In [1]:
```
import pandas as pd
```

In [2]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [3]:
```
USAhousing = pd.read_csv('USA_Housing.csv')
```

In [4]:
```
USAhousing.head()
```

Out[4]:

	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

In [5]:
```
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 [6]:
```
USAhousing.describe()
```

Out[6]:

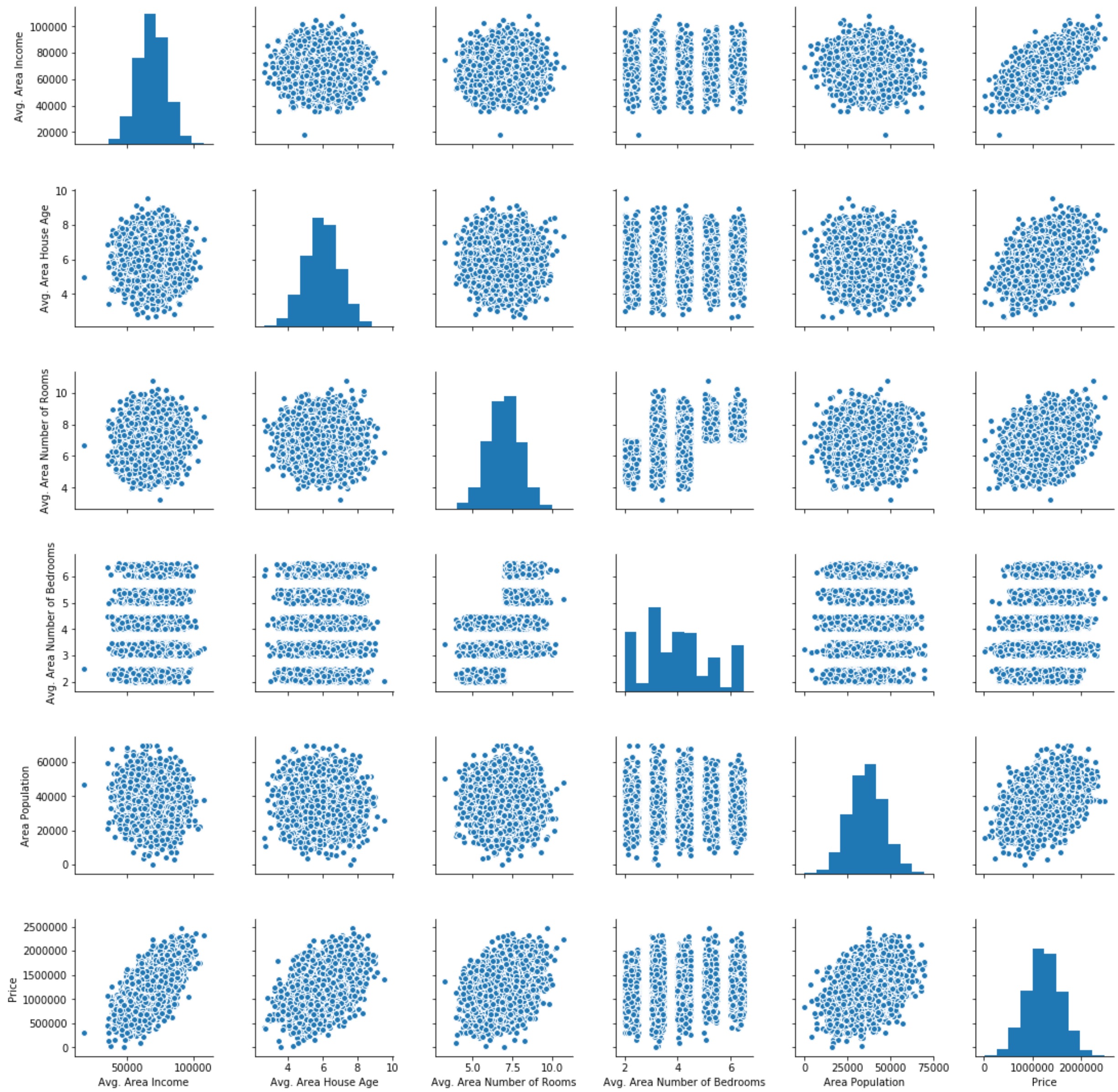
	Avg. Area Income	Avg. Area House Age	Avg. Area Number of Rooms	Avg. Area Number of Bedrooms	Area Population	Price
count	5000.000000	5000.000000	5000.000000	5000.000000	5000.000000	5.000000e+03
mean	68583.108984	5.977222	6.987792	3.981330	36163.516039	1.232073e+06
std	10657.991214	0.991456	1.005833	1.234137	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 [8]:
```
USAhousing.columns
```

Out[8]:
```
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 [9]:
```
sns.pairplot(USAhousing)
```

Out[9]:
```
<seaborn.axisgrid.PairGrid at 0x85d1a58>
```



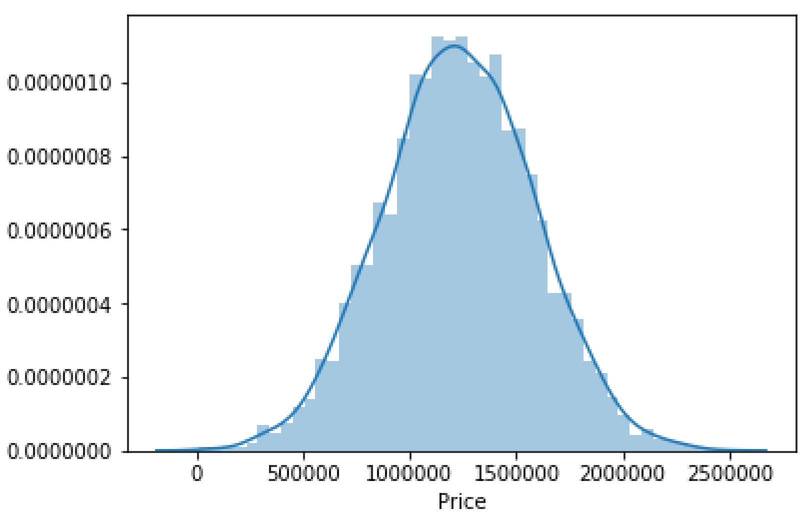
In [10]:
```
sns.heatmap(USAhousing.corr())
```

Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0xc3bab38>

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

```
C:\Users\q21\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWar
ning: The 'normed' kwarg is deprecated, and has been replaced by the 'density'
kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x526d358>



In [12]:
```
X = USAhousing[['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of R
               'Avg. Area Number of Bedrooms', 'Area Population']]
y = USAhousing['Price']
```

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_s
```

In [15]:
```
from sklearn.linear_model import LinearRegression
```

In [16]:
```
lm = LinearRegression()
```

In [17]:
```
lm.fit(X_train,y_train)
```

Out[17]:
```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=1, normalize=False)
```

In [18]:
```
# print the intercept
print(lm.intercept_)
```

```
-2640159.796852679
```

In [19]:
```
coeff_df = pd.DataFrame(lm.coef_,X.columns,columns=['Coefficient'])
coeff_df
```

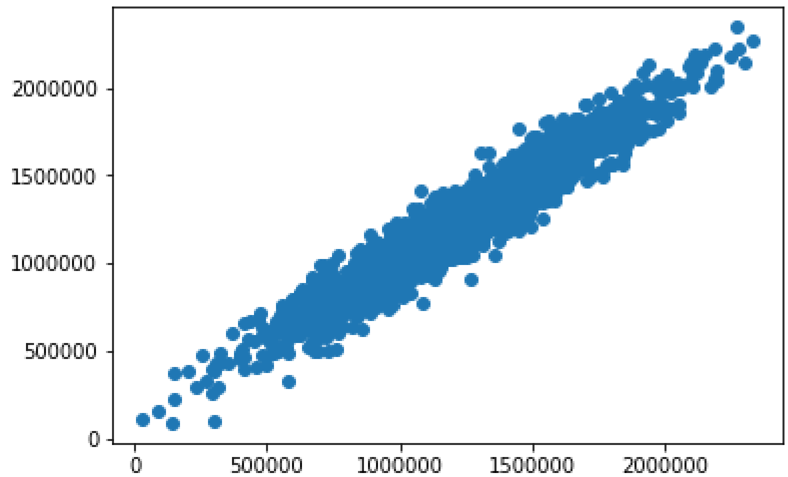
Out[19]:

	Coefficient
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 [20]:
```
predictions = lm.predict(X_test)
```

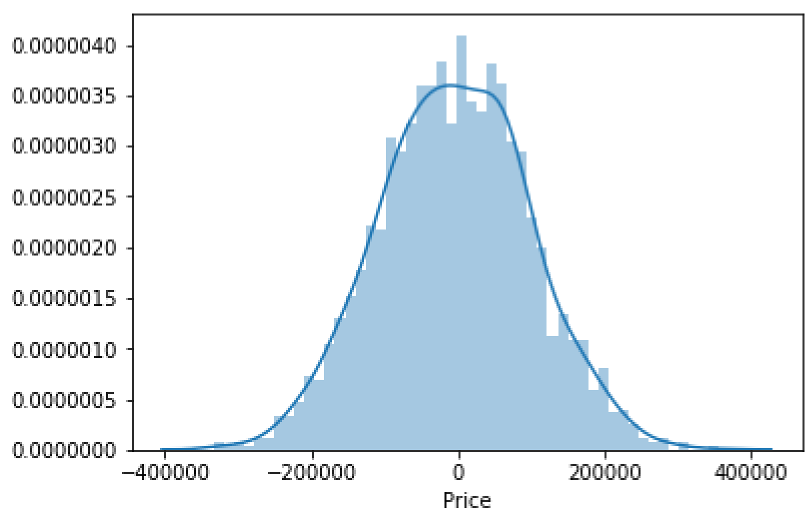
In [21]:
```
plt.scatter(y_test,predictions)
```

Out[21]:
```
<matplotlib.collections.PathCollection at 0xcada9b0>
```



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

```
C:\Users\q21\Anaconda3\lib\site-packages\matplotlib\axes\_axes.py:6462: UserWar
ning: The 'normed' kwarg is deprecated, and has been replaced by the 'density'
kwarg.
  warnings.warn("The 'normed' kwarg is deprecated, and has been "
```



In [23]:
```
from sklearn import metrics
```

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: 82288.22251914951
MSE: 10460958907.208992
RMSE: 102278.82922290904
```

Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

Mean Absolute Error (MAE) is the mean of the absolute value of the errors:

$$\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE) is the mean of the squared errors:

$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Root Mean Squared Error (RMSE) is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

```
In [ ]:
```