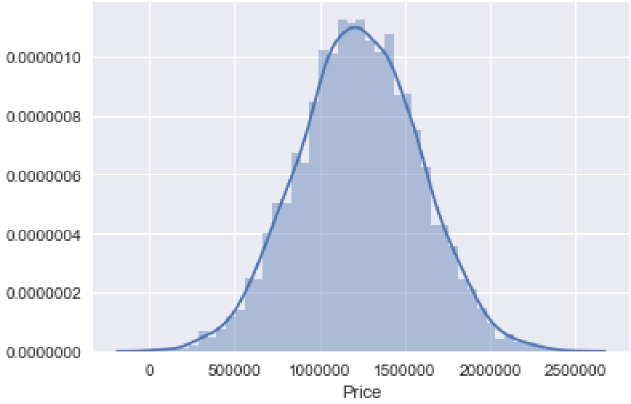
```
df = pd.read_csv('USA_Housing.csv')
```

In [4]:

```
sns.distplot(df['Price'])
```

Out[4]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10f6ee860>
```



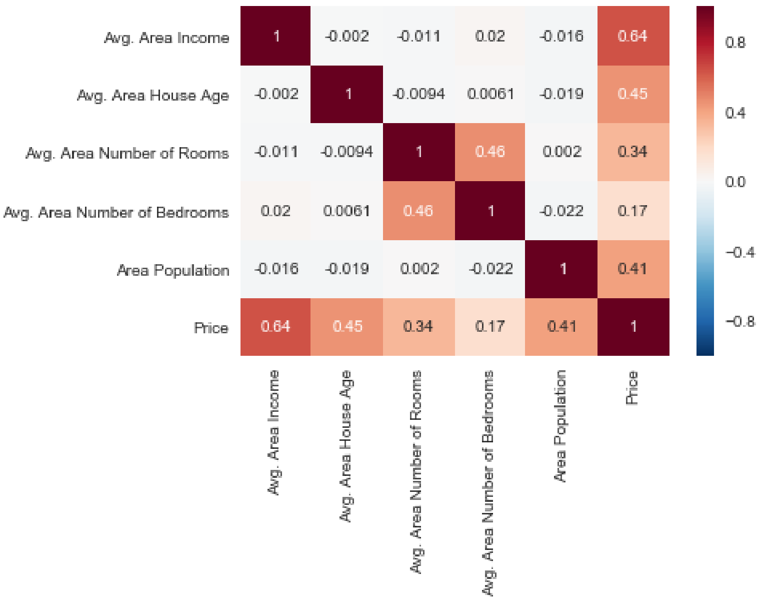
In [5]:

```
sns.heatmap(df.corr(),annot = True
           )

#this corr map is used to find the correlation between the attributes and the pricing so as to
#get an idea about the relationship between them.
```

Out[5]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x10f776080>
```



In [6]:

```
df.columns
```

Out[6]:

```
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 [7]:

```
x = df[['Avg. Area Income','Avg. Area House Age','Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms','Area Population']]
#separate the other attributes from the predicting attribute
```

In [8]:

```
y = df[['Price']]
#separte the predicting attribute into Y for model training
```

In [9]:

```
from sklearn.model_selection import train_test_split

#import model selection train test split for splitting the data into test and train for
#model validation.
```

In [10]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=101)
```

In [11]:

```
from sklearn.linear_model import LinearRegression
```

In [12]:

```
lm = LinearRegression()
#loading the model constructor
```

In [13]:

```
lm.fit(x_train,y_train)
#training or fitting the train data into the model
```

Out[13]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [14]:

```
print(lm.intercept_)
```

```
[-2640159.79685191]
```

In [15]:

```
lm.coef_
#examining the co-efficients of the fitted model.
```

Out[15]:

```
array([[  2.15282755e+01,   1.64883282e+05,   1.22368678e+05,
          2.23380186e+03,   1.51504200e+01]])
```

In [16]:

```
x_train.columns
```

Out[16]:

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population'],
      dtype='object')
```

In [17]:

```
cdf = pd.DataFrame(data=lm.coef_.reshape(5,1),index=x_train.columns,columns=['Coeff'])
```

In [18]:

```
cdf
```

Out[18]:

	Coeff
Avg. Area Income	21.528276
Avg. Area House Age	164883.282027
Avg. Area Number of Rooms	122368.678027
Avg. Area Number of Bedrooms	2233.801864
Area Population	15.150420

In [19]:

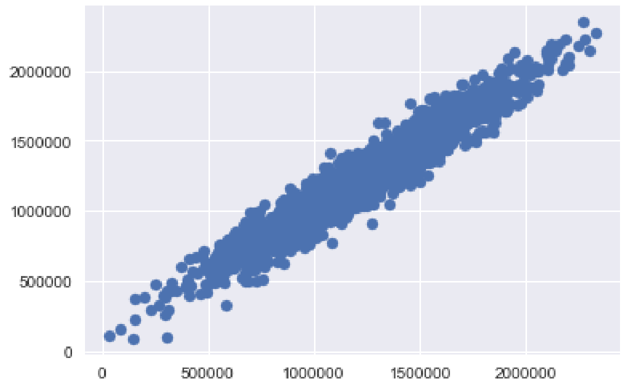
```
predictions = lm.predict(x_test)
```

In [20]:

```
plt.scatter(y_test,predictions)
#to visualise the predictions and the test Y !! almost it is forming a linear line with less deviation
```

Out[20]:

```
<matplotlib.collections.PathCollection at 0x113d992b0>
```

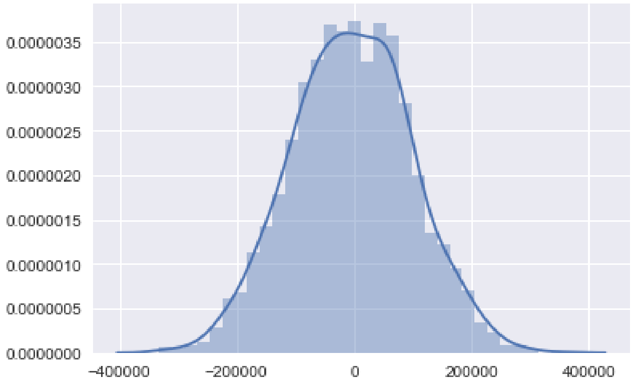


In [21]:

```
sns.distplot((y_test-predictions))
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x113d49d68>
```



In [22]:

```
from sklearn import metrics
```

In [ ]:

```
np.sqrt(metrics.sqrt)
#printing the error values of the model
```

In [24]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 82288.2225191
MSE: 10460958907.2
RMSE: 102278.829223
```

In [ ]: