House price prediction

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn import preprocessing, svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [6]:
```
df=pd.read_csv(r"C:\Users\anu\Downloads\data.csv")
df
```

Out[6]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_basement	yr_built	yr_renovate
0	2014-05-02 00:00:00	3.130000e+05	3.0	1.50	1340	7912	1.5	0	0	3	1340	0	1955	200
1	2014-05-02 00:00:00	2.384000e+06	5.0	2.50	3650	9050	2.0	0	4	5	3370	280	1921	
2	2014-05-02 00:00:00	3.420000e+05	3.0	2.00	1930	11947	1.0	0	0	4	1930	0	1966	
3	2014-05-02 00:00:00	4.200000e+05	3.0	2.25	2000	8030	1.0	0	0	4	1000	1000	1963	
4	2014-05-02 00:00:00	5.500000e+05	4.0	2.50	1940	10500	1.0	0	0	4	1140	800	1976	199
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
4595	2014-07-09 00:00:00	3.081667e+05	3.0	1.75	1510	6360	1.0	0	0	4	1510	0	1954	197
4596	2014-07-09 00:00:00	5.343333e+05	3.0	2.50	1460	7573	2.0	0	0	3	1460	0	1983	200
4597	2014-07-09 00:00:00	4.169042e+05	3.0	2.50	3010	7014	2.0	0	0	3	3010	0	2009	
4598	2014-07-10 00:00:00	2.034000e+05	4.0	2.00	2090	6630	1.0	0	0	3	1070	1020	1974	
4599	2014-07-10 00:00:00	2.206000e+05	3.0	2.50	1490	8102	2.0	0	0	4	1490	0	1990	

4600 rows × 18 columns

In [7]:
```
df=df[['sqft_living','sqft_lot']]
df.columns=['living','lot']
```

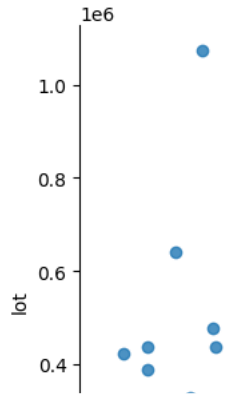
In [8]:
```
df.head(10)
```

Out[8]:

	living	lot
0	1340	7912
1	3650	9050
2	1930	11947
3	2000	8030
4	1940	10500
5	880	6380
6	1350	2560
7	2710	35868
8	2430	88426
9	1520	6200

In [9]:
```
sns.lmplot(x ="living", y= "lot", data = df,order = 2, ci = None)
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x70f4e7b6d0>
```



In [10]:
```
df.describe()
```

Out[10]:

	living	lot
count	4600.000000	4.600000e+03
mean	2139.346957	1.485252e+04
std	963.206916	3.588444e+04
min	370.000000	6.380000e+02
25%	1460.000000	5.000750e+03
50%	1980.000000	7.683000e+03
75%	2620.000000	1.100125e+04
max	13540.000000	1.074218e+06

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   living  4600 non-null   int64
 1   lot     4600 non-null   int64
dtypes: int64(2)
memory usage: 72.0 KB
```

In [12]:
```
df.fillna(method = 'ffill',inplace = True)
```

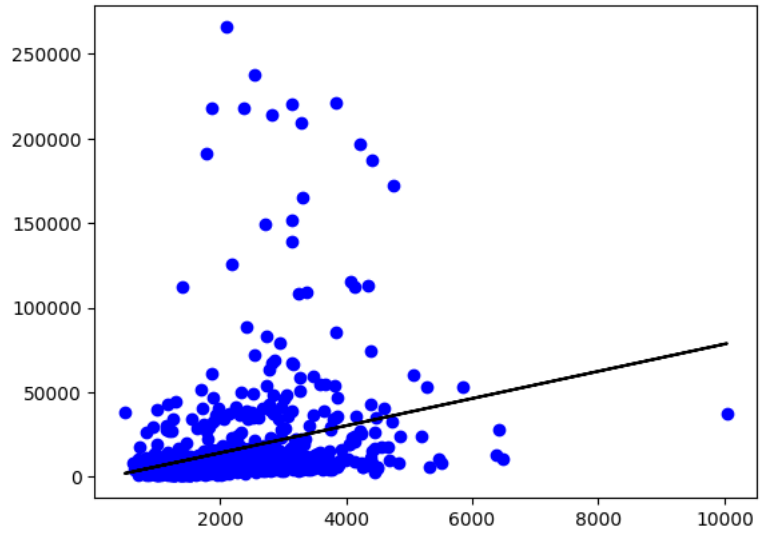
```
C:\Users\anu\AppData\Local\Temp\ipykernel_11440\1930596415.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(method = 'ffill',inplace = True)
```

In [13]:
```
X = np. array(df['living']).reshape(-1, 1)
Y = np.array(df['lot']).reshape(-1, 1)
```

In [14]:
```
X_train,X_test,Y_train,Y_test=train_test_split(X,Y,test_size=0.25)
regr = LinearRegression()
regr.fit(X_train, Y_train)
print(regr.score(X_test, Y_test))
```

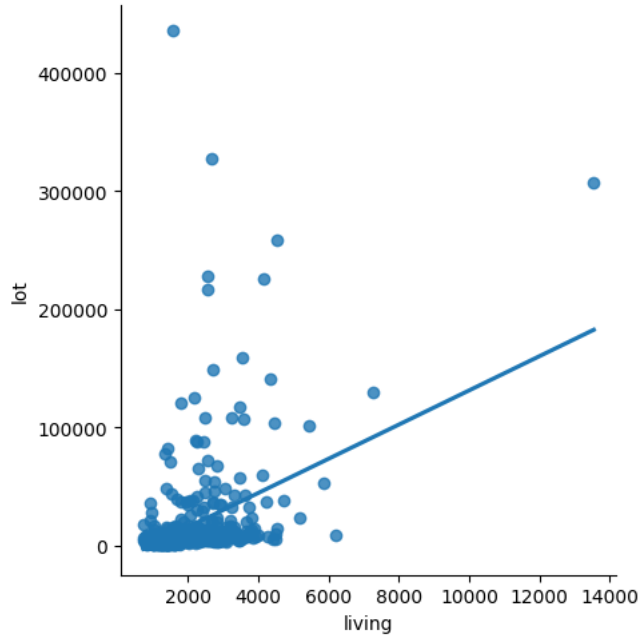
```
0.06406645172236913
```

In [15]:
```
y_pred=regr.predict(X_test)
plt.scatter(X_test, Y_test,color='b')
plt.plot(X_test,y_pred,color='k')
plt.show()
```



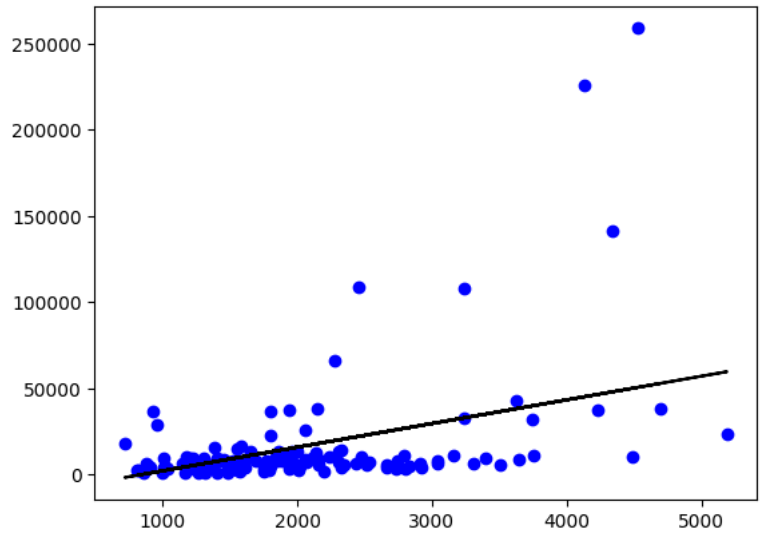
In [16]:
```
df500 = df[:][:500]
sns.lmplot(x ="living",y ="lot", data = df500,order = 1,ci = None)
```

Out[16]: <seaborn.axisgrid.FacetGrid at 0x70e0260410>



In [17]:
```
df500.fillna(method = 'ffill',inplace = True)
X = np. array(df500['living']).reshape(-1, 1)
y = np.array(df500['lot']).reshape(-1, 1)
df500.dropna(inplace = True)
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.25)
regr = LinearRegression()
regr.fit(X_train,y_train)
print("Regression:",regr.score(X_test,y_test))
y_pred=regr.predict(X_test)
plt.scatter(X_test,y_test,color = 'b')
plt.plot(X_test,y_pred,color = 'k')
plt.show()
```

```
Regression: 0.20369364619369335
```



In [18]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
model = LinearRegression()
model.fit(X_train,y_train)
y_pred = model.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2 score:",r2)
```

```
R2 score: 0.20369364619369335
```

In [ ]: