Linear Regression- House price prediction

In [13]:

```
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 [2]:
```
#Step-2: Reading the Dataset
df=pd.read_csv(r"C:\Users\sudheer\Downloads\data.csv")
df
```

Out[2]:

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

4600 rows × 18 columns

In [3]:
```
df=df[['sqft_living','sqft_lot']]
df.columns=['living','lot']
```

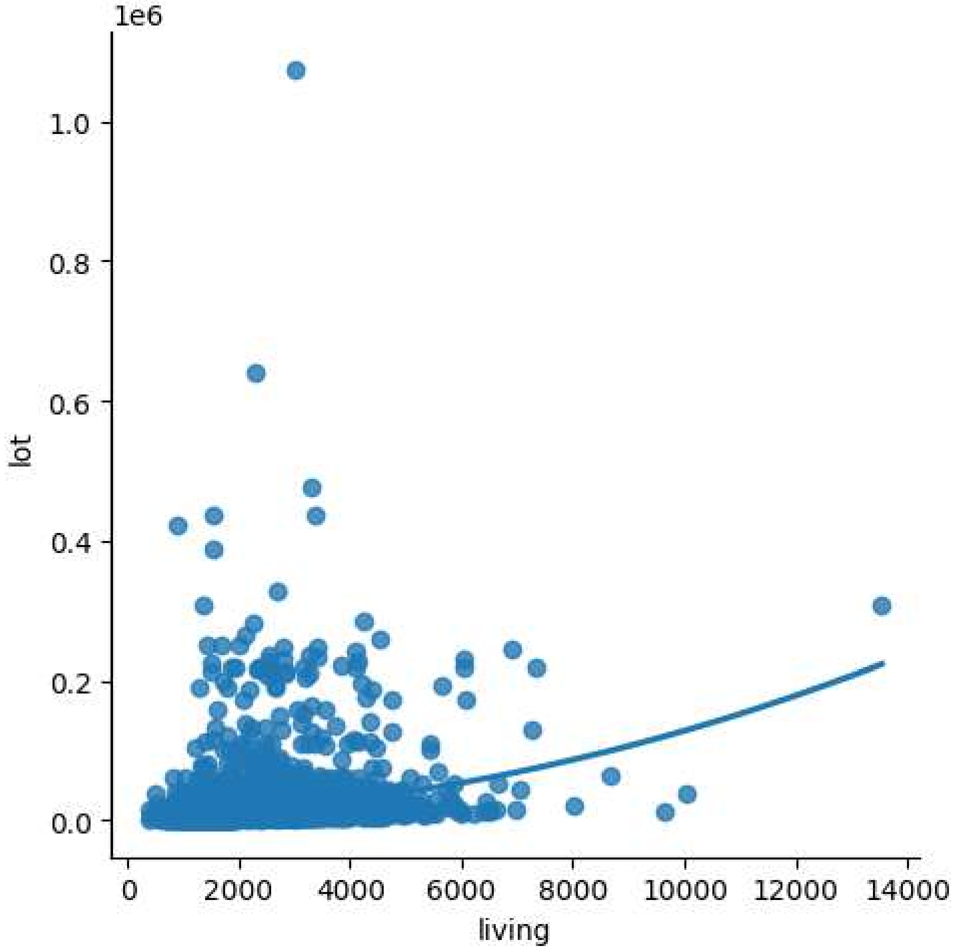
In [4]:
```
df.head()
```

Out[4]:
```
   living    lot
0    1340   7912
1    3650   9050
2    1930  11947
3    2000   8030
4    1940  10500
```

In [5]:

```
#Step-3: Exploring the Data Scatter - plotting the data scatter
sns.lmplot(x="living",y="lot", data = df, order = 2, ci = None)
df.describe()
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 [6]:
```
#Step-4: Data cleaning - Eliminating NaN OR missing input numbers
df.fillna(method ='ffill', inplace = True)
```

```
C:\Users\sudheer\AppData\Local\Temp\ipykernel_26720\3221840372.py:2: SettingW
ithCopyWarning:
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/s
table/user_guide/indexing.html#returning-a-view-versus-a-copy (https://panda
s.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-ver
sus-a-copy)
  df.fillna(method ='ffill', inplace = True)
```

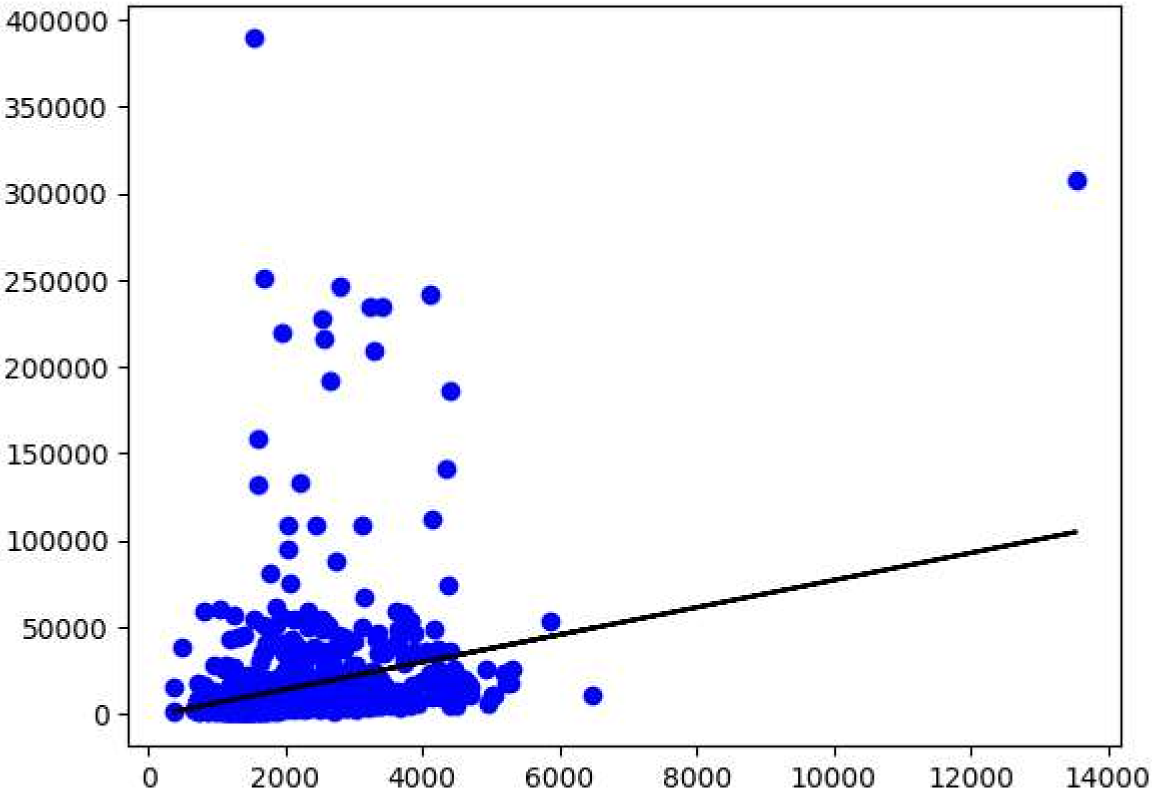
In [7]:
```
# Step-5: Training Our Model
X = np.array(df['living']).reshape(-1, 1)
y = np.array(df['lot']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

In [8]:
```
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size = 0.25)
# Splitting the data into training data and test data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.06668335348446652
```

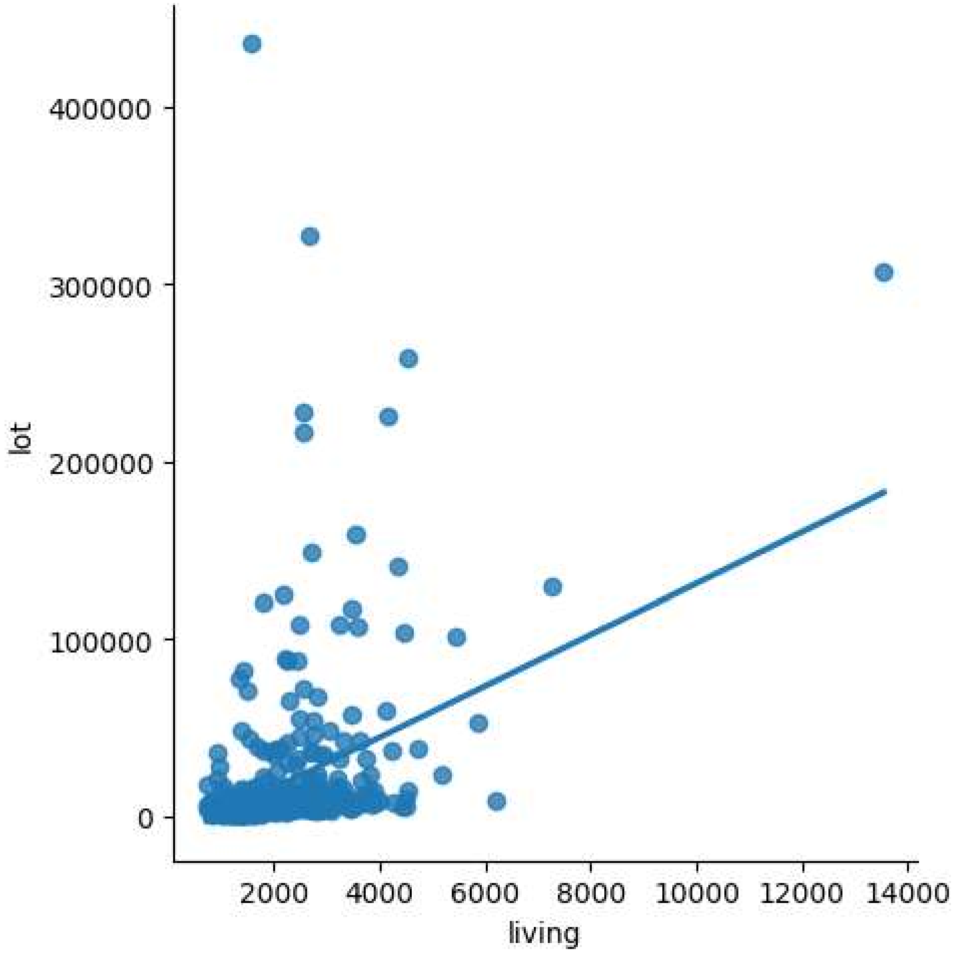
In [9]:

```
#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()
# Data scatter of predicted values
```



In [10]:
```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x = "living", y ="lot", data = df500, order = 1, ci = None)
```

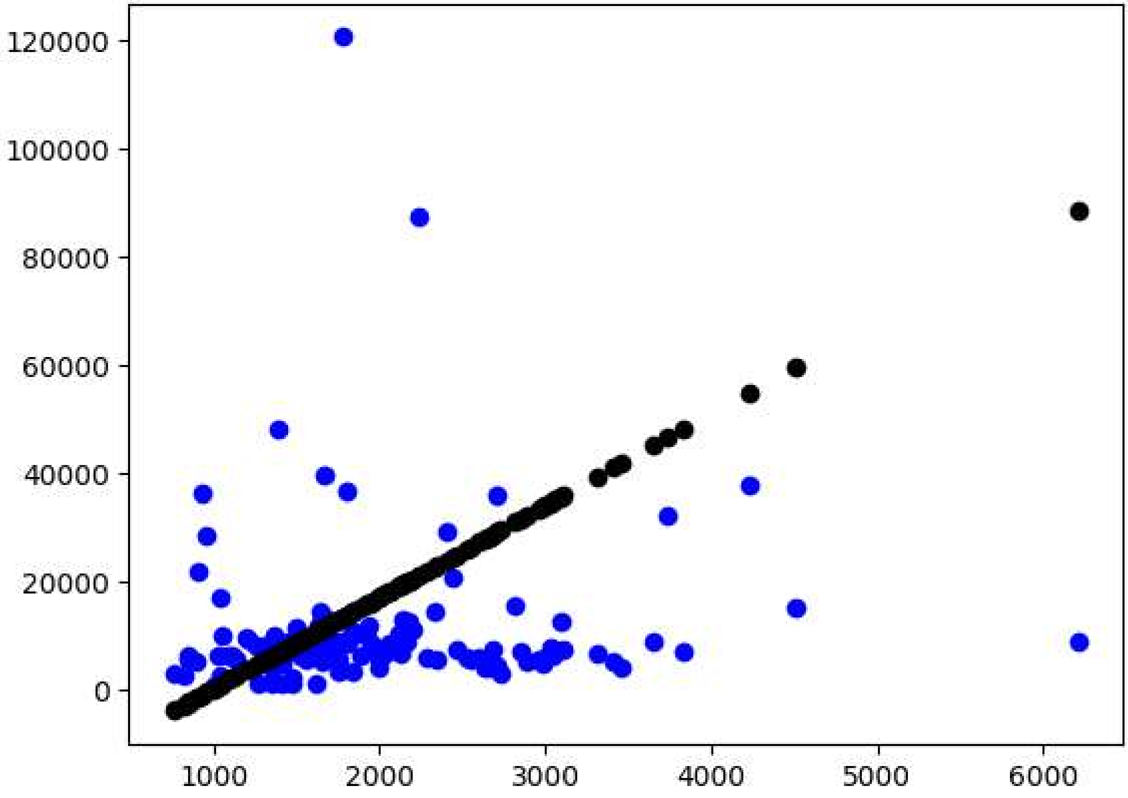
Out[10]:
```
<seaborn.axisgrid.FacetGrid at 0x149943fd8d0>
```



In [11]:

```
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.scatter(X_test, y_pred, color = 'k')
plt.show()
```

```
Regression: -1.0422777500952733
```



In [12]:

```
#Step-8: Evaluation of model

from sklearn.linear_model import LinearRegression

from sklearn.metrics import r2_score

#Train the model

model = LinearRegression()

model.fit(X_train, y_train)

#Evaluating the model on the test set

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)

print("R2 score:",r2)
```

```
R2 score: -1.0422777500952733
```

Step-9:Conclusion:

Dataset we have taken is poor for Linear Model,but with the smaller data works well with Linear Model.

In [ ]:

```
```