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 [2]:
```
df=pd.read_csv(r"C:\Users\Dell\Downloads\bottle.csv")
df
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_6052\1212930938.py:1: DtypeWarnin
g: Columns (47,73) have mixed types. Specify dtype option on import or set l
ow_memory=False.
  df=pd.read_csv(r"C:\Users\Dell\Downloads\bottle.csv")
```

Out[2]:

	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	N
1	1	2	054.0 056.0	19-4903CR-HY-060-0930-05400560-0008A-3	8	10.460	33.4400	NaN	25.65600	N
2	1	3	054.0 056.0	19-4903CR-HY-060-0930-05400560-0010A-7	10	10.460	33.4370	NaN	25.65400	N
3	1	4	054.0 056.0	19-4903CR-HY-060-0930-05400560-0019A-3	19	10.450	33.4200	NaN	25.64300	N
4	1	5	054.0 056.0	19-4903CR-HY-060-0930-05400560-0020A-7	20	10.450	33.4210	NaN	25.64300	N
...	...	...	...	...	...	...	...	...	...	
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 [3]:
```
df=df[['Salnty','T_degC']]
df.columns=['Sal','Temp']
```

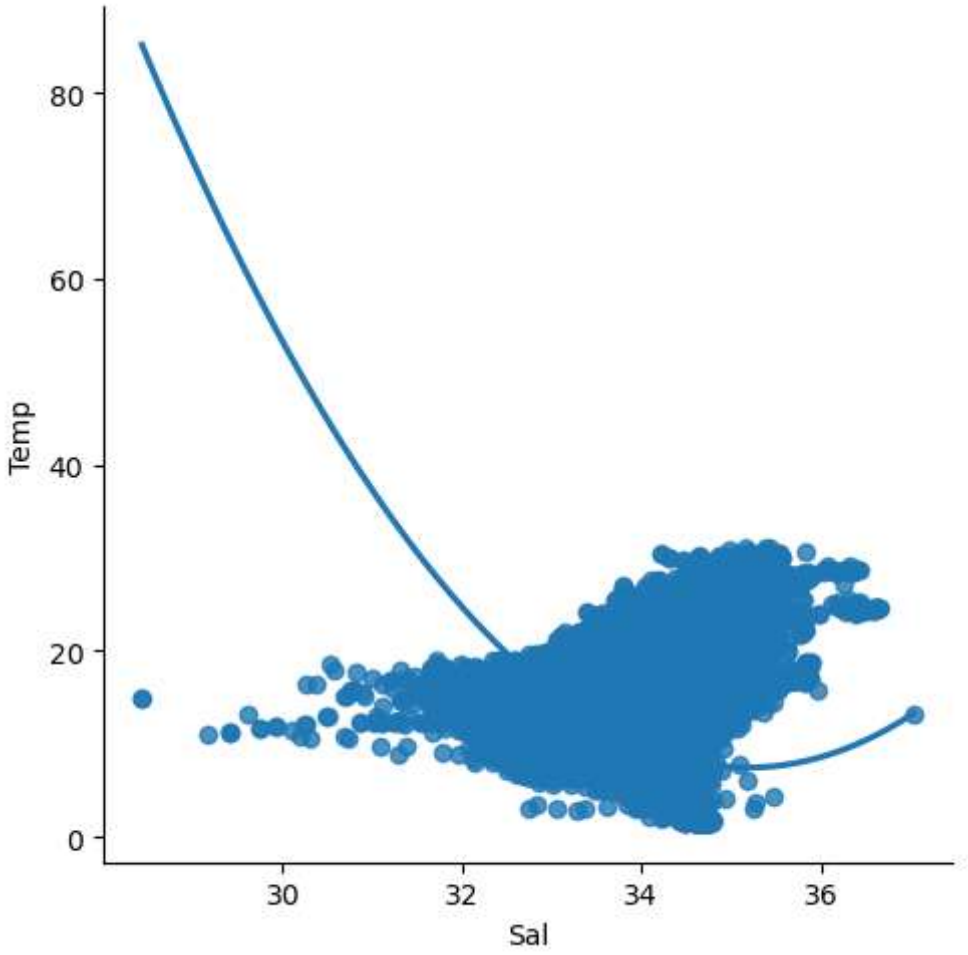
In [4]:
```
df.head(10)
```

Out[4]:

	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 [5]:
```
#Step-3: Exploring the Data Scatter - plotting the data scatter
sns.lmplot(x="Sal",y="Temp", data = df, order = 2, ci = None)
```

Out[5]: <seaborn.axisgrid.FacetGrid at 0x1b8b6f9df00>



In [6]:
```
df.describe()
```

Out[6]:

	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 [8]:
```
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     864863 non-null  float64
 1   Temp    864863 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [10]:
```
#Step-4: Data cleaning - Eliminating NaN OR missing input numbers
df.fillna(method ='ffill', inplace = True)
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_6052\3221840372.py:2: 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/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.fillna(method ='ffill', inplace = True)
```

In [11]:
```
# Step-5: Training Our Model
x = np.array(df['Sal']).reshape(-1, 1)
y = np.array(df['Temp']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

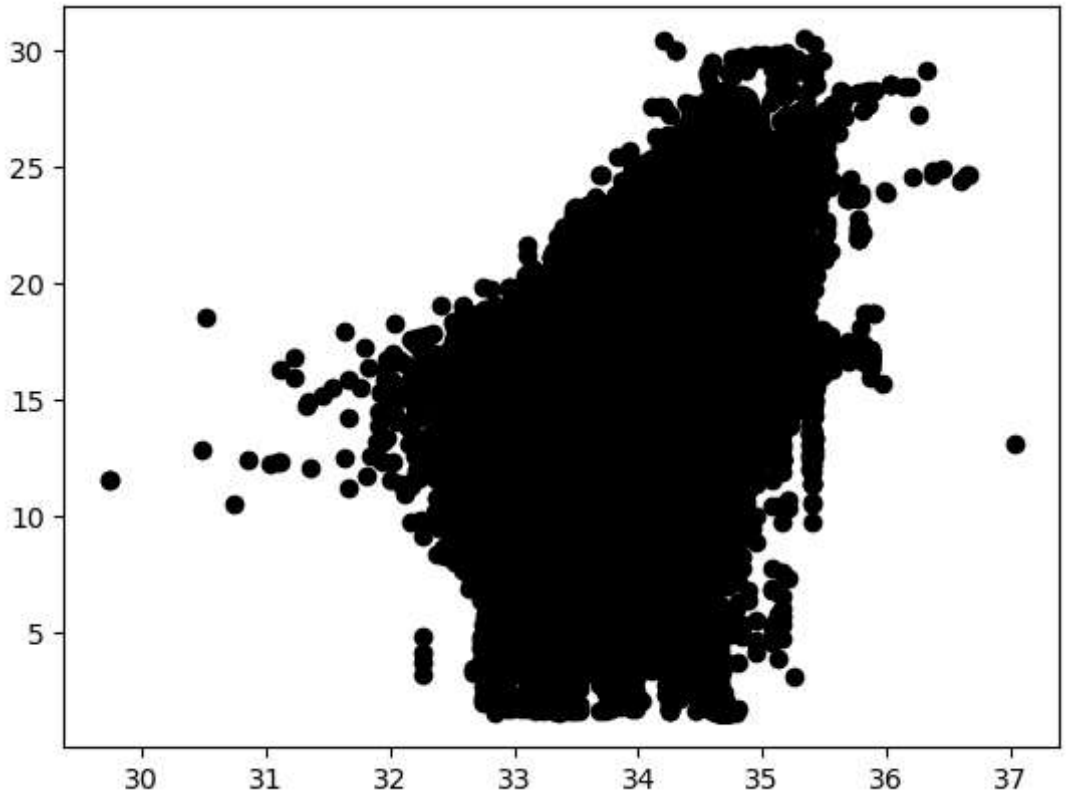
In [12]:
```
df.dropna(inplace = True)
```

```
C:\Users\Dell\AppData\Local\Temp\ipykernel_6052\1791587065.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/
stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.dropna(inplace = True)
```

In [14]:
```
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))
```

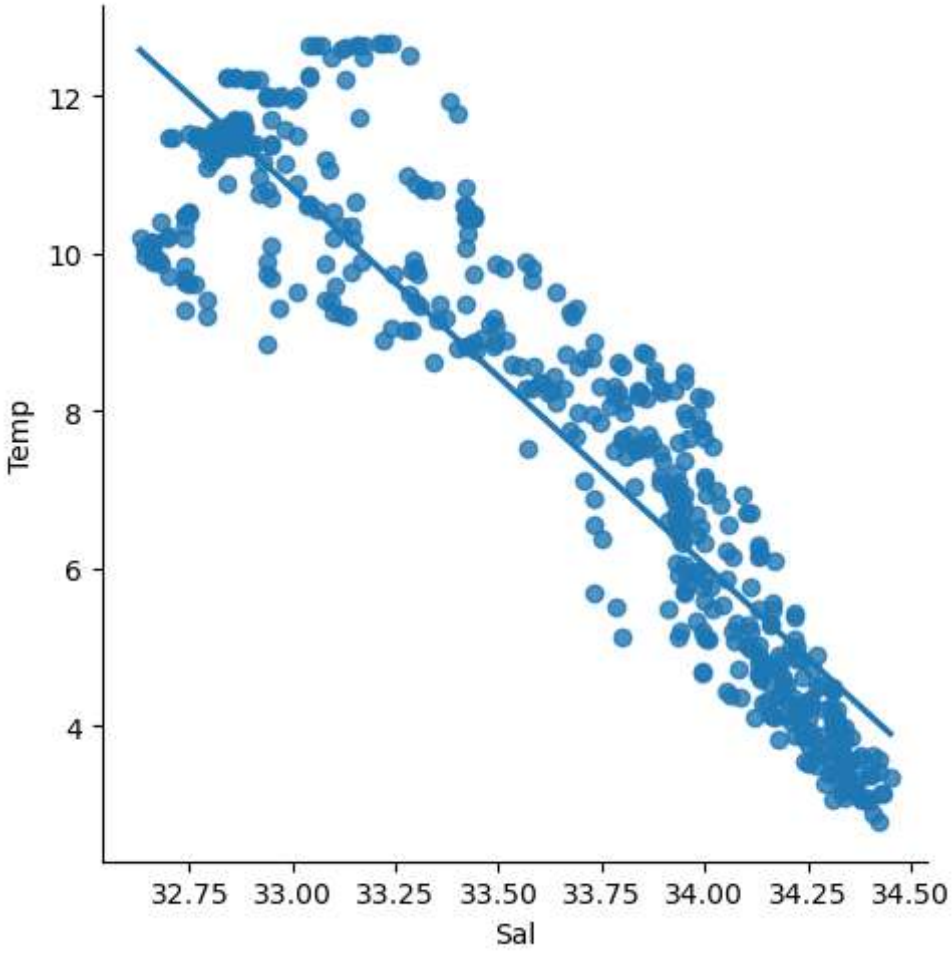
```
0.20493671607123287
```

In [16]:
```
#step-6: Exploring Our Results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'k')
plt.show()
# Data scatter of predicted values
```



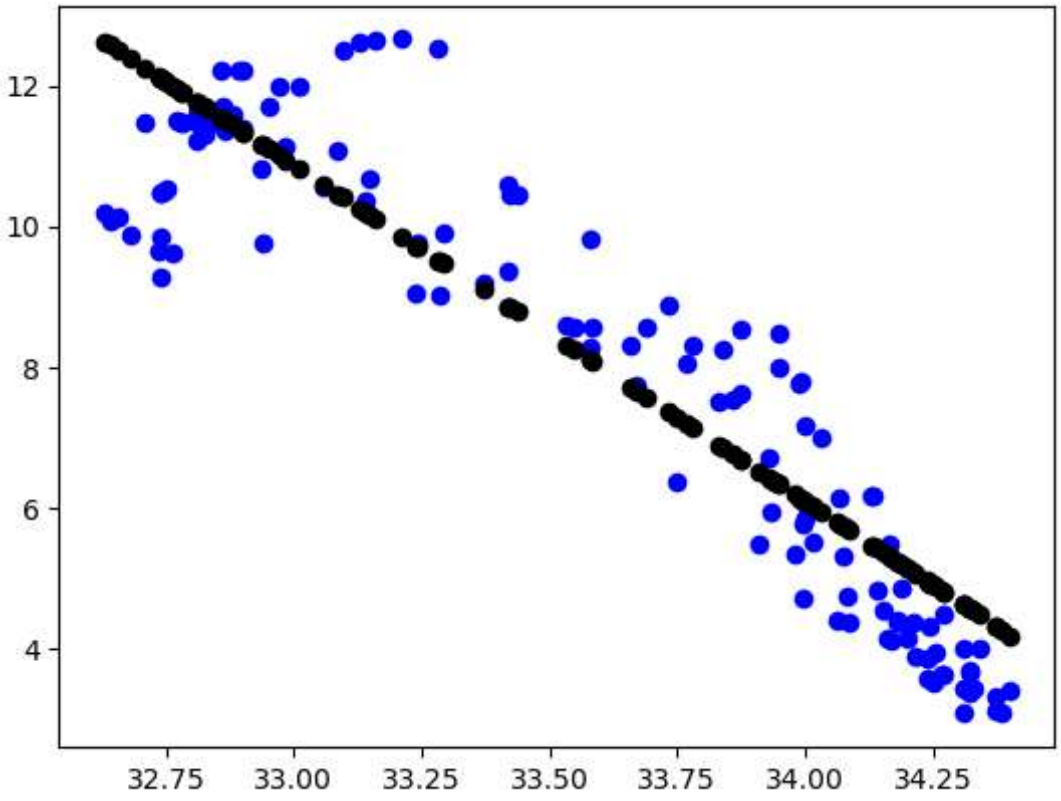
In [18]:
```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x="Sal", y ="Temp", data = df500, order = 1, ci = None)
```

Out[18]: <seaborn.axisgrid.FacetGrid at 0x1b8be975cf0>



In [19]:
```
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("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: 0.8416963974042857
```



In [20]:
```
#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: 0.8416963974042857
```

Step-9:Conclusion: Dataset we have taken is poor for Linear Model,but with the smaller data works well with Linear Model.

In [47]:
```
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 [48]:
```
df=pd.read_csv(r"C:\Users\Dell\Downloads\fiat500_VehicleSelection_Dataset.csv'
df
```

Out[48]:

	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 [49]:
```
df=df[['age_in_days','km']]
df.columns=['age','km']
```

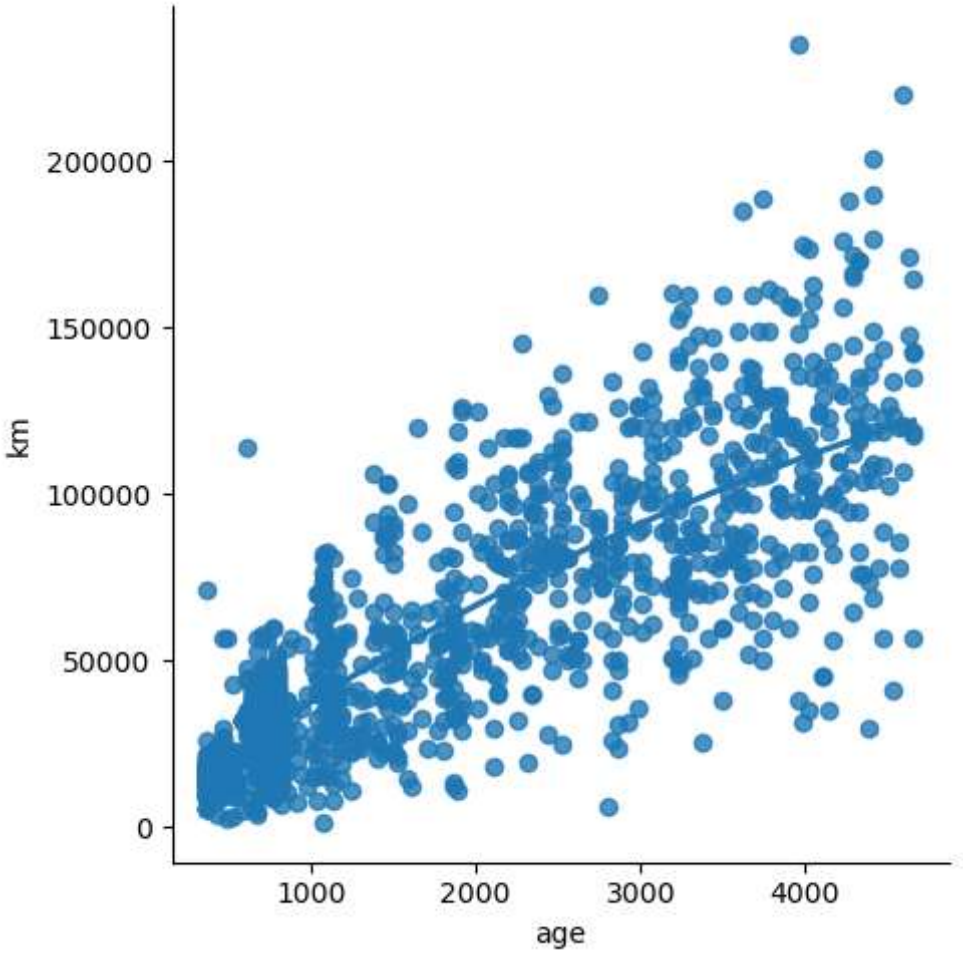
In [50]:
```
df.head(10)
```

Out[50]:

	age	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 [52]:
```
sns.lmplot(x="age",y="km",data=df,order=2,ci=None)
```

Out[52]: <seaborn.axisgrid.FacetGrid at 0x1b8e5052230>



In [53]:
```
df.describe()
```

Out[53]:

	age	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 [54]:
```
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   km      1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [55]:
```
df.fillna(method='ffill',inplace=True)
```

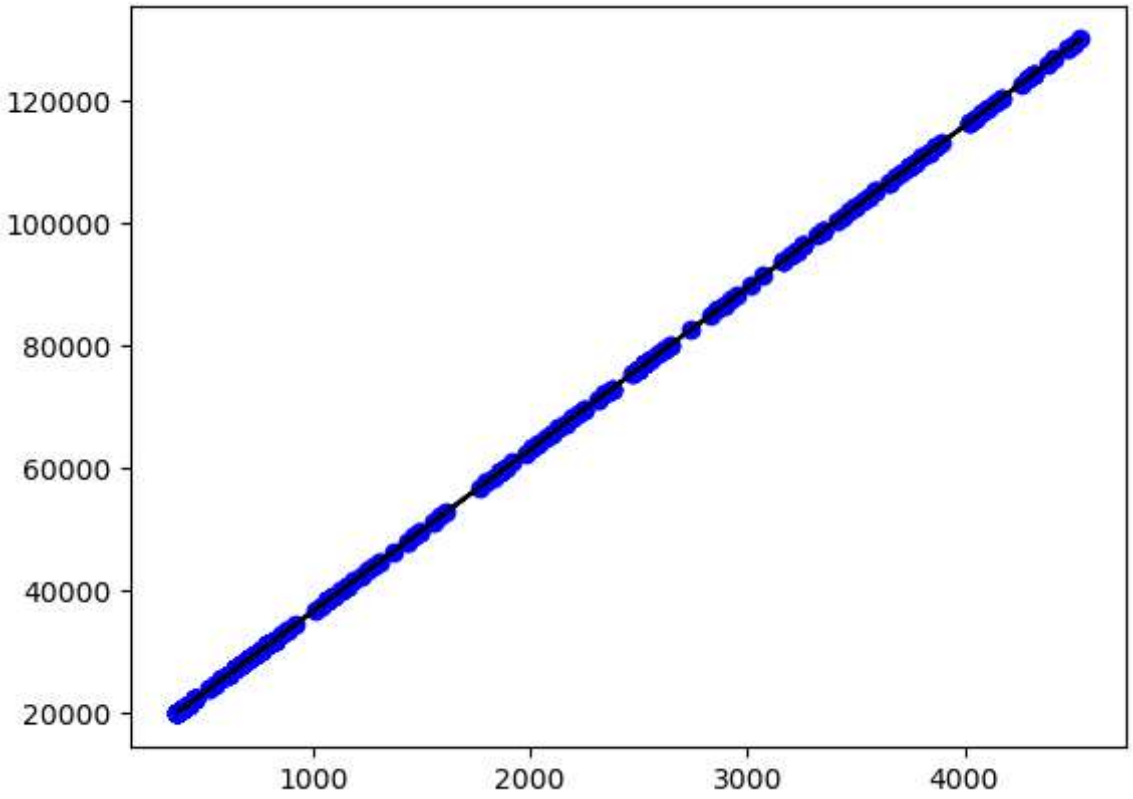
```
C:\Users\Dell\AppData\Local\Temp\ipykernel_6052\4116506308.py:1: SettingWith
CopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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

In [56]:
```
x=np.array(df['age']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
```

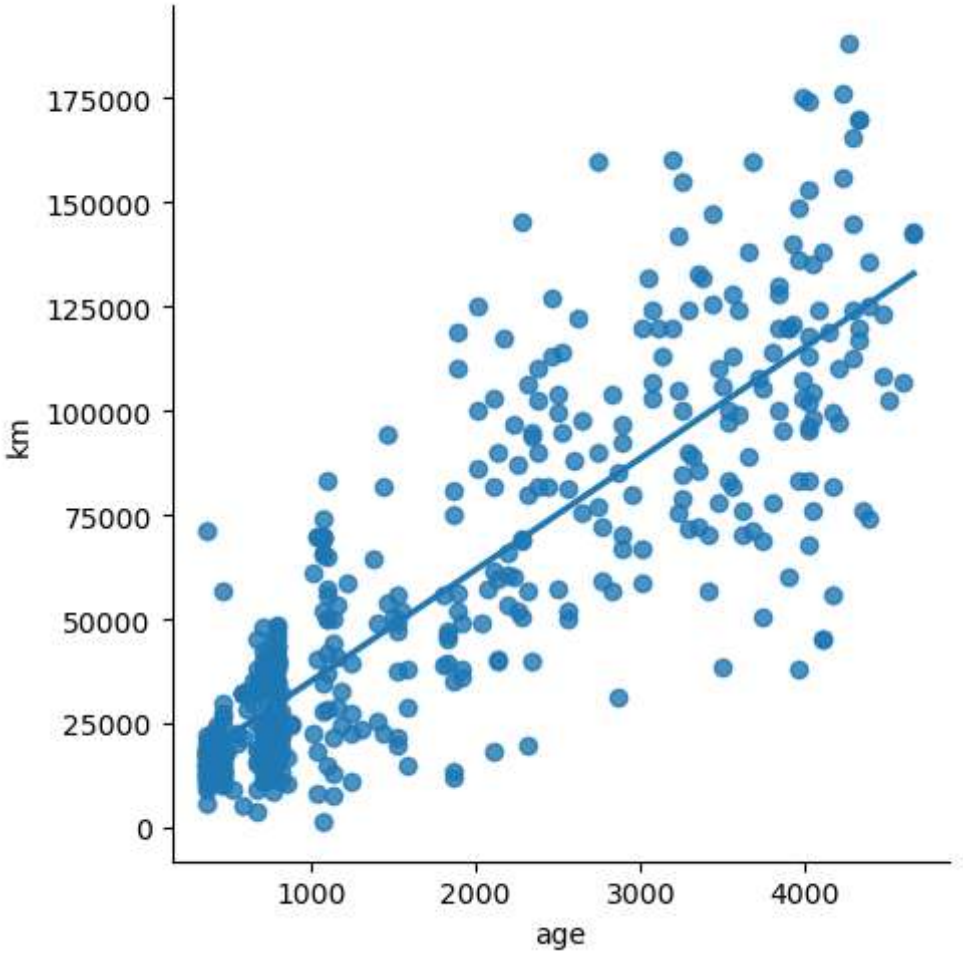
In [57]:
```
df.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.6800513417376057

C:\Users\Dell\AppData\Local\Temp\ipykernel_6052\693062840.py:1: SettingWithC
opyWarning:
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://pand
as.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-v
ersus-a-copy)
  df.dropna(inplace=True)
```

In [58]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_pred,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



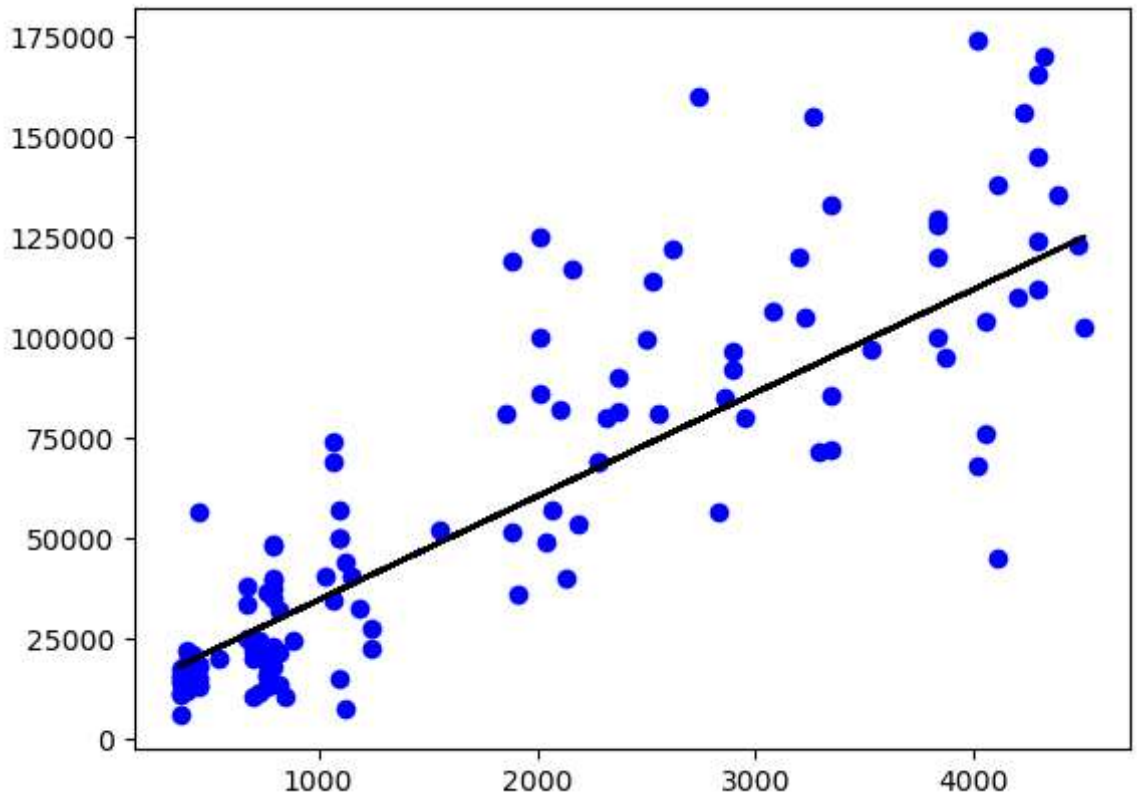
In [59]:
```
df500=df[:][:500]
sns.lmplot(x="age",y="km",data=df500,order=1,ci=None)
x=np.array(df500['age']).reshape(-1,1)
y=np.array(df500['km']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```



In [60]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: 0.73459817118305
```

In [61]:
```
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 [62]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing,svm
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

In [65]:
```
df=pd.read_csv(r"C:\Users\Dell\Downloads\Housing.csv")
df
```

Out[65]:

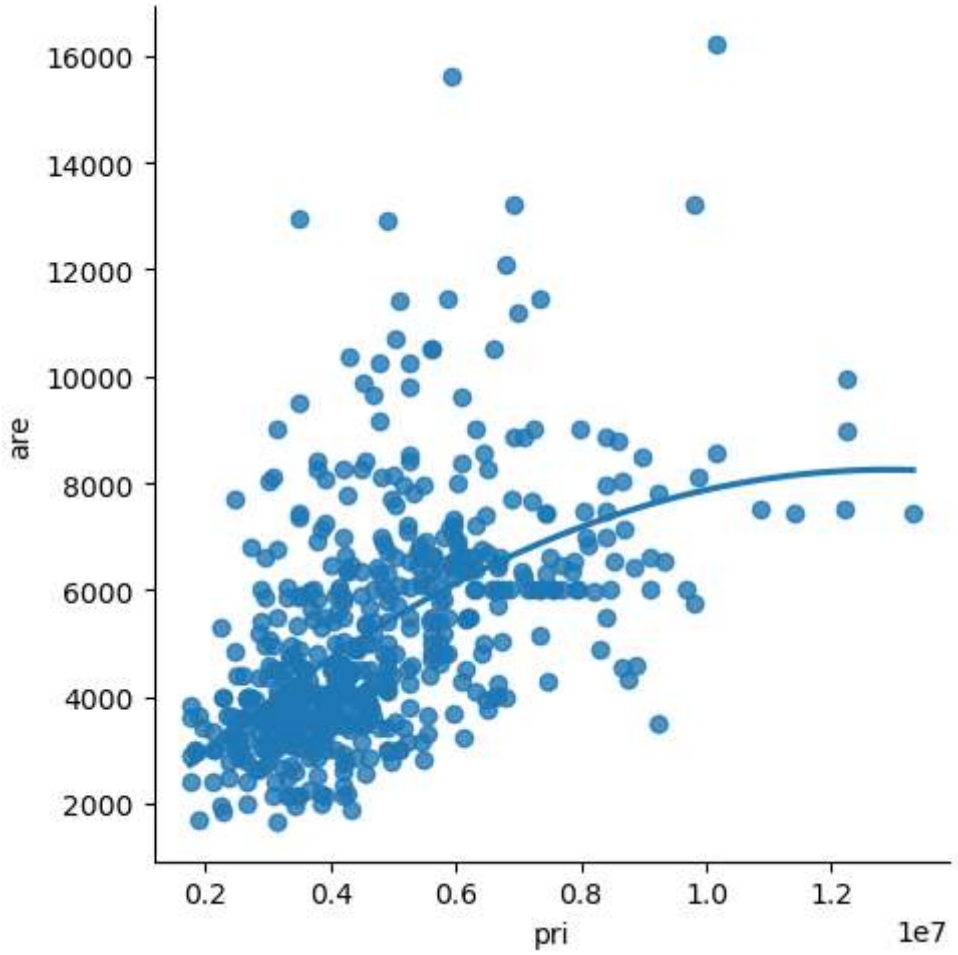
	price	area	bedrooms	bathrooms	stories	mainroad	guestroom	basement	hotwaterl
0	13300000	7420	4	2	3	yes	no	no	
1	12250000	8960	4	4	4	yes	no	no	
2	12250000	9960	3	2	2	yes	no	yes	
3	12215000	7500	4	2	2	yes	no	yes	
4	11410000	7420	4	1	2	yes	yes	yes	
...	...	...	...	...	...	...	...	...	
540	1820000	3000	2	1	1	yes	no	yes	
541	1767150	2400	3	1	1	no	no	no	
542	1750000	3620	2	1	1	yes	no	no	
543	1750000	2910	3	1	1	no	no	no	
544	1750000	3850	3	1	2	yes	no	no	

545 rows × 13 columns

In [68]:
```
df=df[['price','area']]
df.columns=['pri','are']
```

In [69]:
```
sns.lmplot(x='pri',y='are',data=df,order=2,ci=None)
```

Out[69]:
```
<seaborn.axisgrid.FacetGrid at 0x1b8e8d463b0>
```



In [70]:
```
df.head(10)
```

Out[70]:

	pri	are
0	13300000	7420
1	12250000	8960
2	12250000	9960
3	12215000	7500
4	11410000	7420
5	10850000	7500
6	10150000	8580
7	10150000	16200
8	9870000	8100
9	9800000	5750

In [71]:
```
df.describe()
```

Out[71]:

	pri	are
count	5.450000e+02	545.000000
mean	4.766729e+06	5150.541284
std	1.870440e+06	2170.141023
min	1.750000e+06	1650.000000
25%	3.430000e+06	3600.000000
50%	4.340000e+06	4600.000000
75%	5.740000e+06	6360.000000
max	1.330000e+07	16200.000000

In [72]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 545 entries, 0 to 544
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   pri     545 non-null    int64
 1   are     545 non-null    int64
dtypes: int64(2)
memory usage: 8.6 KB
```

In [73]:
```
df.fillna(method='ffill',inplace=True)
```

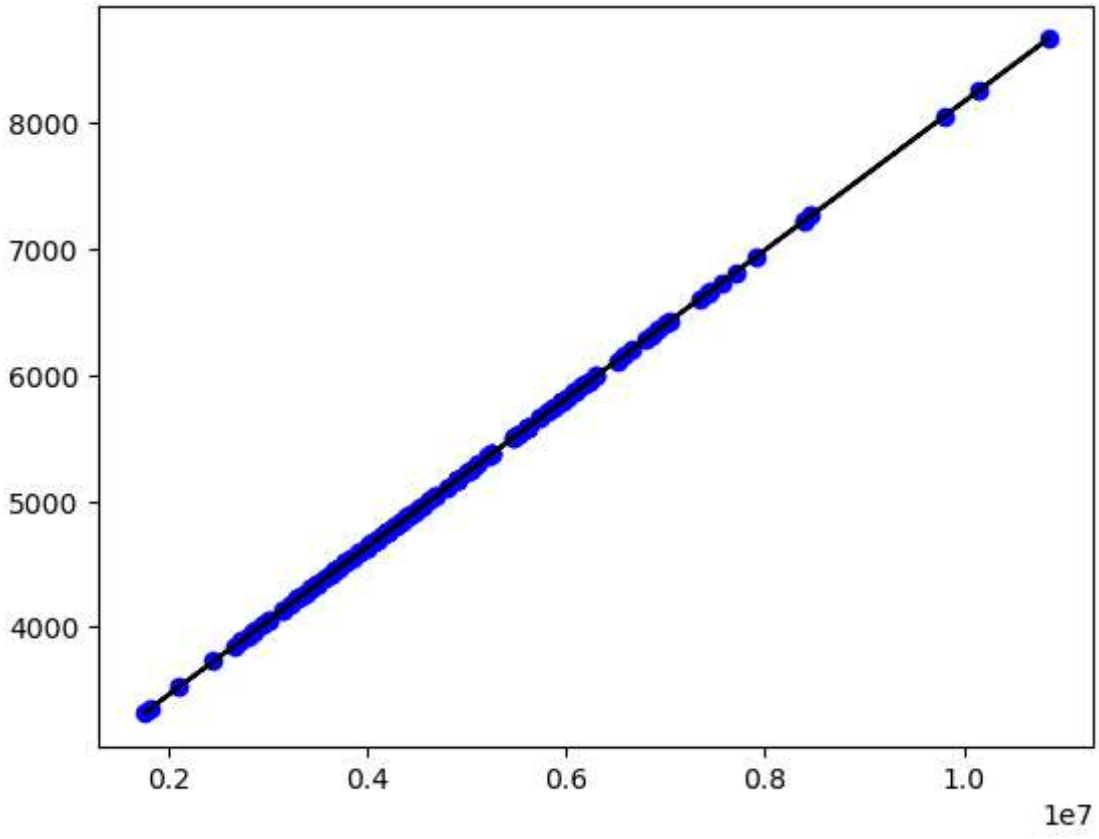
In [74]:
```
x=np.array(df['pri']).reshape(-1,1)
y=np.array(df['are']).reshape(-1,1)
```

In [76]:
```
df.dropna(inplace=True)
```

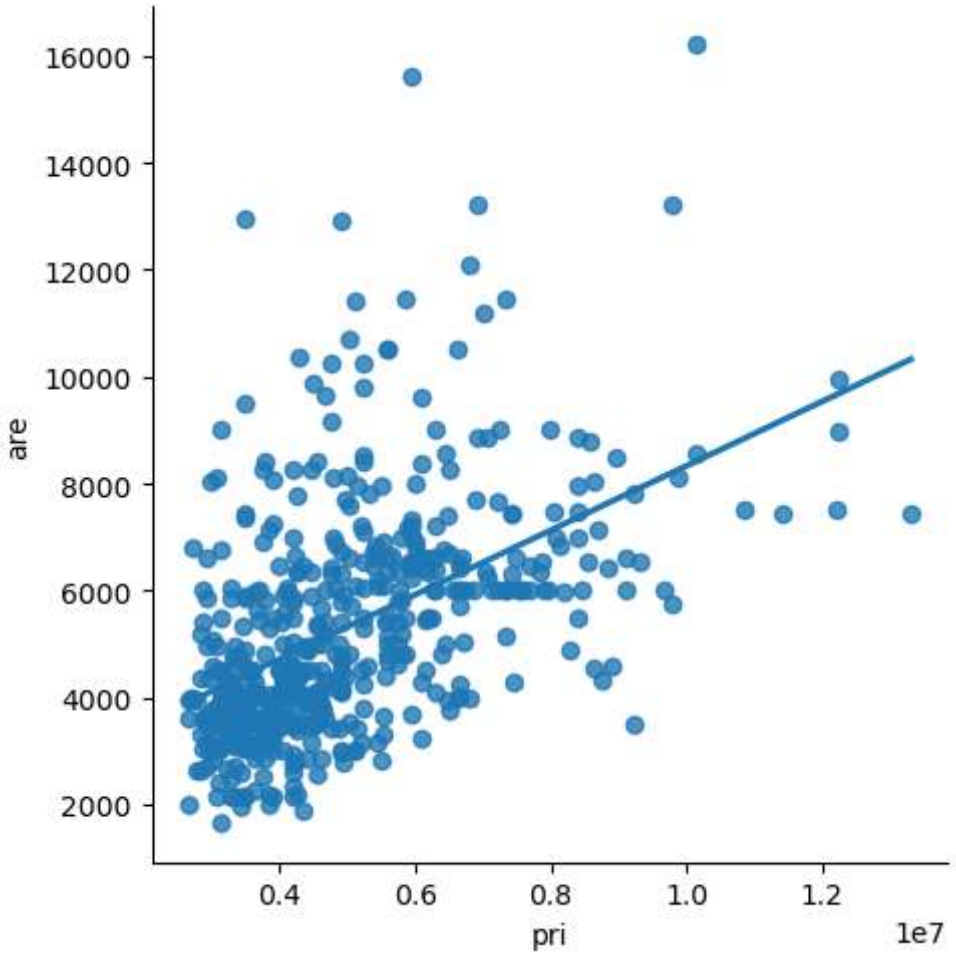
In [77]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.27990190403880166
```

In [78]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_pred,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



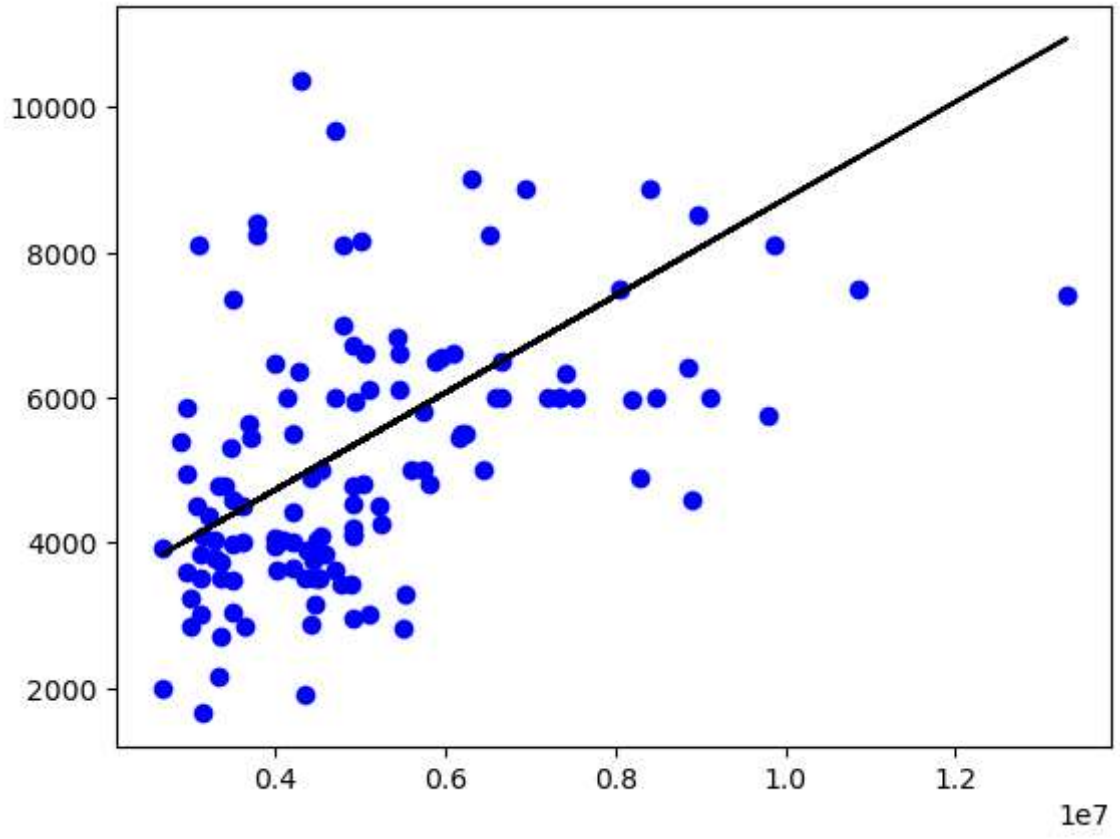
In [79]:
```
df500=df[:][:500]
sns.lmplot(x="pri",y="are",data=df500,order=1,ci=None)
x=np.array(df500['pri']).reshape(-1,1)
y=np.array(df500['are']).reshape(-1,1)
df500.dropna(inplace=True)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
```



In [80]:
```
regr=LinearRegression()
regr.fit(x_train,y_train)
print("Regression:",regr.score(x_test,y_test))
```

```
Regression: 0.13520660954672648
```

In [81]:
```
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 [ ]: