In [1]:
```
#Linear Regression model
#step1:problem statement-How Best Fit The Dataset?
```

In [2]:
```
#step 1:importing all the required libraries
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 [4]:
```
#step 2:reading the dataset
s=pd.read_csv(r"C:\Users\mouni\Downloads\data (1).csv")
s
```

Out[4]:

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

4600 rows × 18 columns

In [5]:
```
s=s[['sqft_living', 'sqft_lot']]
s.columns=['living', 'lot']
```

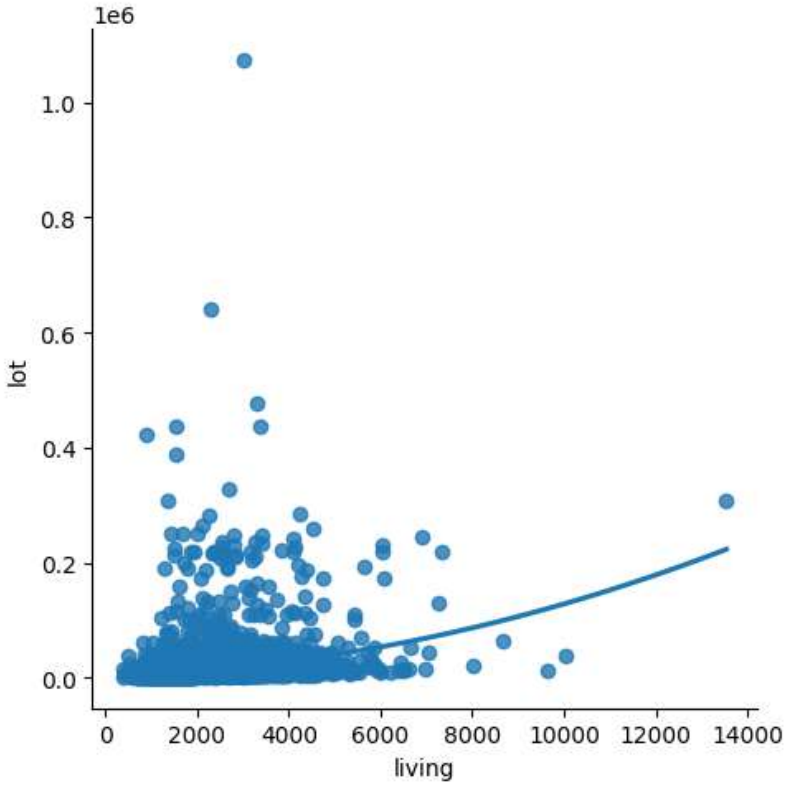
In [6]:
```
df.head(20)
```

Out[6]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	sqft_above	sqft_base
0	2014-05-02 00:00:00	313000.0	3.0	1.50	1340	7912	1.5	0	0	3	1340	
1	2014-05-02 00:00:00	2384000.0	5.0	2.50	3650	9050	2.0	0	4	5	3370	
2	2014-05-02 00:00:00	342000.0	3.0	2.00	1930	11947	1.0	0	0	4	1930	
3	2014-05-02 00:00:00	420000.0	3.0	2.25	2000	8030	1.0	0	0	4	1000	
4	2014-05-02 00:00:00	550000.0	4.0	2.50	1940	10500	1.0	0	0	4	1140	
5	2014-05-02 00:00:00	490000.0	2.0	1.00	880	6380	1.0	0	0	3	880	
6	2014-05-02 00:00:00	335000.0	2.0	2.00	1350	2560	1.0	0	0	3	1350	
7	2014-05-02 00:00:00	482000.0	4.0	2.50	2710	35868	2.0	0	0	3	2710	
8	2014-05-02 00:00:00	452500.0	3.0	2.50	2430	88426	1.0	0	0	4	1570	
9	2014-05-02 00:00:00	640000.0	4.0	2.00	1520	6200	1.5	0	0	3	1520	
10	2014-05-02 00:00:00	463000.0	3.0	1.75	1710	7320	1.0	0	0	3	1710	
11	2014-05-02 00:00:00	1400000.0	4.0	2.50	2920	4000	1.5	0	0	5	1910	
12	2014-05-02 00:00:00	588500.0	3.0	1.75	2330	14892	1.0	0	0	3	1970	
13	2014-05-02 00:00:00	365000.0	3.0	1.00	1090	6435	1.0	0	0	4	1090	
14	2014-05-02 00:00:00	1200000.0	5.0	2.75	2910	9480	1.5	0	0	3	2910	
15	2014-05-02 00:00:00	242500.0	3.0	1.50	1200	9720	1.0	0	0	4	1200	
16	2014-05-02 00:00:00	419000.0	3.0	1.50	1570	6700	1.0	0	0	4	1570	
17	2014-05-02 00:00:00	367500.0	4.0	3.00	3110	7231	2.0	0	0	3	3110	
18	2014-05-02 00:00:00	257950.0	3.0	1.75	1370	5858	1.0	0	0	3	1370	
19	2014-05-02 00:00:00	275000.0	3.0	1.50	1180	10277	1.0	0	0	3	1180	

In [8]:
```
#step-3:explaining the data scatter -plotting the data scatter
sns.lmplot(x='living',y='lot',data=s,order=2,ci=None)
```

Out[8]: <seaborn.axisgrid.FacetGrid at 0x16662c63790>



In [13]:
```
s.describe()
```

Out[13]:

	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 [14]:
```
s.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 [15]:
```
#step-4:data cleaning-eliminating nan or missing input numbers
s.fillna(method='ffill')
```

Out[15]:

	living	lot
0	1340	7912
1	3650	9050
2	1930	11947
3	2000	8030
4	1940	10500
...	...	...
4595	1510	6360
4596	1460	7573
4597	3010	7014
4598	2090	6630
4599	1490	8102

4600 rows × 2 columns

In [16]:
```
#step-5:training our model
x=np.array(s['living']).reshape(-1,1)
y=np.array(s['lot']).reshape(-1,1)
```

In [17]:
```
s.dropna(inplace=True)
```

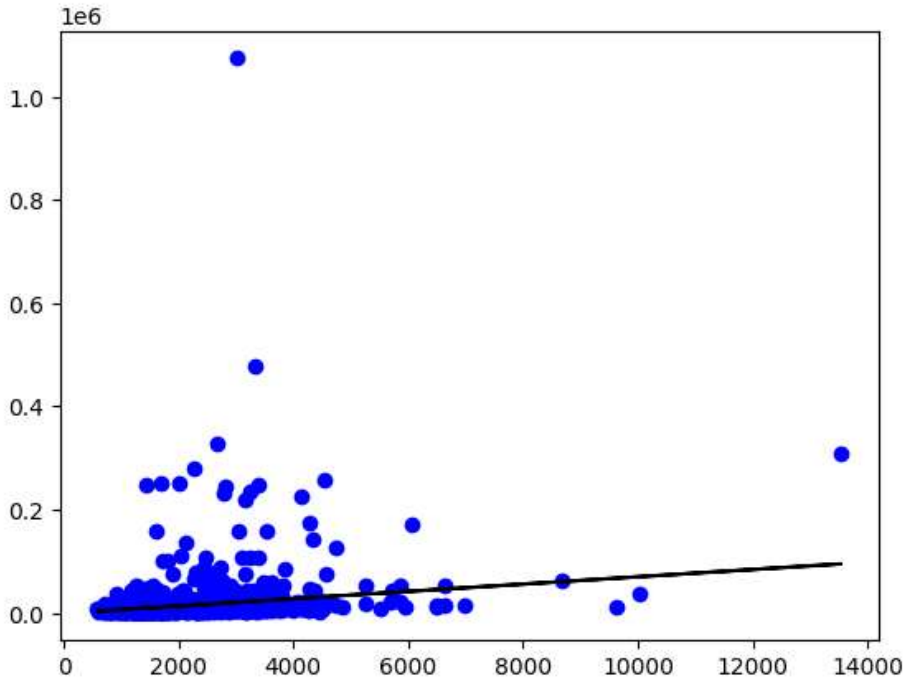
```
C:\Users\mouni\AppData\Local\Temp\ipykernel_9076\3276880757.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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)
  s.dropna(inplace=True)
```

In [18]:
```
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))
```

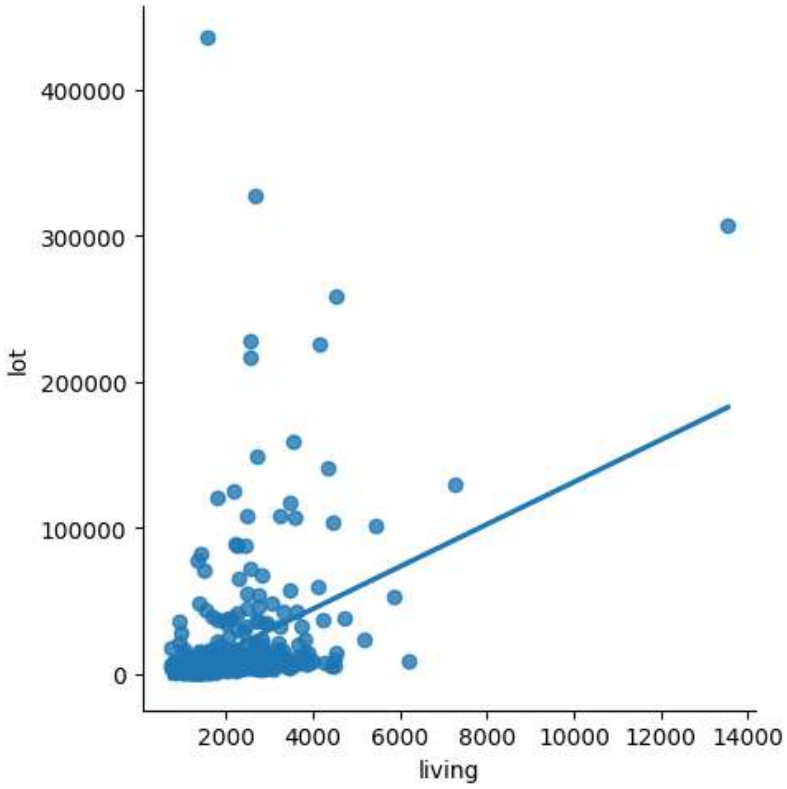
```
0.04451600860876814
```

In [19]:
```
#step-6:exploring our results
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



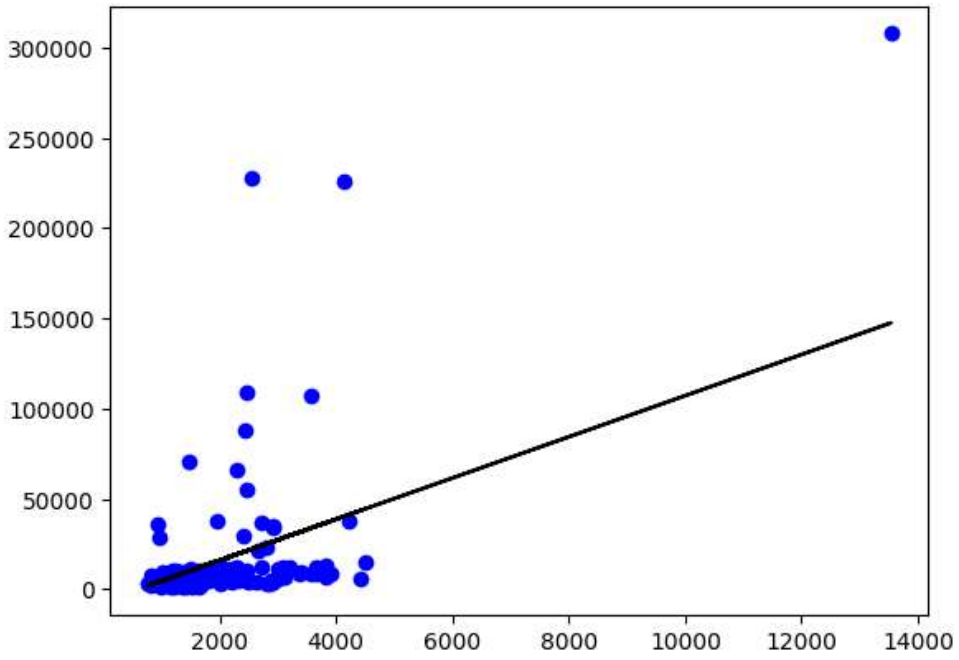
In [21]:
```
#step-7:working with a smallest dataset
s500=s[:][:500]
sns.lmplot(x="living",y="lot",data=s500,order=1,ci=None)
```

Out[21]: <seaborn.axisgrid.FacetGrid at 0x16676ccbdc0>



In [22]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(s500['living']).reshape(-1,1)
y=np.array(s500['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: 0.3175357866960121
```



In [23]:
```
#step-8:evaluation of model
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: 0.3175357866960121
```

In [24]:
```
#step-9:conclusion
#dataset we have taken is poor for linear model but with the smaller data works well with linear mode
```

In [ ]:
```
```