In [27]:
```
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 [28]:
```
df=pd.read_csv(r"C:\Users\rubin\Downloads\bottle.csv.zip")
df
```

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_13020\3996045872.py:1: DtypeWarni
ng: Columns (47,73) have mixed types. Specify dtype option on import or set l
ow_memory=False.
  df=pd.read_csv(r"C:\Users\rubin\Downloads\bottle.csv.zip")
```

Out[28]:

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
0	1	1	054.0 056.0	19-4903CR-HY-060-0930-05400560-0000A-3	0	10.500	33.4400	NaN	25.64900	Na
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	Na
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	Na
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	Na
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	Na
...	...	...	...	...	...	...	...	...	...	
864858	34404	864859	093.4 026.4	20-1611SR-MX-310-2239-09340264-0000A-7	0	18.744	33.4083	5.805	23.87055	108.
864859	34404	864860	093.4 026.4	20-1611SR-MX-310-2239-09340264-0002A-3	2	18.744	33.4083	5.805	23.87072	108.
864860	34404	864861	093.4 026.4	20-1611SR-MX-310-2239-09340264-0005A-3	5	18.692	33.4150	5.796	23.88911	108.
864861	34404	864862	093.4 026.4	20-1611SR-MX-310-2239-09340264-0010A-3	10	18.161	33.4062	5.816	24.01426	107.

	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297	105.

864863 rows × 74 columns

In [29]:
```
df=df[['Salnty','T_degC']]
df.columns=['Sal','Temp']
```

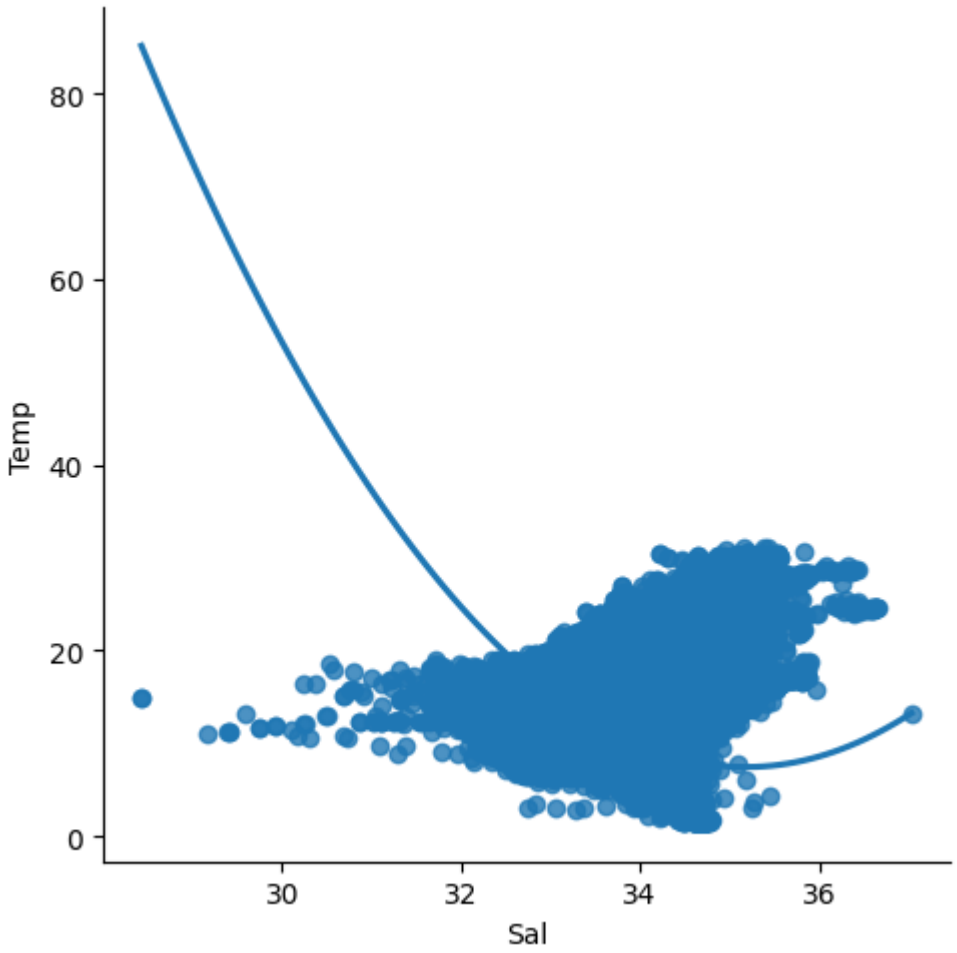
In [19]:
```
df.head(10)
```

Out[19]:

	Sal	Temp
0	33.440	10.50
1	33.440	10.46
2	33.437	10.46
3	33.420	10.45
4	33.421	10.45
5	33.431	10.45
6	33.440	10.45
7	33.424	10.24
8	33.420	10.06
9	33.494	9.86

In [36]:
```
sns.lmplot(x="Sal",y="Temp",data=df,order=2,ci=None)
```

Out[36]:
```
<seaborn.axisgrid.FacetGrid at 0x1c06f785120>
```



In [37]:
```
df.describe()
```

Out[37]:

	Sal	Temp
count	817509.000000	853900.000000
mean	33.840350	10.799677
std	0.461843	4.243825
min	28.431000	1.440000
25%	33.488000	7.680000
50%	33.863000	10.060000
75%	34.196900	13.880000
max	37.034000	31.140000

In [38]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 864863 entries, 0 to 864862
Data columns (total 2 columns):
 #   Column  Non-Null Count   Dtype  
---  ------  --------------   -----  
 0   Sal     817509 non-null  float64
 1   Temp    853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [39]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_13020\4116506308.py:1: SettingWit
hCopyWarning:
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 [40]:
```
x=np.array(df['Sal']).reshape(-1,1)
y=np.array(df['Temp']).reshape(-1,1)
```

In [41]:
```
df.dropna(inplace=True)
```

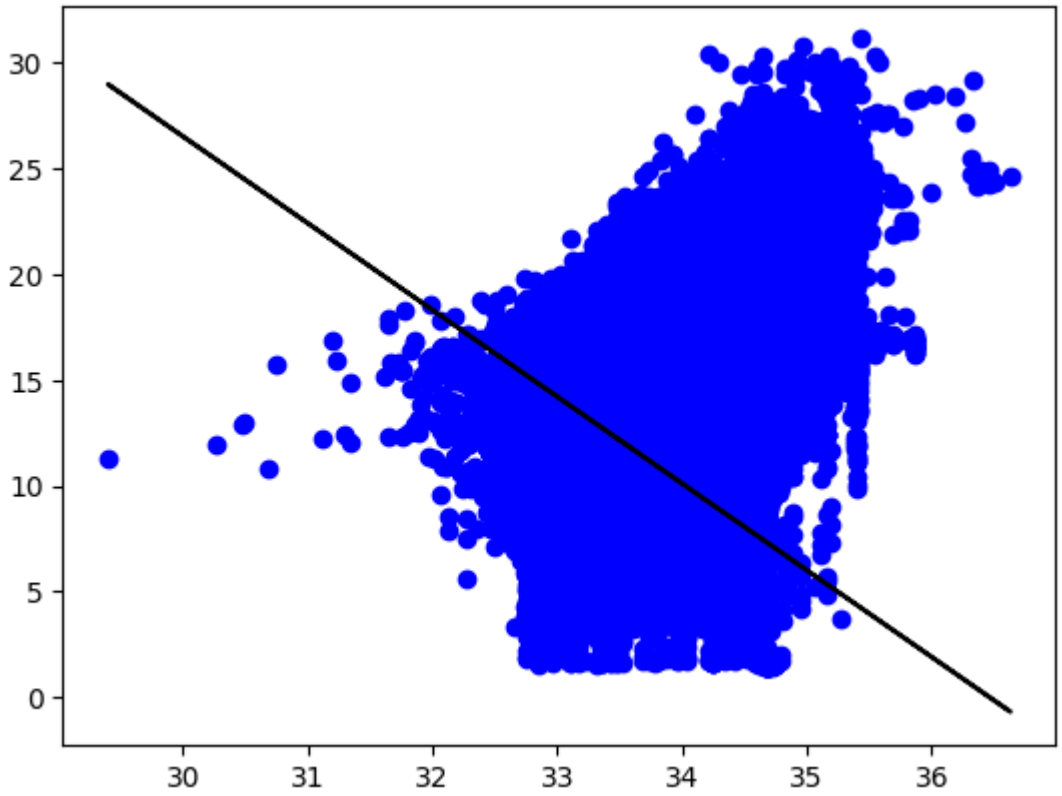
```
C:\Users\rubin\AppData\Local\Temp\ipykernel_13020\1379821321.py:1: SettingWit
hCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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.dropna(inplace=True)
```

In [44]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```

In [45]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.2042328493125658
```

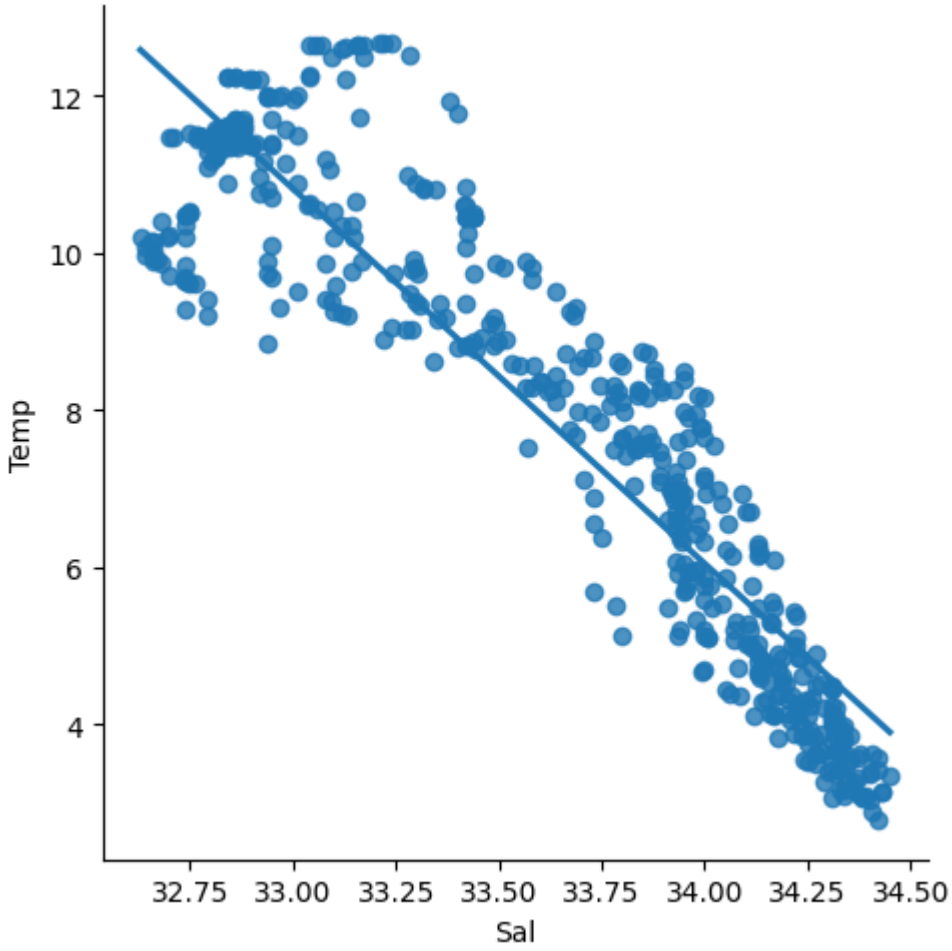
In [47]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [48]:
```
df500=df[:][:500]
```

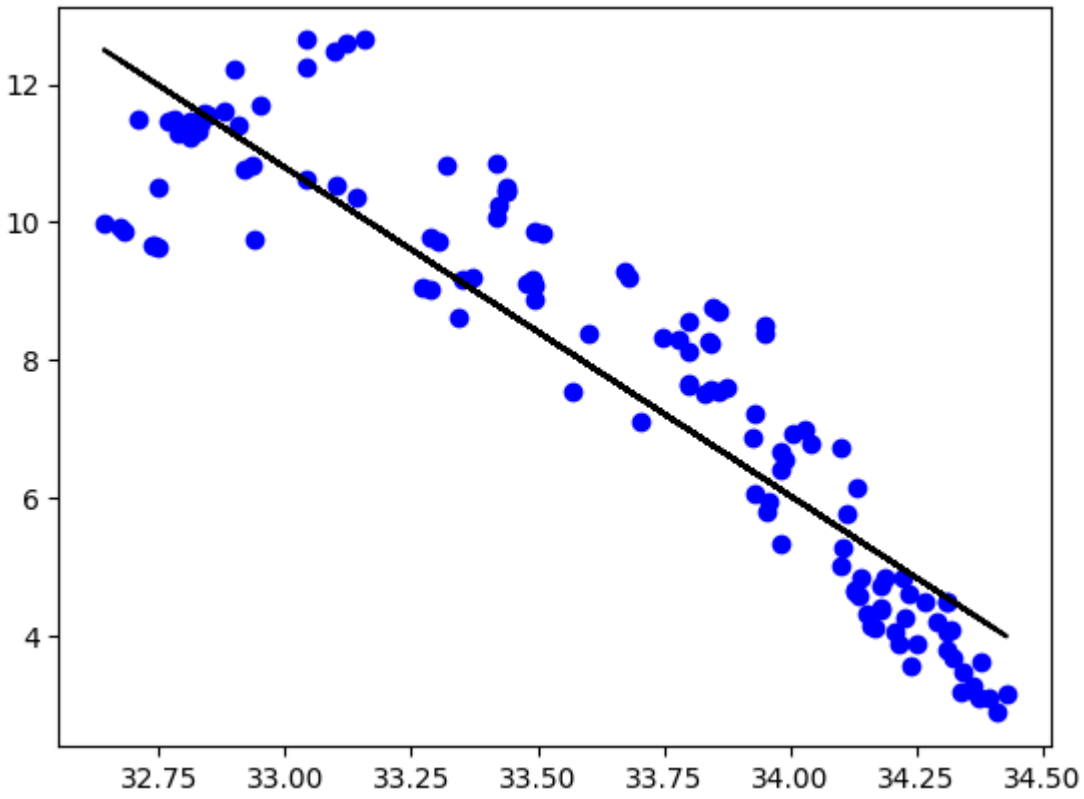
In [49]:
```
sns.lmplot(x="Sal",y="Temp",data=df500,order=1,ci=None)
```

Out[49]:
```
<seaborn.axisgrid.FacetGrid at 0x1c0810edf00>
```



In [51]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['Sal']).reshape(-1,1)
y=np.array(df500['Temp']).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("Regresion:",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()
```

```
Regresion: 0.8413182838843871
```



In [53]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [55]:
```
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: 0.8413182838843871
```

vehicles data set

In [4]:
```
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\rubin\Downloads\fiat500_VehicleSelection_Dataset.csv
df
```

Out[6]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1	lounge	51	882	25000	1	44.907242	8.611560
1	2	pop	51	1186	32500	1	45.666359	12.241890
2	3	sport	74	4658	142228	1	45.503300	11.417840
3	4	lounge	51	2739	160000	1	40.633171	17.634609
4	5	pop	73	3074	106880	1	41.903221	12.495650
...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870
1535	1536	pop	51	2223	60457	1	45.481541	9.413480
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270
1537	1538	pop	51	1766	54276	1	40.323410	17.568270

1538 rows × 9 columns

In [7]:
```
df=df[['age_in_days','km']]
df.columns=['age','distance in km']
```

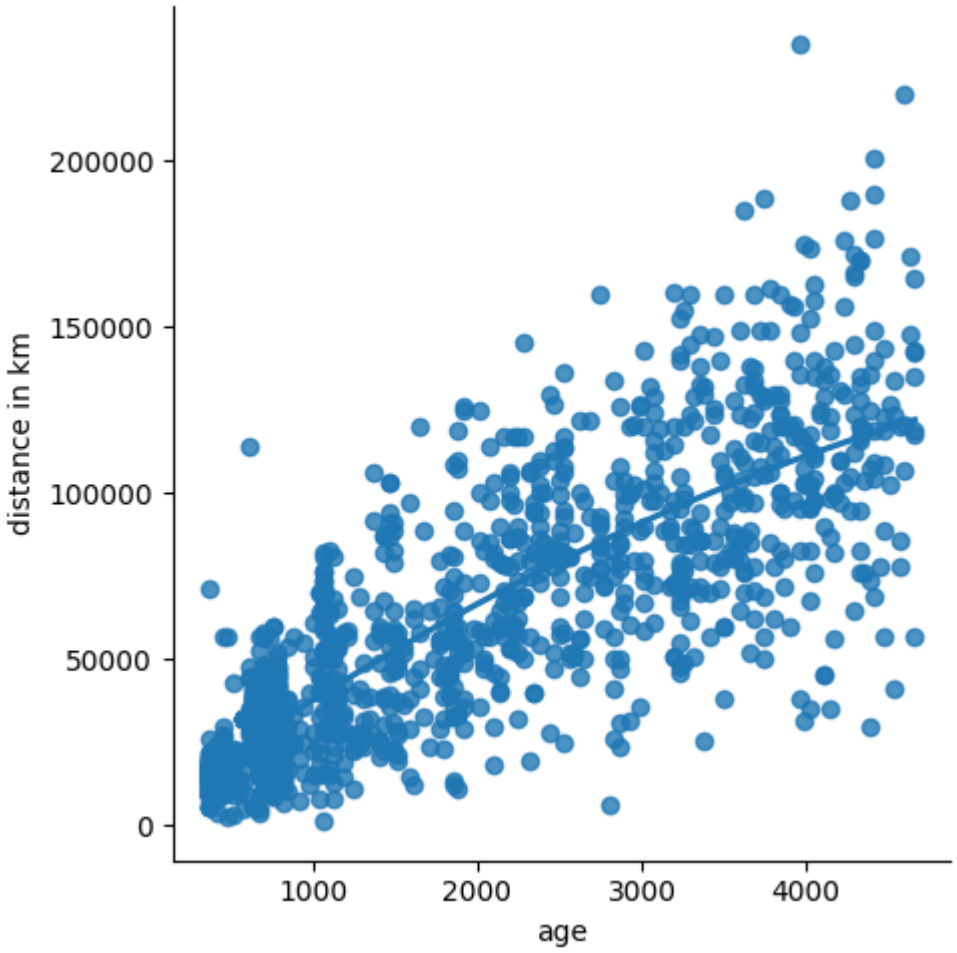
In [8]:
```
df.head(10)
```

Out[8]:

	age	distance in km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
5	3623	70225
6	731	11600
7	1521	49076
8	4049	76000
9	3653	89000

In [9]:
```
sns.lmplot(x="age",y="distance in km",data=df,order=2,ci=None)
```

Out[9]:
```
<seaborn.axisgrid.FacetGrid at 0x1b2400f5ab0>
```



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

Out[10]:

	age	distance in km
count	1538.000000	1538.000000
mean	1650.980494	53396.011704
std	1289.522278	40046.830723
min	366.000000	1232.000000
25%	670.000000	20006.250000
50%	1035.000000	39031.000000
75%	2616.000000	79667.750000
max	4658.000000	235000.000000

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 2 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             1538 non-null   int64
 1   distance in km  1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

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

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_6980\4116506308.py:1: SettingWith
CopyWarning:
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 [13]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['distance in km']).reshape(-1,1)
```

In [14]:
```
df.dropna(inplace=True)
```

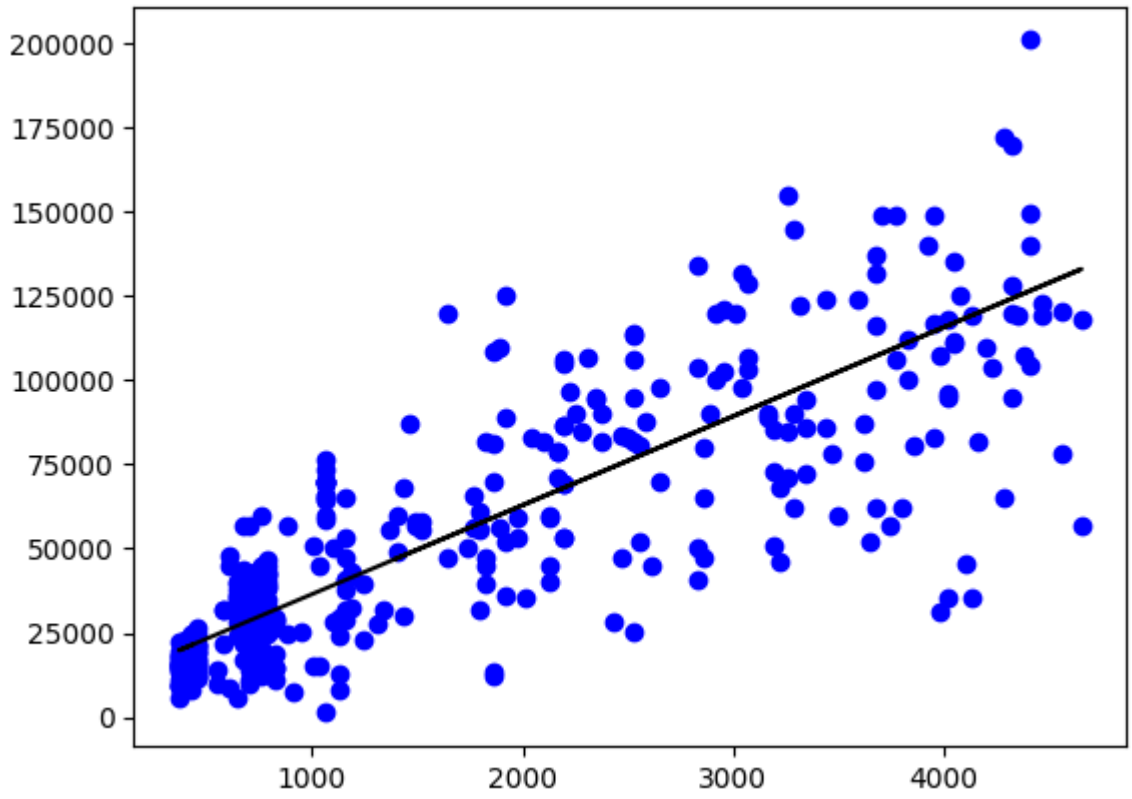
```
C:\Users\rubin\AppData\Local\Temp\ipykernel_6980\1379821321.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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.dropna(inplace=True)
```

In [15]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.26)
```

In [16]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.6711479452695219
```

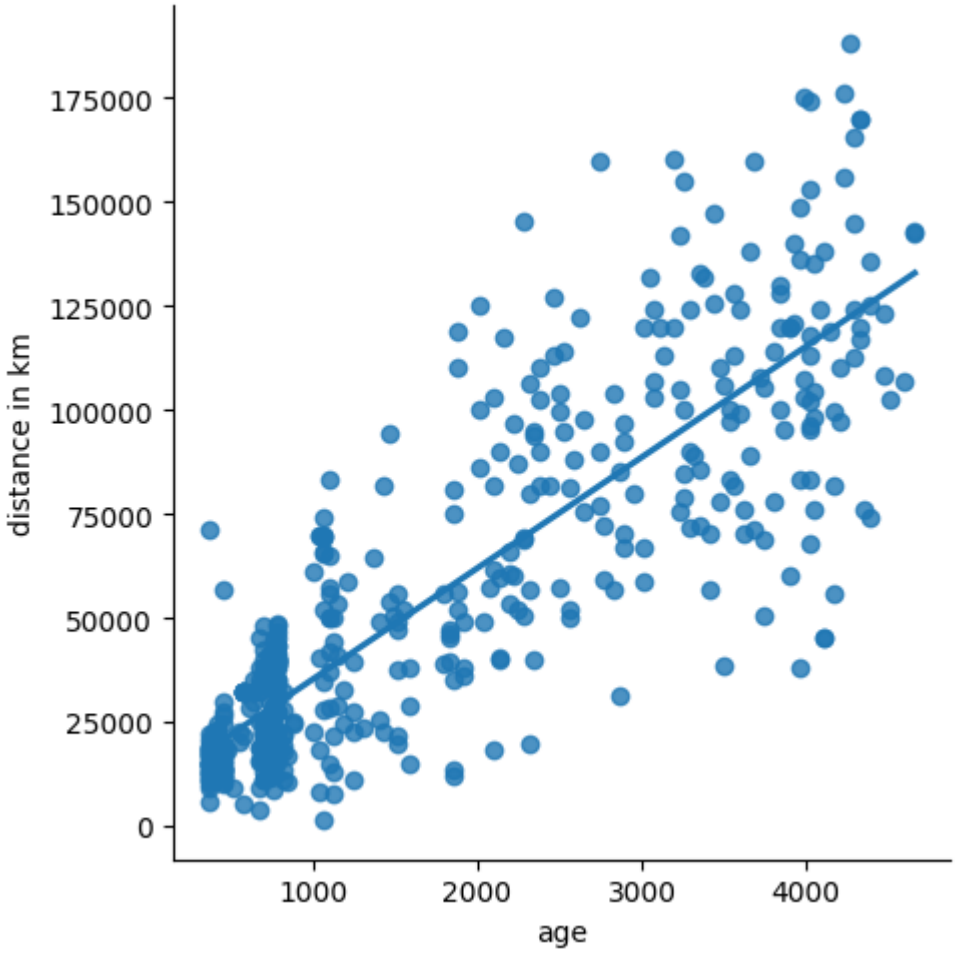
In [17]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [20]:
```
df500=df[:][:500]
```

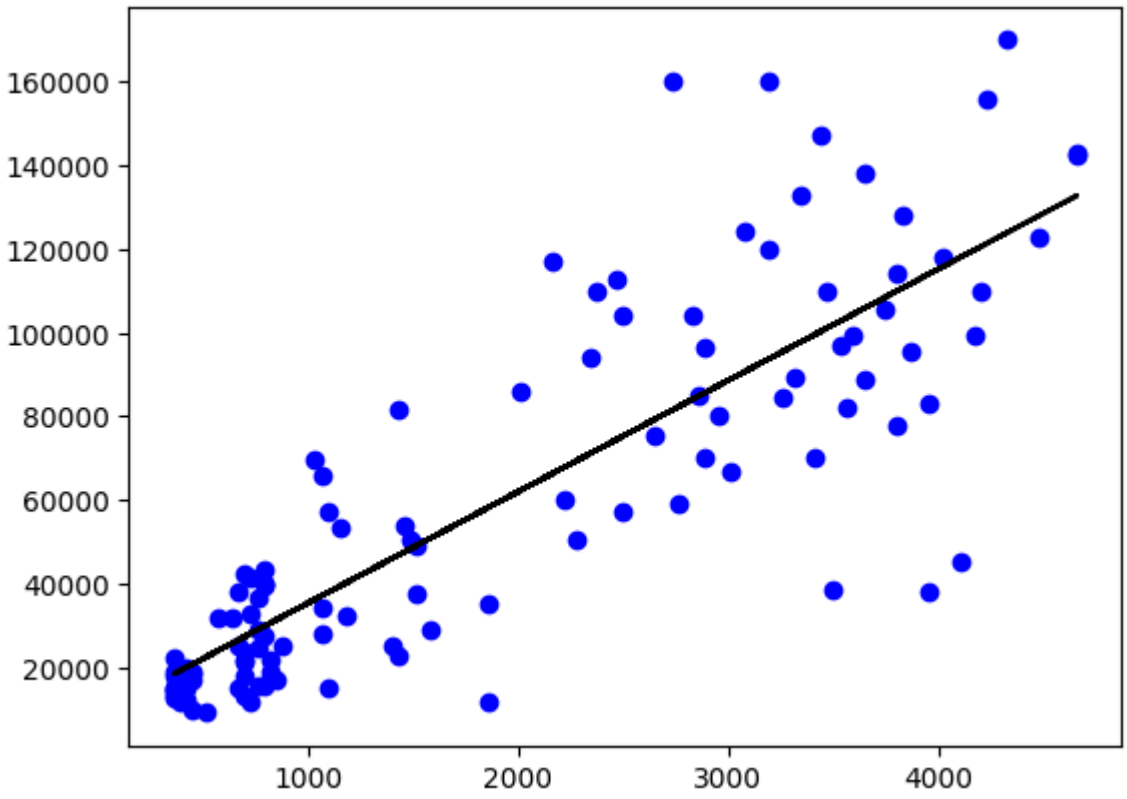
In [21]:
```
sns.lmplot(x="age",y="distance in km",data=df500,order=1,ci=None)
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x1b240369960>
```



In [22]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['age']).reshape(-1,1)
y=np.array(df500['distance in km']).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("Regresion:",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()
```

```
Regresion: 0.7280940037857009
```



In [23]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [25]:
```
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: 0.7280940037857009
```

House price prediction

In [7]:
```
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 [8]:
```
df=pd.read_csv(r"C:\Users\rubin\Downloads\data.csv")
df
```

Out[8]:

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	viev
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 [9]:
```
df=df[['price','sqft_living']]
df.columns=['cost','living area']
```

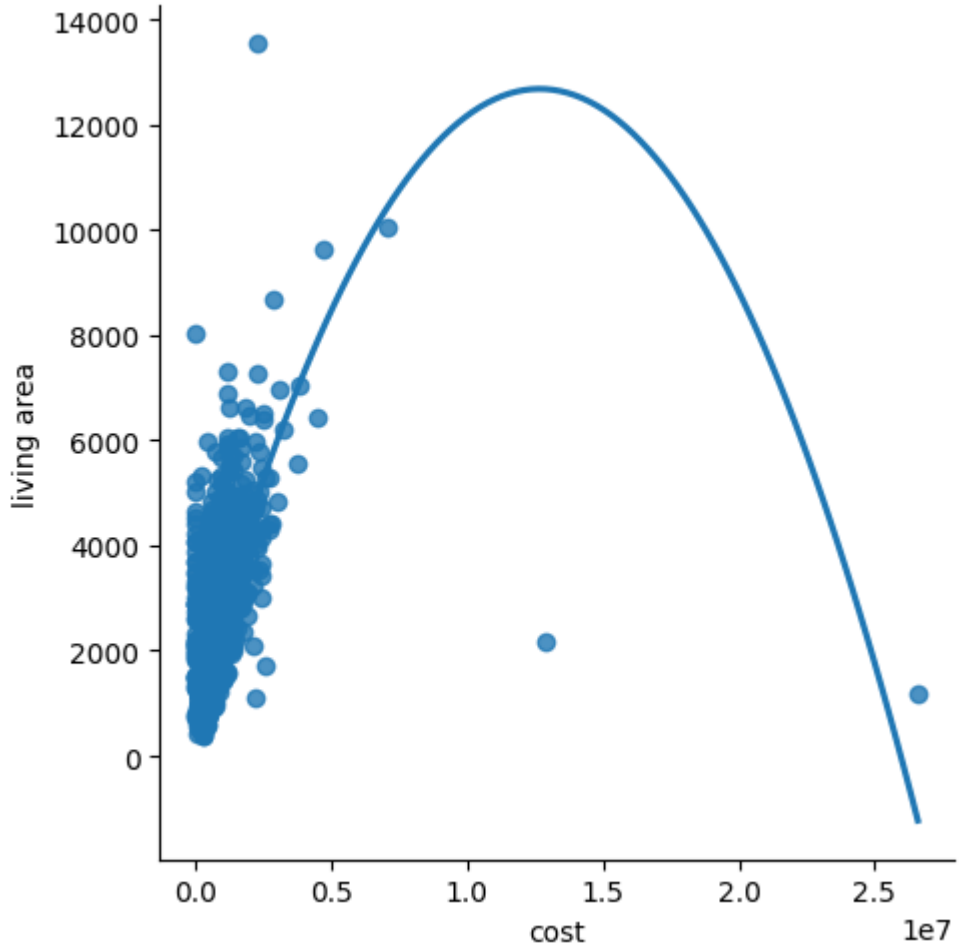
In [10]:
```
df.head(15)
```

Out[10]:

	cost	living area
0	313000.0	1340
1	2384000.0	3650
2	342000.0	1930
3	420000.0	2000
4	550000.0	1940
5	490000.0	880
6	335000.0	1350
7	482000.0	2710
8	452500.0	2430
9	640000.0	1520
10	463000.0	1710
11	1400000.0	2920
12	588500.0	2330
13	365000.0	1090
14	1200000.0	2910

In [11]:
```
sns.lmplot(x="cost",y="living area",data=df,order=2,ci=None)
```

Out[11]:
```
<seaborn.axisgrid.FacetGrid at 0x2648065f040>
```



In [12]:
```
df.describe()
```

Out[12]:

	cost	living area
count	4.600000e+03	4600.000000
mean	5.519630e+05	2139.346957
std	5.638347e+05	963.206916
min	0.000000e+00	370.000000
25%	3.228750e+05	1460.000000
50%	4.609435e+05	1980.000000
75%	6.549625e+05	2620.000000
max	2.659000e+07	13540.000000

In [13]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4600 entries, 0 to 4599
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   cost         4600 non-null   float64
 1   living area  4600 non-null   int64  
dtypes: float64(1), int64(1)
memory usage: 72.0 KB
```

In [14]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\rubin\AppData\Local\Temp\ipykernel_4004\4116506308.py:1: SettingWith
CopyWarning:
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 [15]:
```
x=np.array(df['cost']).reshape(-1,1)
y=np.array(df['living area']).reshape(-1,1)
```

In [16]:
```
df.dropna(inplace=True)
```

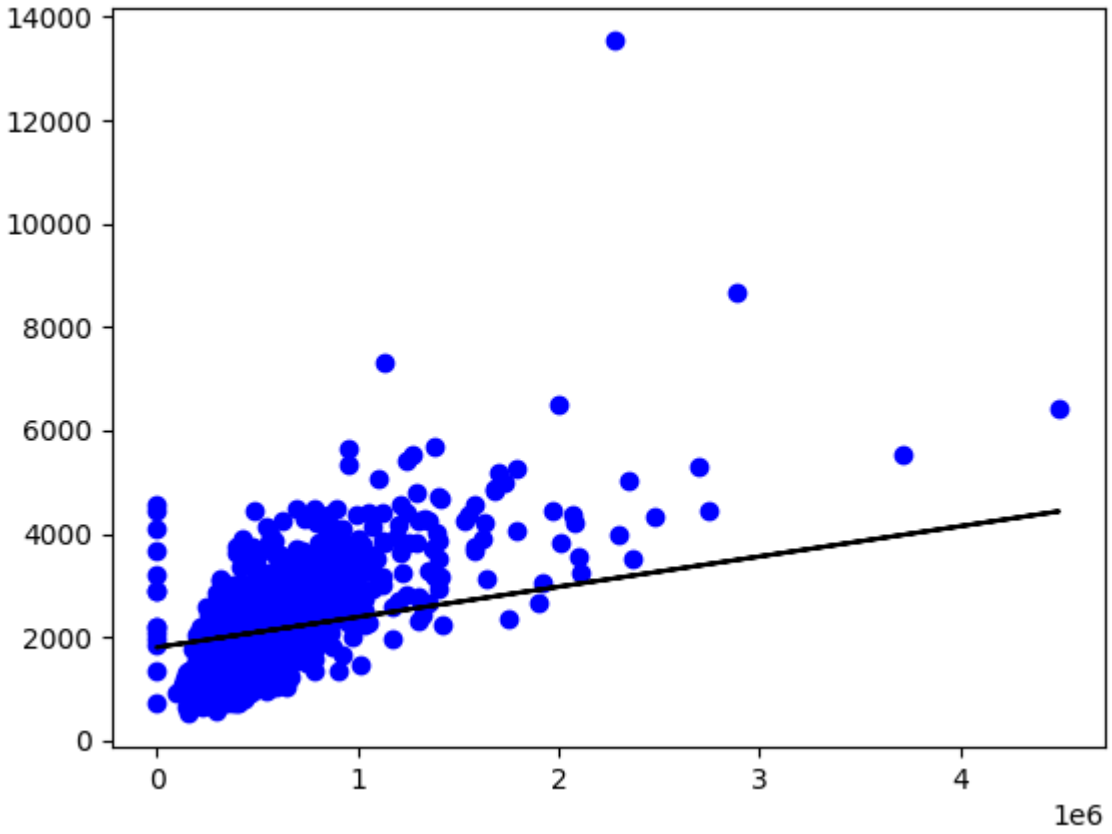
```
C:\Users\rubin\AppData\Local\Temp\ipykernel_4004\1379821321.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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.dropna(inplace=True)
```

In [17]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.27)
```

In [18]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.2501992129189662
```

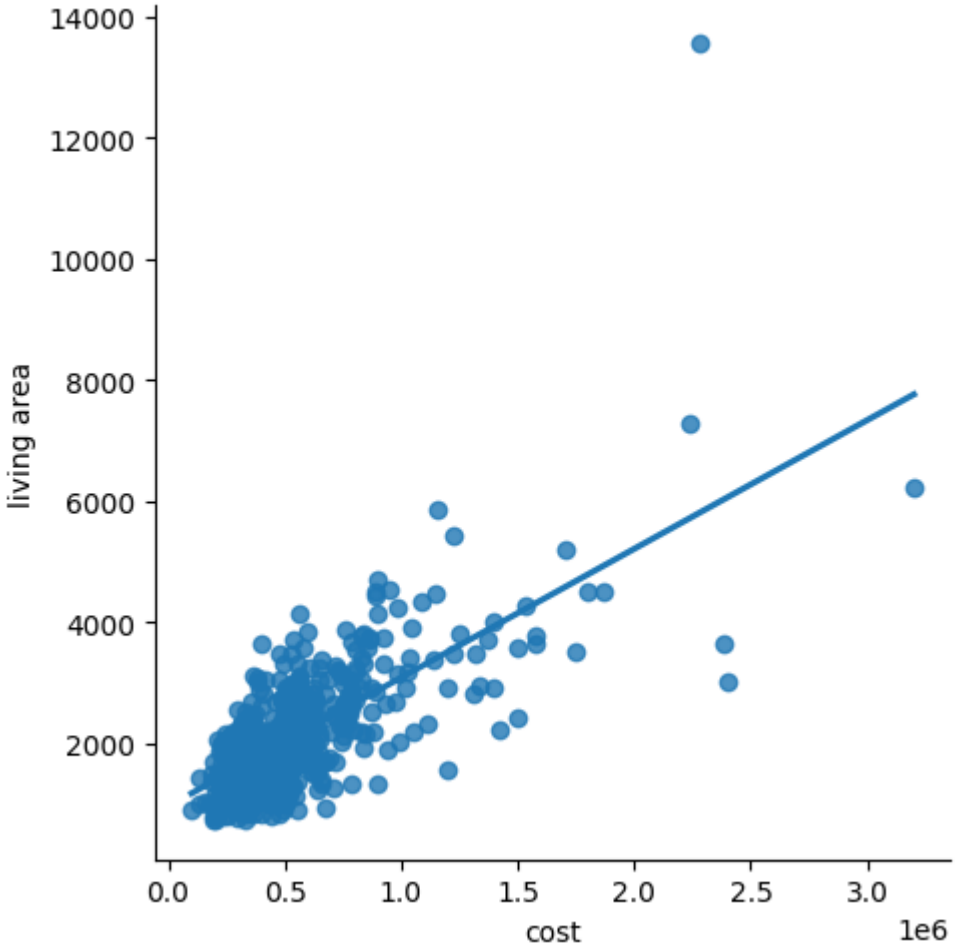
In [19]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



In [20]:
```
df500=df[:][:500]
```

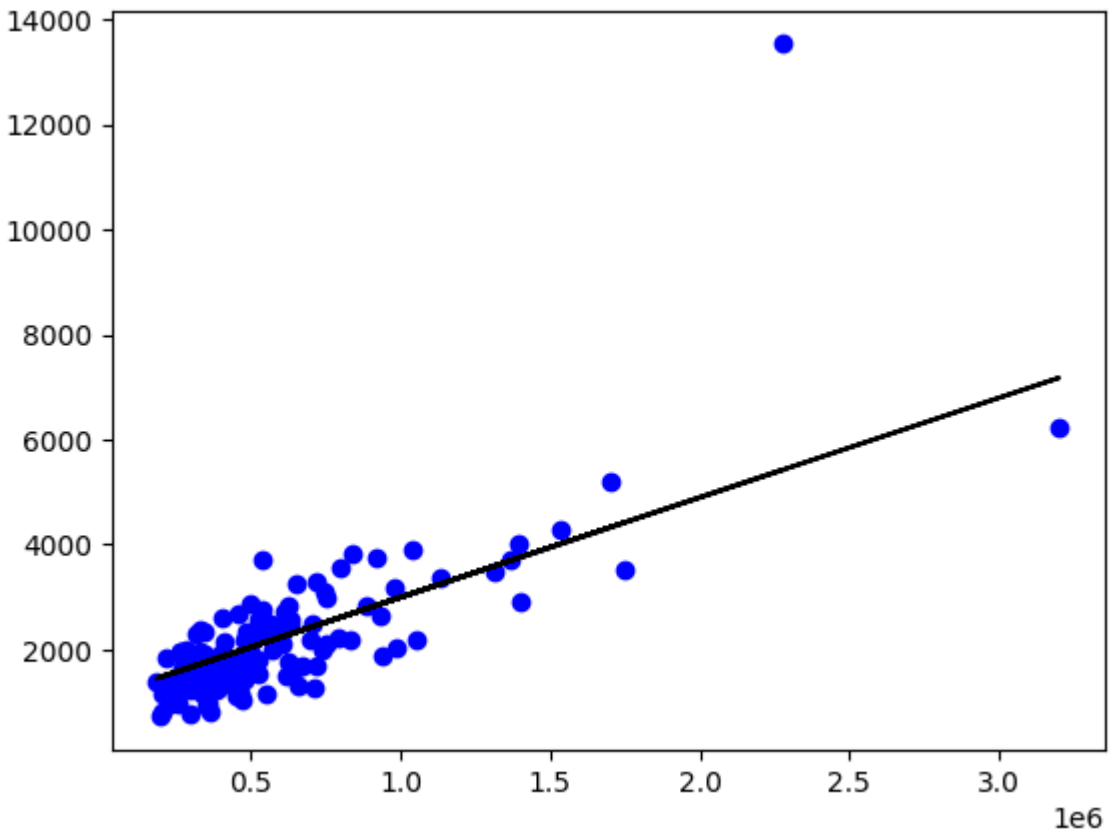
In [21]:
```
sns.lmplot(x="cost",y="living area",data=df500,order=1,ci=None)
```

Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x2648065fc10>
```



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

```
Regresion: 0.5561929794130172
```



In [23]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [25]:
```
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: 0.5561929794130172
```

In [ ]:
```
conclusion:The data set we have taken is average for this model.
```