TASK-4 : HOUSE-PRICE-PREDICTION

In [2]:
```
#importing libraries to use the library functions
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.graph_objects as go
import plotly.express as px
```

In [3]:
```
#train contains the file information  which is in csv format
train=pd.read_csv('data.csv')
```

In [4]:
```
train
```

Out[4]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0
...	...	...	...	...	...	...	...	...	...
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0

4600 rows × 18 columns

In [5]:
```
#listing the column names of the dataset/dataframe
train.columns
```

Out[5]:
```
Index(['date', 'price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot',
       'floors', 'waterfront', 'view', 'condition', 'sqft_above',
       'sqft_basement', 'yr_built', 'yr_renovated', 'street', 'city',
       'statezip', 'country'],
      dtype='object')
```

In [6]:
```
#checking the datatypes of different columns of the dataframe
train.dtypes
```

Out[6]:
```
date              object
price            float64
bedrooms         float64
bathrooms        float64
sqft_living        int64
sqft_lot           int64
floors           float64
waterfront         int64
view               int64
condition          int64
sqft_above         int64
sqft_basement      int64
yr_built           int64
yr_renovated       int64
street            object
city              object
statezip          object
country           object
dtype: object
```

In [7]:
```
train.head
```

Out[7]:
```
<bound method NDFrame.head of                     date         price  bedrooms  bat
hrooms  sqft_living  \
0     2014-05-02 00:00:00  3.130000e+05       3.0       1.50         1340
1     2014-05-02 00:00:00  2.384000e+06       5.0       2.50         3650
2     2014-05-02 00:00:00  3.420000e+05       3.0       2.00         1930
3     2014-05-02 00:00:00  4.200000e+05       3.0       2.25         2000
4     2014-05-02 00:00:00  5.500000e+05       4.0       2.50         1940
...                   ...           ...       ...        ...          ...
4595  2014-07-09 00:00:00  3.081667e+05       3.0       1.75         1510
4596  2014-07-09 00:00:00  5.343333e+05       3.0       2.50         1460
4597  2014-07-09 00:00:00  4.169042e+05       3.0       2.50         3010
4598  2014-07-10 00:00:00  2.034000e+05       4.0       2.00         2090
4599  2014-07-10 00:00:00  2.206000e+05       3.0       2.50         1490

      sqft_lot  floors  waterfront  view  condition  sqft_above  \
0         7912     1.5           0     0          3        1340
1         9050     2.0           0     4          5        3370
2        11947     1.0           0     0          4        1930
3         8030     1.0           0     0          4        1000
4        10500     1.0           0     0          4        1140
...        ...     ...         ...   ...        ...         ...
4595      6360     1.0           0     0          4        1510
4596      7573     2.0           0     0          3        1460
4597      7014     2.0           0     0          3        3010
4598      6630     1.0           0     0          3        1070
4599      8102     2.0           0     0          4        1490
```

```
      sqft_basement  yr_built  yr_renovated                    street  \
0                 0      1955          2005      18810 Densmore Ave N
1               280      1921             0           709 W Blaine St
2                 0      1966             0  26206-26214 143rd Ave SE
3              1000      1963             0           857 170th Pl NE
4               800      1976          1992         9105 170th Ave NE
...             ...       ...           ...                       ...
4595              0      1954          1979           501 N 143rd St
4596              0      1983          2009          14855 SE 10th Pl
4597              0      2009             0          759 Ilwaco Pl NE
4598           1020      1974             0         5148 S Creston St
4599              0      1990             0         18717 SE 258th St

           city  statezip country
0     Shoreline  WA 98133     USA
1       Seattle  WA 98119     USA
2          Kent  WA 98042     USA
3      Bellevue  WA 98008     USA
4       Redmond  WA 98052     USA
...         ...       ...     ...
4595    Seattle  WA 98133     USA
4596   Bellevue  WA 98007     USA
4597     Renton  WA 98059     USA
4598    Seattle  WA 98178     USA
4599  Covington  WA 98042     USA

[4600 rows x 18 columns]>
```

In [32]:

```
train.tail
```

Out[32]:

```
<bound method NDFrame.tail of                        date         price  bedrooms  bat
hrooms  sqft_living  \
0     2014-05-02 00:00:00  3.130000e+05       3.0       1.50         1340
1     2014-05-02 00:00:00  2.384000e+06       5.0       2.50         3650
2     2014-05-02 00:00:00  3.420000e+05       3.0       2.00         1930
3     2014-05-02 00:00:00  4.200000e+05       3.0       2.25         2000
4     2014-05-02 00:00:00  5.500000e+05       4.0       2.50         1940
...                   ...           ...       ...        ...          ...
4595  2014-07-09 00:00:00  3.081667e+05       3.0       1.75         1510
4596  2014-07-09 00:00:00  5.343333e+05       3.0       2.50         1460
4597  2014-07-09 00:00:00  4.169042e+05       3.0       2.50         3010
4598  2014-07-10 00:00:00  2.034000e+05       4.0       2.00         2090
4599  2014-07-10 00:00:00  2.206000e+05       3.0       2.50         1490

      sqft_lot  floors  waterfront  view  condition  sqft_above  \
0         7912     1.5           0     0          3        1340
1         9050     2.0           0     4          5        3370
2        11947     1.0           0     0          4        1930
3         8030     1.0           0     0          4        1000
4        10500     1.0           0     0          4        1140
...        ...     ...         ...   ...        ...         ...
4595      6360     1.0           0     0          4        1510
4596      7573     2.0           0     0          3        1460
4597      7014     2.0           0     0          3        3010
4598      6630     1.0           0     0          3        1070
4599      8102     2.0           0     0          4        1490

      sqft_basement  yr_built  yr_renovated                    street  \
0                 0      1955          2005      18810 Densmore Ave N
1               280      1921             0           709 W Blaine St
2                 0      1966             0  26206-26214 143rd Ave SE
3              1000      1963             0           857 170th Pl NE
4               800      1976          1992         9105 170th Ave NE
...             ...       ...           ...                       ...
4595              0      1954          1979           501 N 143rd St
4596              0      1983          2009          14855 SE 10th Pl
4597              0      2009             0          759 Ilwaco Pl NE
4598           1020      1974             0         5148 S Creston St
```

```
4599              0      1990             0         18717 SE 258th St

           city  statezip country
0     Shoreline  WA 98133     USA
1       Seattle  WA 98119     USA
2          Kent  WA 98042     USA
3      Bellevue  WA 98008     USA
4       Redmond  WA 98052     USA
...         ...       ...     ...
4595    Seattle  WA 98133     USA
4596   Bellevue  WA 98007     USA
4597     Renton  WA 98059     USA
4598    Seattle  WA 98178     USA
4599  Covington  WA 98042     USA

[4600 rows x 18 columns]>
```

In [8]:
```
#shape of the dataframe ie no. of rows and columns
train.shape
```

Out[8]:
```
(4600, 18)
```

In [9]:
```
#checking for any duplicates in the data
train.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
#checking for any null values in the data
train.isnull().sum()
```

Out[10]:
```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
street           0
city             0
statezip         0
country          0
dtype: int64
```

In [11]:
```
#removing the null values in the data
train.dropna(inplace=True,axis=0)
```

In [12]:
```
#getting the information of dataframe such as no. of entries,data columns,non-null c
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 4600 entries, 0 to 4599
Data columns (total 18 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
```

```
 0   date           4600 non-null   object
 1   price          4600 non-null   float64
 2   bedrooms       4600 non-null   float64
 3   bathrooms      4600 non-null   float64
 4   sqft_living    4600 non-null   int64
 5   sqft_lot       4600 non-null   int64
 6   floors         4600 non-null   float64
 7   waterfront     4600 non-null   int64
 8   view           4600 non-null   int64
 9   condition      4600 non-null   int64
 10  sqft_above     4600 non-null   int64
 11  sqft_basement  4600 non-null   int64
 12  yr_built       4600 non-null   int64
 13  yr_renovated   4600 non-null   int64
 14  street         4600 non-null   object
 15  city           4600 non-null   object
 16  statezip       4600 non-null   object
 17  country        4600 non-null   object
dtypes: float64(4), int64(9), object(5)
memory usage: 682.8+ KB
```

In [13]:
```
#checking for statistical summary such as count,mean,etc. of numeric columns
train.describe()
```

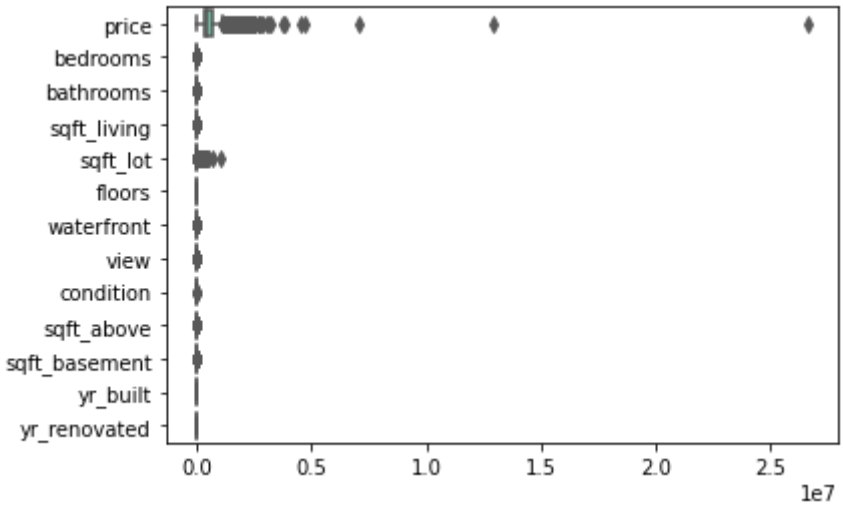
Out[13]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfro
count	4.600000e+03	4600.000000	4600.000000	4600.000000	4.600000e+03	4600.000000	4600.0000
mean	5.519630e+05	3.400870	2.160815	2139.346957	1.485252e+04	1.512065	0.0071
std	5.638347e+05	0.908848	0.783781	963.206916	3.588444e+04	0.538288	0.0844
min	0.000000e+00	0.000000	0.000000	370.000000	6.380000e+02	1.000000	0.0000
25%	3.228750e+05	3.000000	1.750000	1460.000000	5.000750e+03	1.000000	0.0000
50%	4.609435e+05	3.000000	2.250000	1980.000000	7.683000e+03	1.500000	0.0000
75%	6.549625e+05	4.000000	2.500000	2620.000000	1.100125e+04	2.000000	0.0000
max	2.659000e+07	9.000000	8.000000	13540.000000	1.074218e+06	3.500000	1.0000

DATA VISUALIZATION

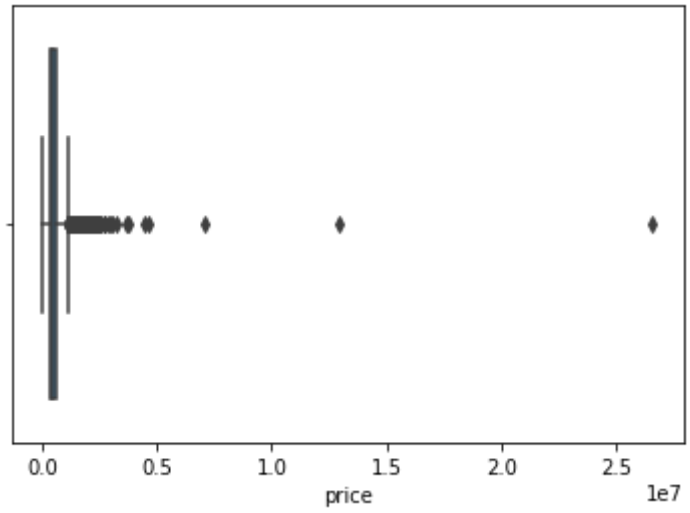
In [14]:
```
#checking for any outliers in the data
sns.boxplot(data=train,orient='h',palette='Set2')
```

Out[14]: <AxesSubplot:>



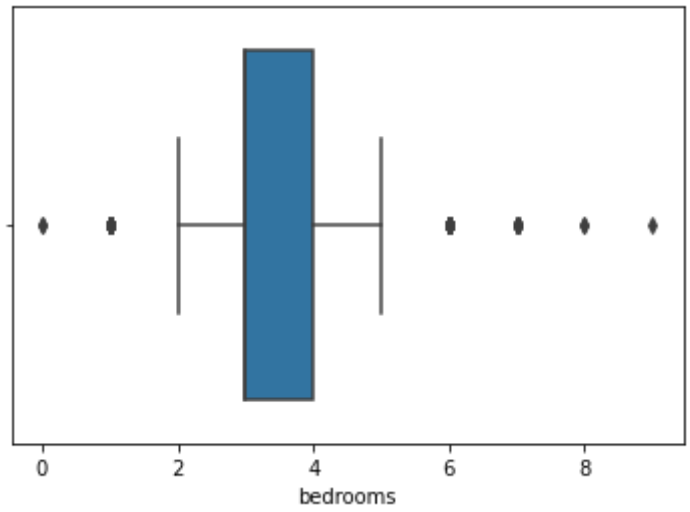
In [15]:
```
sns.boxplot(x=train['price'])
```

Out[15]: <AxesSubplot:xlabel='price'>



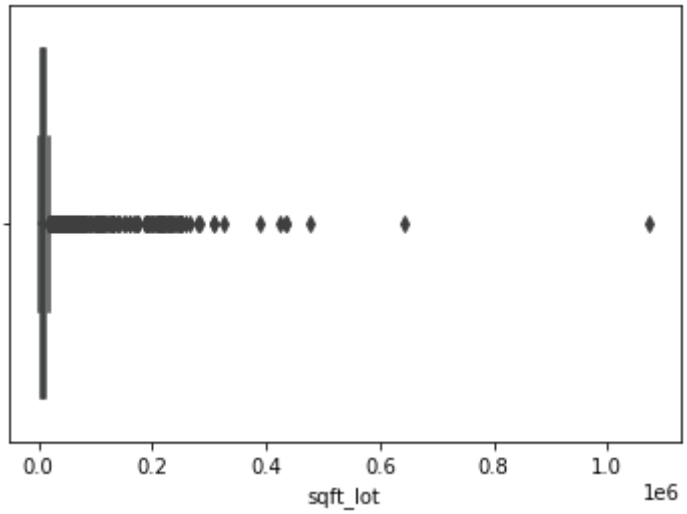
In [16]:
```
sns.boxplot(x=train['bedrooms'])
```

Out[16]: <AxesSubplot:xlabel='bedrooms'>



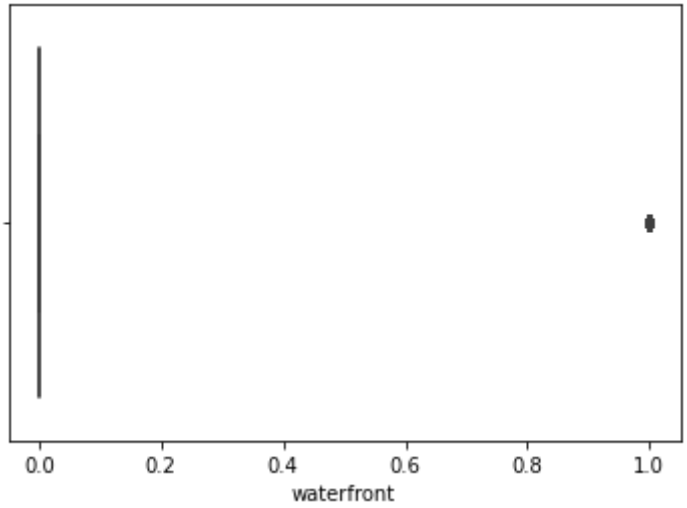
In [17]:
```
sns.boxplot(x=train['sqft_lot'])
```

Out[17]: <AxesSubplot:xlabel='sqft_lot'>



In [18]:
```
sns.boxplot(x=train['waterfront'])
```

Out[18]: <AxesSubplot:xlabel='waterfront'>



In [19]:
```
Q1 = train.quantile(0.25)
Q3 = train.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
price            332087.50
bedrooms              1.00
bathrooms             0.75
sqft_living        1160.00
sqft_lot           6000.50
floors                1.00
waterfront            0.00
view                  0.00
condition             1.00
sqft_above         1110.00
sqft_basement       610.00
yr_built             46.00
yr_renovated       1999.00
dtype: float64
```

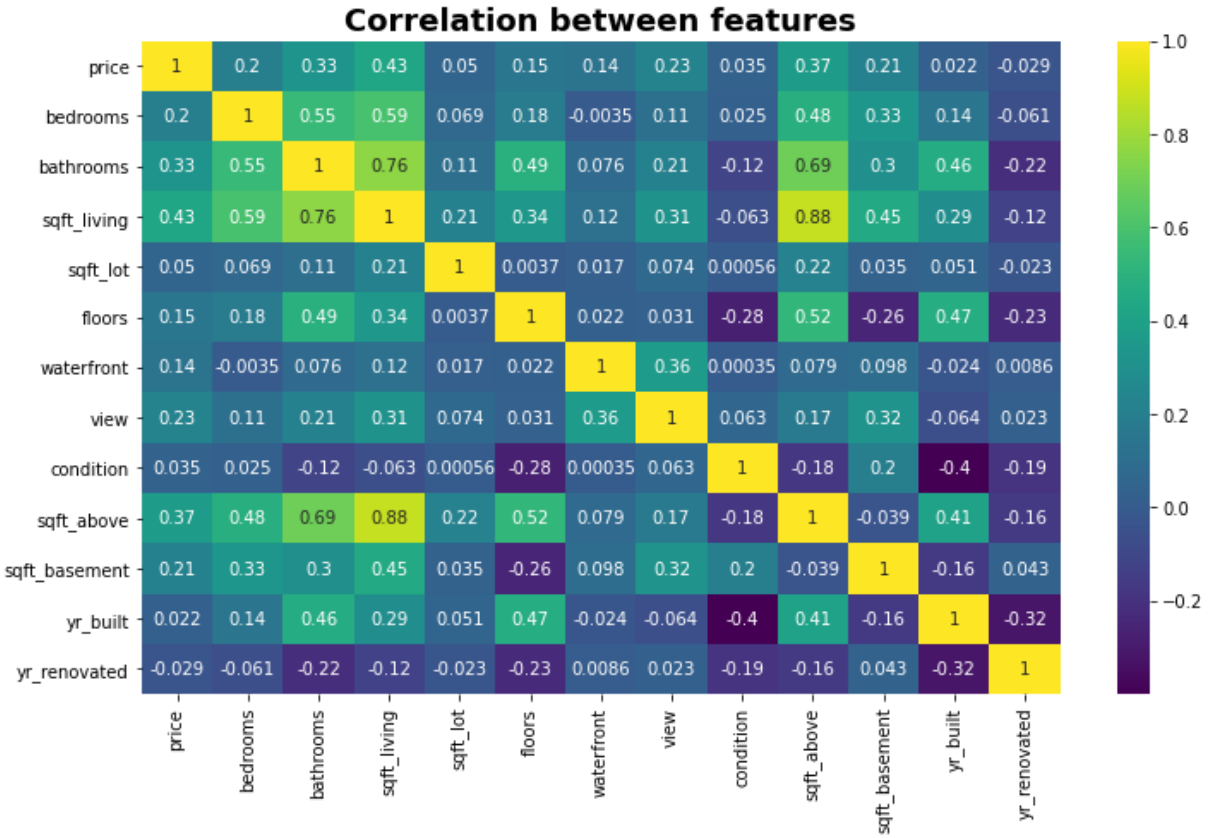
In [20]:
```
#finding the correlation between different variables/features
train.corr()
```

Out[20]:

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	
price	1.000000	0.200336	0.327110	0.430410	0.050451	0.151461	0.135648	0.22
bedrooms	0.200336	1.000000	0.545920	0.594884	0.068819	0.177895	-0.003483	0.11
bathrooms	0.327110	0.545920	1.000000	0.761154	0.107837	0.486428	0.076232	0.21
sqft_living	0.430410	0.594884	0.761154	1.000000	0.210538	0.344850	0.117616	0.31
sqft_lot	0.050451	0.068819	0.107837	0.210538	1.000000	0.003750	0.017241	0.07
floors	0.151461	0.177895	0.486428	0.344850	0.003750	1.000000	0.022024	0.03
waterfront	0.135648	-0.003483	0.076232	0.117616	0.017241	0.022024	1.000000	0.36
view	0.228504	0.111028	0.211960	0.311009	0.073907	0.031211	0.360935	1.00
condition	0.034915	0.025080	-0.119994	-0.062826	0.000558	-0.275013	0.000352	0.06
sqft_above	0.367570	0.484705	0.689918	0.876443	0.216455	0.522814	0.078911	0.17
sqft_basement	0.210427	0.334165	0.298020	0.447206	0.034842	-0.255510	0.097501	0.32
yr_built	0.021857	0.142461	0.463498	0.287775	0.050706	0.467481	-0.023563	-0.06
yr_renovated	-0.028774	-0.061082	-0.215886	-0.122817	-0.022730	-0.233996	0.008625	0.02

In [21]:

```
train_corr=train.corr()
f,ax=plt.subplots(figsize=(12,7))
sns.heatmap(train_corr,cmap='viridis',annot=True)
plt.title("Correlation between features",weight='bold',fontsize=18)
plt.show()
```



In [22]:

```
#X = train.drop('yr_renovated', axis = 1)
#y = train['yr_renovated']
```

MACHINE LEARNING ALGORITHMS

In [23]:
```
X = train[['bedrooms','bathrooms','sqft_living','sqft_lot','floors','waterfront','vi

y = train['price']
```

In [24]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_stat
```

1. Linear Regression

In [25]:
```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()

lr.fit(X_train,y_train)
```

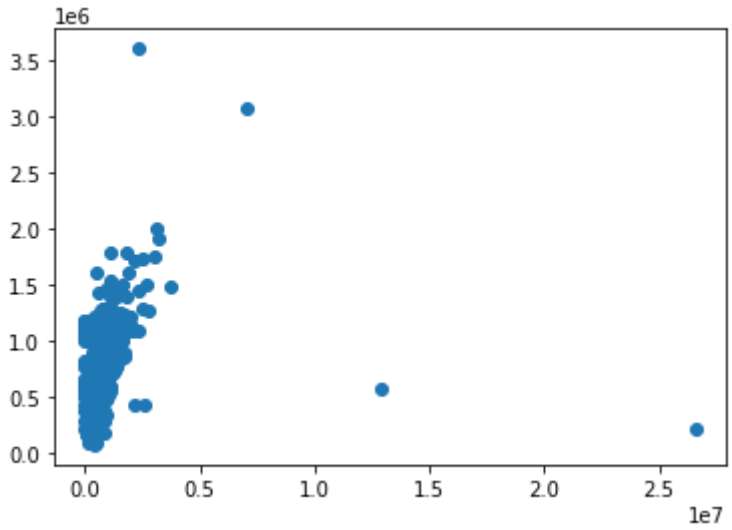
Out[25]:
```
LinearRegression()
```

In [26]:
```
print(lr.intercept_)
```

```
5399305.140280506
```

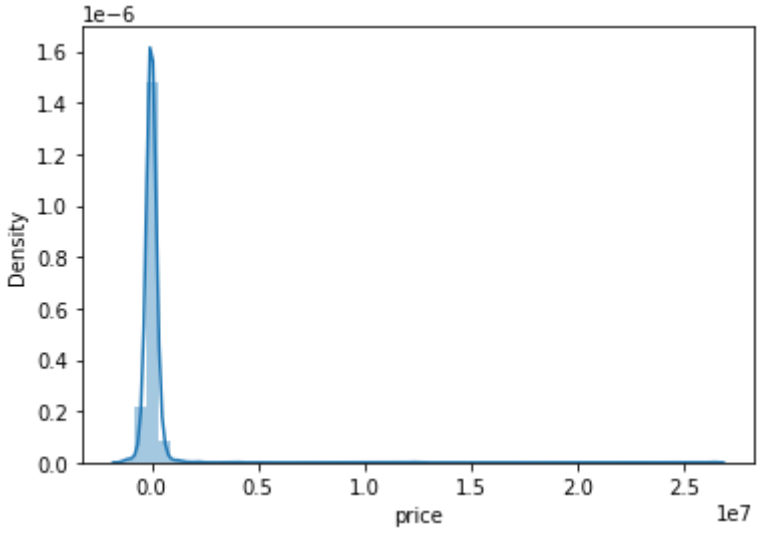
In [27]:
```
predictions = lr.predict(X_test)

plt.scatter(y_test,predictions)
```

Out[27]:
```
<matplotlib.collections.PathCollection at 0x16ee205f5f8>
```



In [28]:
```
sns.distplot((y_test-predictions),bins=50);
```

```
c:\users\vrinda bajaj\python 3.7.2\lib\site-packages\seaborn\distributions.py:2557:
FutureWarning: `distplot` is a deprecated function and will be removed in a future v
ersion. Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



2. Random Forest

In [29]:

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=1000)
rf.fit(X_train, y_train)

ts_pred = rf.predict(X_test)
tr_pred = rf.predict(X_train)

print('Test set evaluation:')
print('_____________________________________')
print(y_test, ts_pred)

print('Train set evaluation:')
print('_____________________________________')
print(y_train, tr_pred)
```

```
Test set evaluation:
_____________________________________
4032    1360000.0
1558     332000.0
2004     343000.0
3186     660000.0
4176     310000.0
          ...
1105     220000.0
3797     535000.0
166      425000.0
154      609000.0
1196     622500.0
Name: price, Length: 1840, dtype: float64 [1383493.616       475629.2457619   35660
8.50119048 ...  430437.89385348
  470961.7996211   643025.62466667]
Train set evaluation:
_____________________________________
695     707000.0
1170    555000.0
684     267800.0
2490    129000.0
2882    549000.0
          ...
4079    513000.0
4171    749950.0
599     450000.0
1361    283200.0
1547    550000.0
```

```
Name: price, Length: 2760, dtype: float64 [741504.51695402 534002.68608333 266882.78
356833 ... 476731.67756522
 311271.14633332 623831.58385714]
```

3. SVR(Support Vector Regression)

In [30]:

```
from sklearn.svm import SVR

svm_reg = SVR(kernel='rbf', C=1000000, epsilon=0.001)
svm_reg.fit(X_train, y_train)

ts_pred = svm_reg.predict(X_test)
tr_pred = svm_reg.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print(y_test, ts_pred)

print('Train set evaluation:\n_____________________________________')
print(y_train, tr_pred)
```

```
Test set evaluation:
_____________________________________
4032    1360000.0
1558     332000.0
2004     343000.0
3186     660000.0
4176     310000.0
          ...
1105     220000.0
3797     535000.0
166      425000.0
154      609000.0
1196     622500.0
Name: price, Length: 1840, dtype: float64 [752707.21935021 531742.26537051 552002.93
350059 ... 362249.72311489
 526832.44857172 683358.14134946]
Train set evaluation:
_____________________________________
695     707000.0
1170    555000.0
684     267800.0
2490    129000.0
2882    549000.0
          ...
4079    513000.0
4171    749950.0
599     450000.0
1361    283200.0
1547    550000.0
Name: price, Length: 2760, dtype: float64 [762197.76154306 533657.1306912  242238.30
880289 ... 570887.62337239
 505539.35742764 563357.98319783]
```

4. Lasso

In [31]:

```
from sklearn.linear_model import Lasso

model = Lasso(alpha=0.1,
              precompute=True,
              positive=True,
              selection='random',
              random_state=42)
model.fit(X_train, y_train)
```

```
ts_pred = model.predict(X_test)
tr_pred = model.predict(X_train)

print('Test set evaluation:\n_____________________________________')
print(y_test, ts_pred)
print('Train set evaluation:\n_____________________________________')
print(y_train, tr_pred)
```

```
Test set evaluation:
_____________________________________
4032    1360000.0
1558     332000.0
2004     343000.0
3186     660000.0
4176     310000.0
          ...
1105     220000.0
3797     535000.0
166      425000.0
154      609000.0
1196     622500.0
Name: price, Length: 1840, dtype: float64 [1363093.1137011   534772.16246687  54853
7.20485388 ...  430901.27224793
  513624.49285387  713112.3109121 ]
Train set evaluation:
_____________________________________
695     707000.0
1170    555000.0
684     267800.0
2490    129000.0
2882    549000.0
          ...
4079    513000.0
4171    749950.0
599     450000.0
1361    283200.0
1547    550000.0
Name: price, Length: 2760, dtype: float64 [820129.05353557 560684.05933548 226339.62
173257 ... 647351.91226946
 527386.68920434 634859.15043455]
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: