IMPORT LIBRARIES

In [5]:

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

Check out the data

In [6]:

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

In [9]:

```
USAhousing.head()
```

Out[9]:

	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price	Addr
0	79545.458574	5.682861	7.009188	4.09	23086.800503	1.059034e+06	208 Michael Ferry 674\nLaurabury, 37(
1	79248.642455	6.002900	6.730821	3.09	40173.072174	1.505891e+06	188 Johnson Vi Suite 079\nL Kathleen, C
2	61287.067179	5.865890	8.512727	5.13	36882.159400	1.058988e+06	9127 Elizal Stravenue\nDanieltc WI 064
3	63345.240046	7.188236	5.586729	3.26	34310.242831	1.260617e+06	USS Barnett\nFPC 44
4	59982.197226	5.040555	7.839388	4.23	26354.109472	6.309435e+05	USNS Raymond\nF AE 09

In [10]:

```
USAhousing.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 7 columns):
Avg. Area Income                5000 non-null float64
Avg. Area House Age             5000 non-null float64
Avg. Area Number of Rooms       5000 non-null float64
Avg. Area Number of Bedrooms    5000 non-null float64
Area Population                 5000 non-null float64
Price                           5000 non-null float64
Address                         5000 non-null object
dtypes: float64(6), object(1)
memory usage: 273.5+ KB
```

In [11]:

```
USAhousing.columns
```

Out[11]:

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

In [12]:

```
USAhousing.describe()
```

Out[12]:

	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

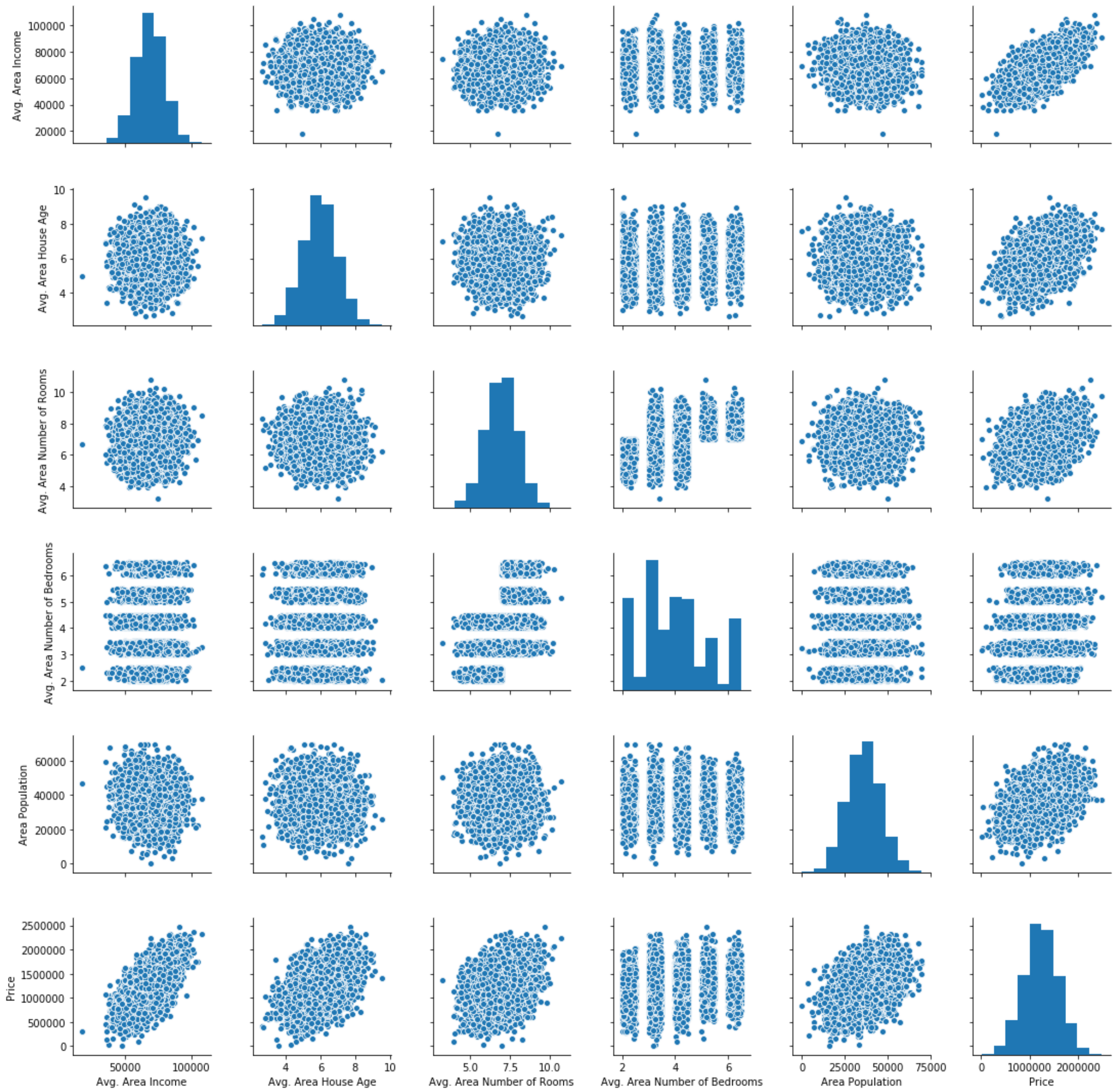
let's create some simple plots to check out the data

In [13]:

```
sns.pairplot(USAhousing)
```

Out[13]:

```
<seaborn.axisgrid.PairGrid at 0x15318a880f0>
```

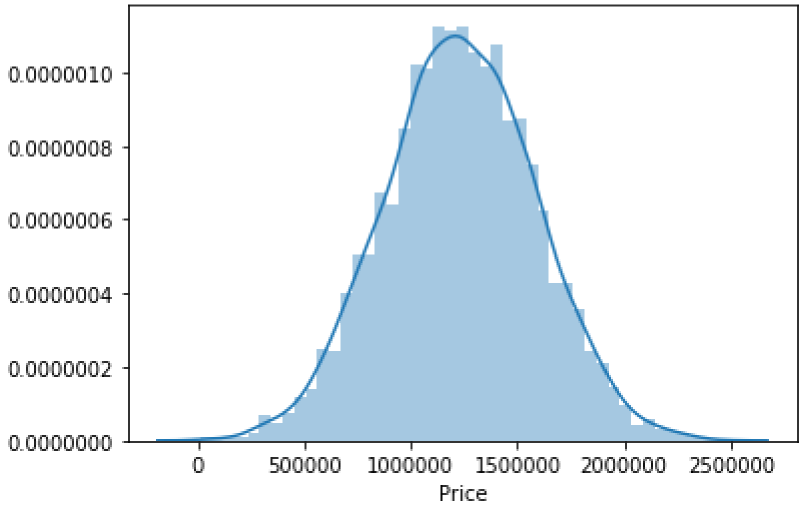


In [15]:

```
sns.distplot(USAhousing['Price']) # it is better you check the dist for the column you are
```

Out[15]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x15319a46898>
```



In [16]:

```
sns.heatmap(USAhousing.corr()) #corr mean correction
```

Out[16]:

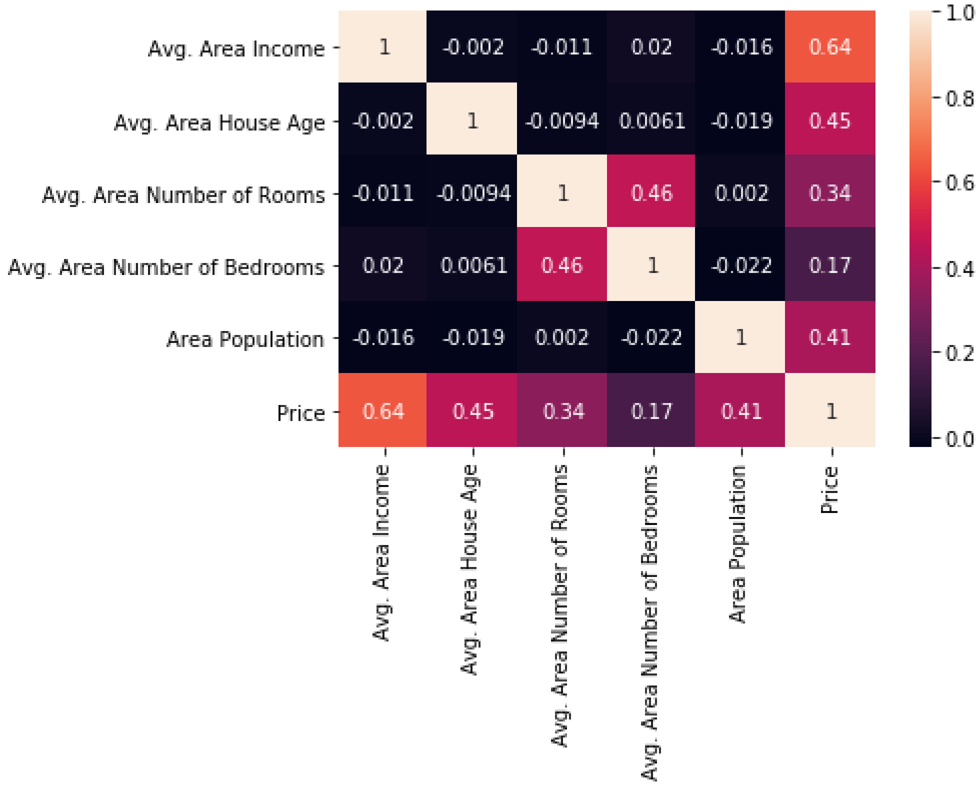
```
<matplotlib.axes._subplots.AxesSubplot at 0x1531a552128>
```

In [17]:

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

Out[17]:

<matplotlib.axes._subplots.AxesSubplot at 0x1531a51cf98>



In [18]:

```
sns.heatmap(USAhousing.corr(), annot = False)
```

Out[18]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1531a6dbfd0>
```

TRAINNING A LINEAR REGRESSION MODEL

LET'S now begin to train out regression model we will need to first splitup our data into an X array that contain the features to train on. and a Y array with target variable in this case the price columns we will loss out of the Address column because it only has text into that the linear regression model cant use

X and Y arrays

In [19]:

```
x = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',
       'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price'] # Target variable what we are trying to prediction
```

Train Test Split

Now let split the data into a training set and a testing set.we will train out model on the training out and then use the test set to evaluate the model

In [20]:

```
from sklearn.model_selection import train_test_split
```

In [21]:

```
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.4, random_state=101)
```

Creating and Training the model

In [24]:

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

In [29]:

```
lm = LinearRegression()
```

In [30]:

```
lm.fit(X_train,y_train)
```

Out[30]:

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

MODEL EVALUATION

Let's evaluate the model by checking out it's coefficients and how we can interpret them:

In [32]:

```
# print the intercept
print(lm.intercept_)
```

```
-2640159.796852678
```

In [34]:

```
lm.coef_
```

Out[34]:

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

In [36]:

```
X_train.columns
```

Out[36]:

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

In [38]:

```
coeff_df = pd.DataFrame(lm.coef_, x.columns,columns=['Coeffiecient'])
coeff_df
```

Out[38]:

	Coeffiecient
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 [ ]:

```
Interpreting the coefficients:
    1. Holding all other features fixed, a 1 unit increase in Avg.Area income is associated
    2. Holding all other features fixed, a 1 unit increase in Avg.Area House Age is associa
    3. Holding all other features fixed, a 1 unit increase in Avg.Area Number of rooms is a
    4. Holding all other features fixed, a 1 unit increase in Avg.Area Population is associ
```

PREDICTIONS FROM THE MODEL

LET US GRAB PREDICTIONS OFF OUR TEST SET AND SEE HOW WELL IT DID

#This take just 1 feature X, THE X_test is a feature the model has not seen before

In [39]:

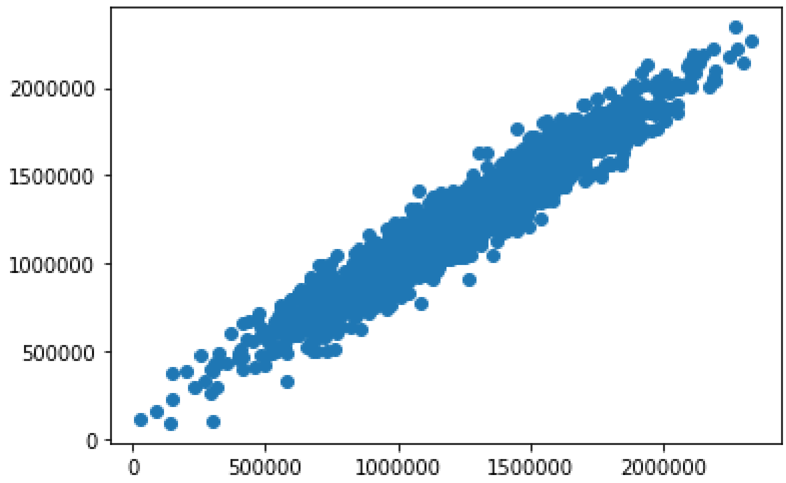
```
predictions = lm.predict(X_test)
```

In [40]:

```
plt.scatter(y_test,predictions)
```

Out[40]:

```
<matplotlib.collections.PathCollection at 0x1531bf5df28>
```



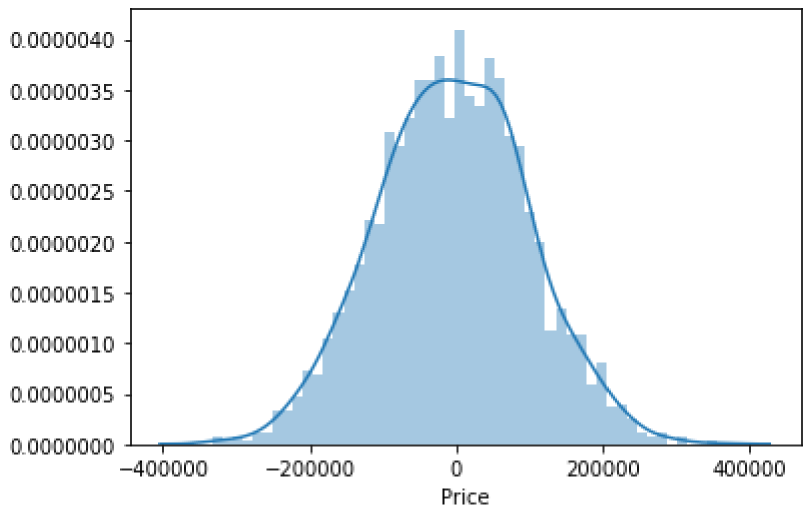
RESIDUAL Histogram Residual are the differnece between the actual valuey_test and predicted values

In [46]:

```
sns.distplot((y_test- predictions), bins=50)
```

Out[46]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x1531bf84cc0>
```



In [ ]:

```

```