In [2]:

```
# importing all the 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
```

Bottle Dataset

(linear regression model)

```
In [3]: # reading the file
        df=pd.read_csv(r"C:\Users\91756\Downloads\bottle.csv.zip")
        df
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_28160\2818008035.py:2: DtypeWarn
ing: Columns (47,73) have mixed types. Specify dtype option on import or set
low_memory=False.
  df=pd.read_csv(r"C:\Users\91756\Downloads\bottle.csv.zip")
```

Out[3]:

	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

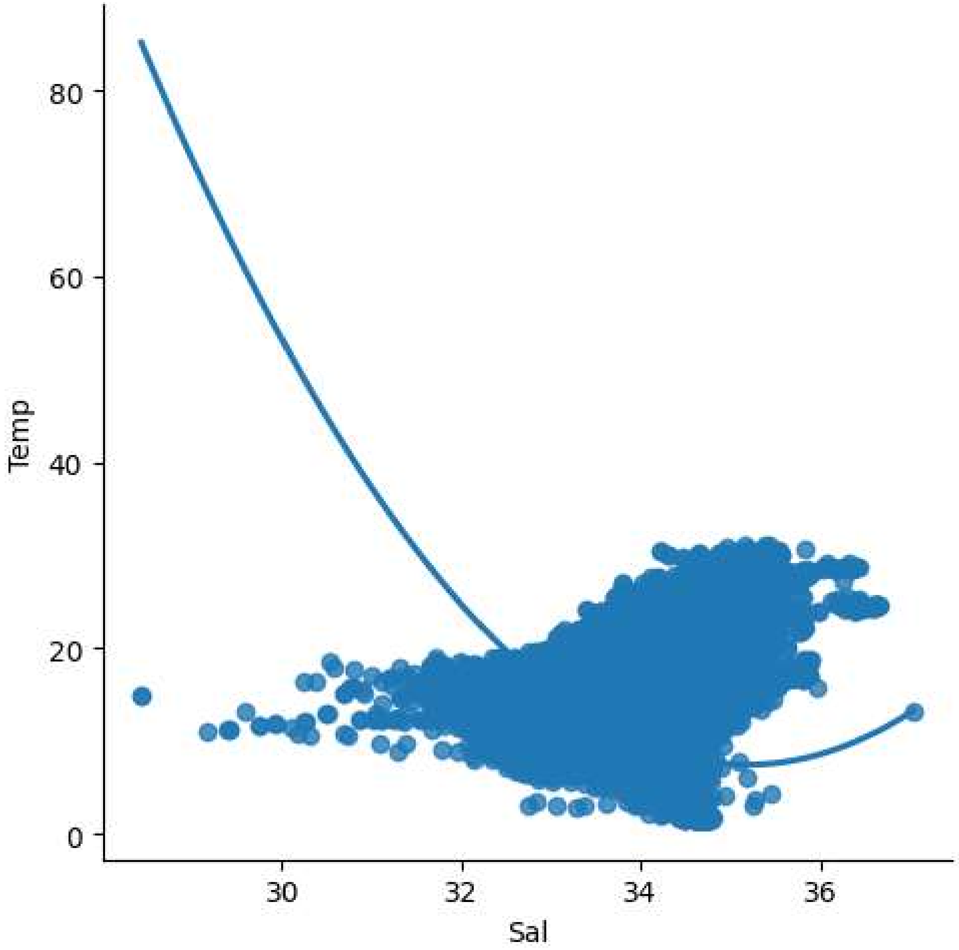
	Cst_Cnt	Btl_Cnt	Sta_ID	Depth_ID	Depthm	T_degC	Salnty	O2ml_L	STheta	O2S
864862	34404	864863	093.4 026.4	20-1611SR-MX-310-2239-09340264-0015A-3	15	17.533	33.3880	5.774	24.15297	105

864863 rows × 74 columns

```
In [5]: df = df[['Salnty', 'T_degC']]
        df.columns=['Sal','Temp']
```

```
In [6]: # step 3: Exploring the data scatter _plotting the data scatter
        sns.lmplot(x="Sal", y="Temp", data=df, order=2, ci= None)
```

```
Out[6]: <seaborn.axisgrid.FacetGrid at 0x228b5e703a0>
```



In [7]:
```
df.describe()
```

Out[7]:

	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     817509 non-null  float64
 1   Temp    853900 non-null  float64
dtypes: float64(2)
memory usage: 13.2 MB
```

In [9]:
```
# step-4: Data cleaning- Eliminating NaN or missing input numbers
df.fillna(method='ffill', inplace=True)
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_28160\1327383682.py:2: SettingWi
thCopyWarning:
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 [10]:
```
# step-5: Training our Model
X = np.array(df['Sal']).reshape(-1,1)
y = np.array(df['Temp']).reshape(-1,1)
# Separating the data into independent and dependent variables and converting
# Now each dataframe contains only one column
```

In [11]:

```
df.dropna(inplace = True)
# Dropping any rows with Nan values
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_28160\3378209027.py:1: SettingWi
thCopyWarning:
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)
```

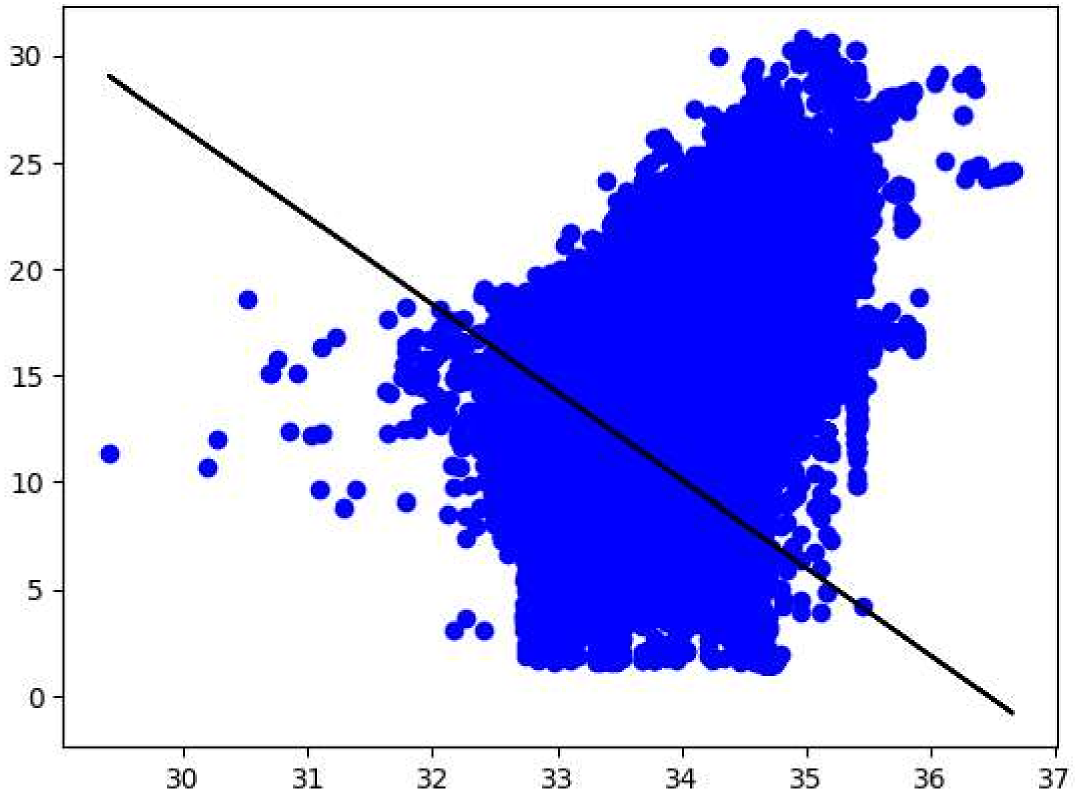
In [12]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.25)
# Splitting the data into training and testing data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.20148413040458024
```

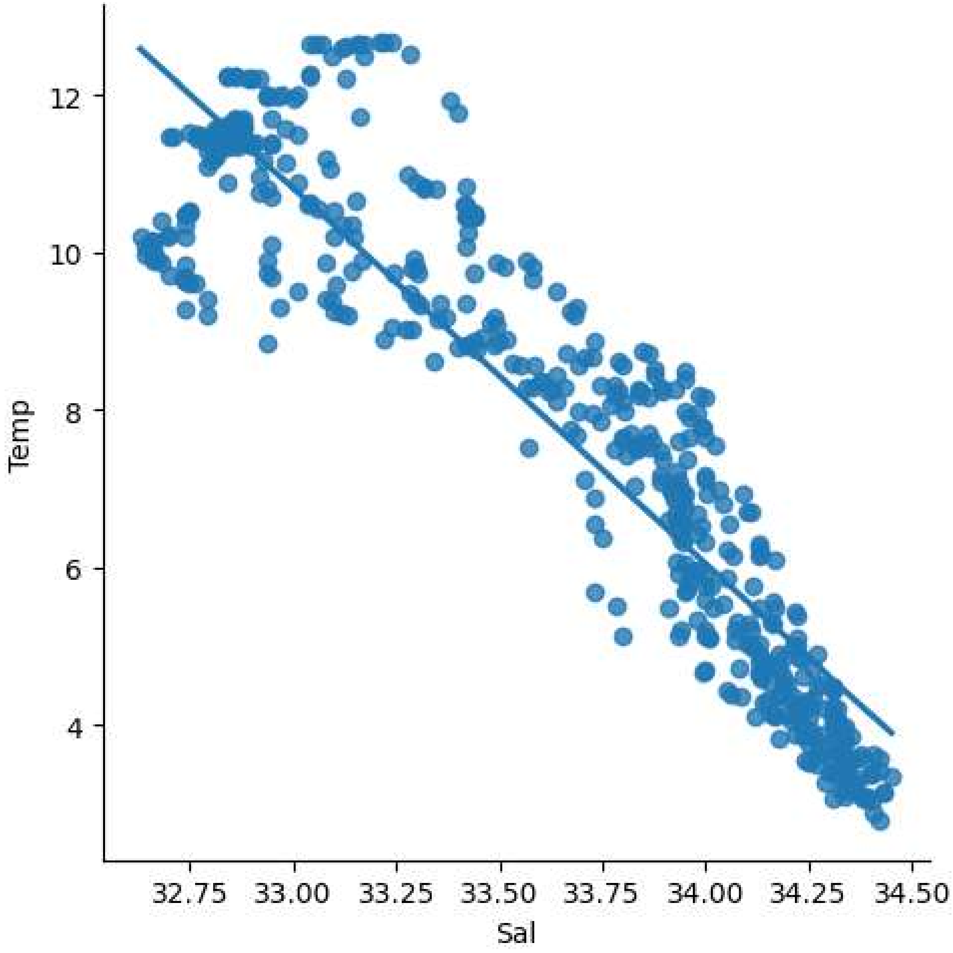
In [13]:

```
# 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 [14]:
```
# 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)
```

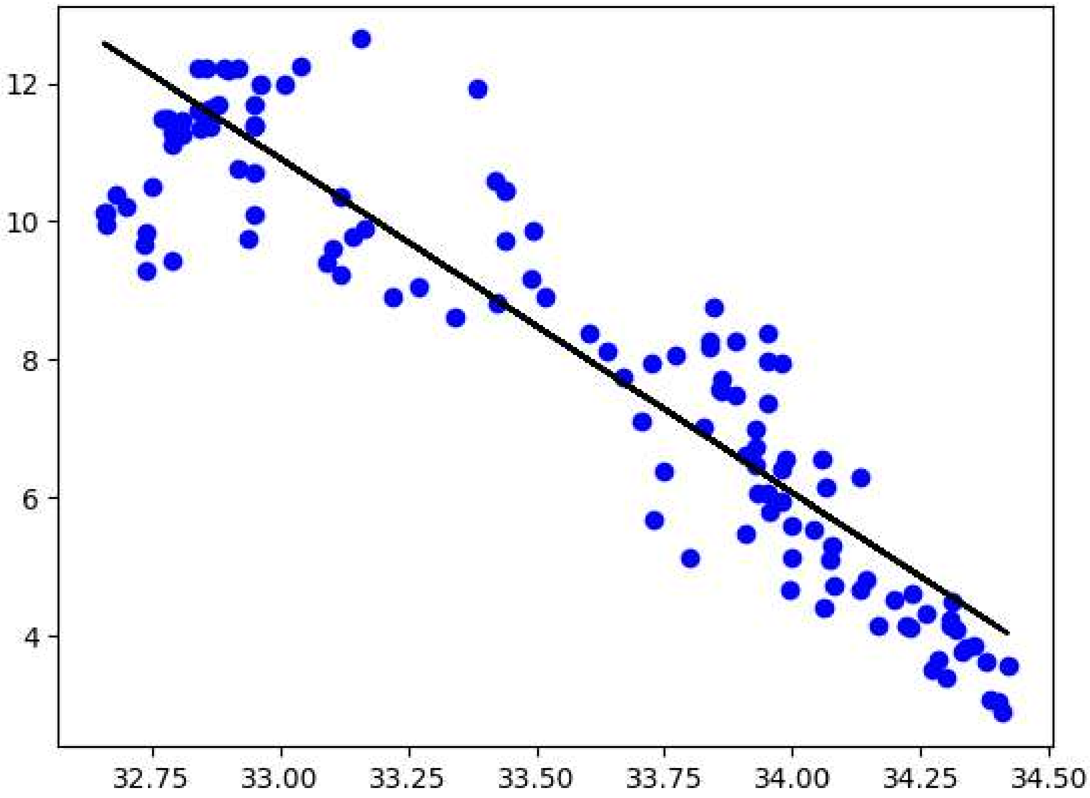
Out[14]:
```
<seaborn.axisgrid.FacetGrid at 0x228b6768130>
```



In [15]:

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

```
Regression:  0.8408670535898797
```



In [31]:

```
# 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)

y_pred=model.predict(X_test)

r2=r2_score(y_test,y_pred)

print("R2 score: ",r2)
# Evaluate the model on the test set
```

```
R2 score:  0.04327592860944873
```

In [17]:

```
# importing all the 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 [18]:

```
df=pd.read_csv(r"C:\Users\91756\Documents\python\fiat500_VehicleSelection_Data
df
```

Out[18]:

	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

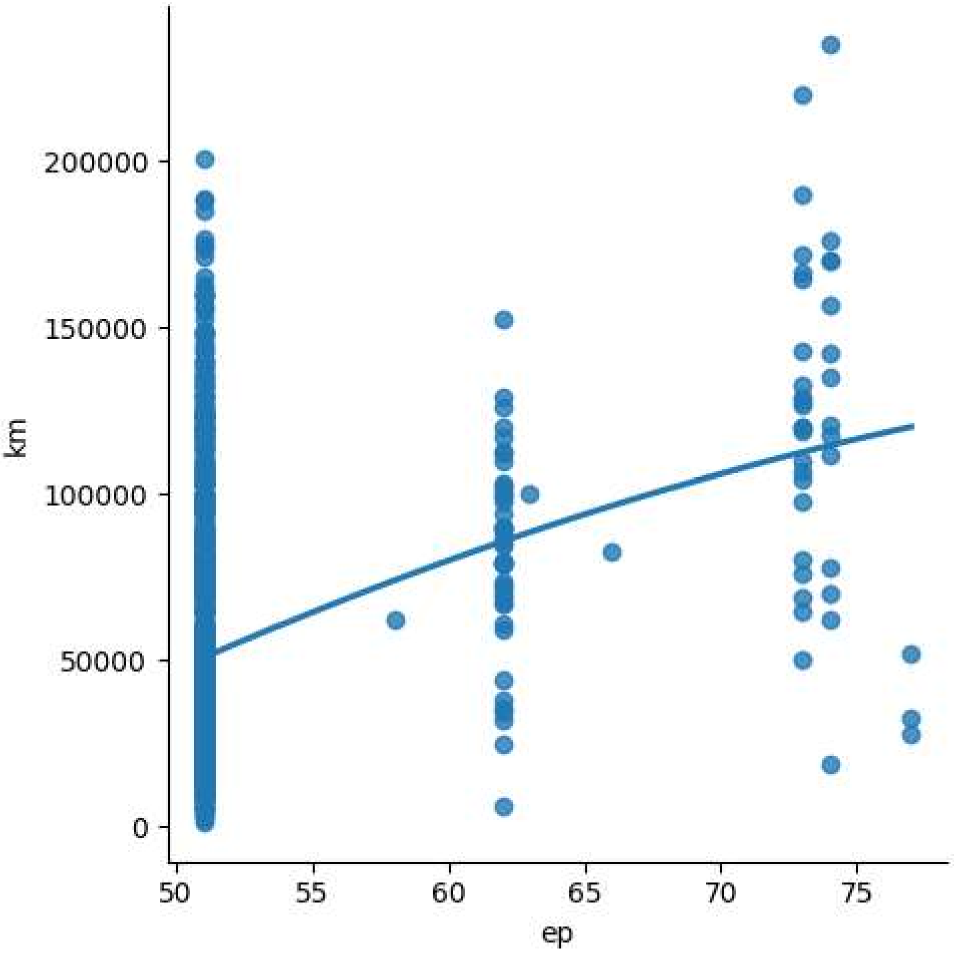
1538 rows × 9 columns

In [19]:

```
df = df[['engine_power', 'km']]
df.columns=['ep','km']
```

In [20]:
```
sns.lmplot(x="ep", y="km", data=df, order=2, ci= None)
```

Out[20]:
```
<seaborn.axisgrid.FacetGrid at 0x228b6456680>
```



In [21]:
```
df.describe()
```

Out[21]:

	ep	km
count	1538.000000	1538.000000
mean	51.904421	53396.011704
std	3.988023	40046.830723
min	51.000000	1232.000000
25%	51.000000	20006.250000
50%	51.000000	39031.000000
75%	51.000000	79667.750000
max	77.000000	235000.000000

In [22]:

```
df.info()
```

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

In [23]:

```
df.fillna(method='ffill', inplace=True)
```

```
C:\Users\91756\AppData\Local\Temp\ipykernel_28160\3970806690.py:1: SettingWi
thCopyWarning:
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 [24]:

```
X = np.array(df['ep']).reshape(-1,1)
y = np.array(df['km']).reshape(-1,1)
```

In [25]:

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

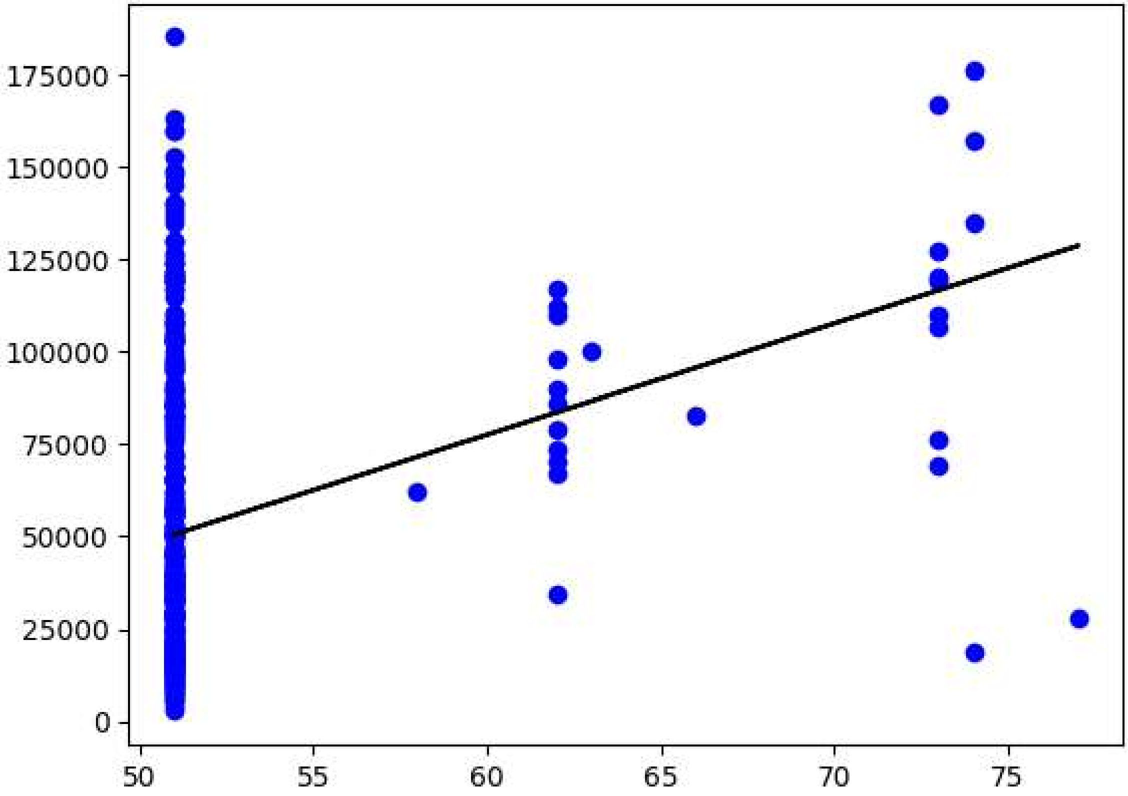
```
C:\Users\91756\AppData\Local\Temp\ipykernel_28160\1791587065.py:1: SettingWi
thCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

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 [26]:

```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.25)
# Splitting the data into training and testing data
regr = LinearRegression()
regr.fit(X_train, y_train)
print(regr.score(X_test, y_test))
```

```
0.08434610179886837
```

```
In [27]: y_pred = regr.predict(X_test)
         plt.scatter(X_test, y_test, color ='b')
         plt.plot(X_test, y_pred, color ='k')
         plt.show()
```

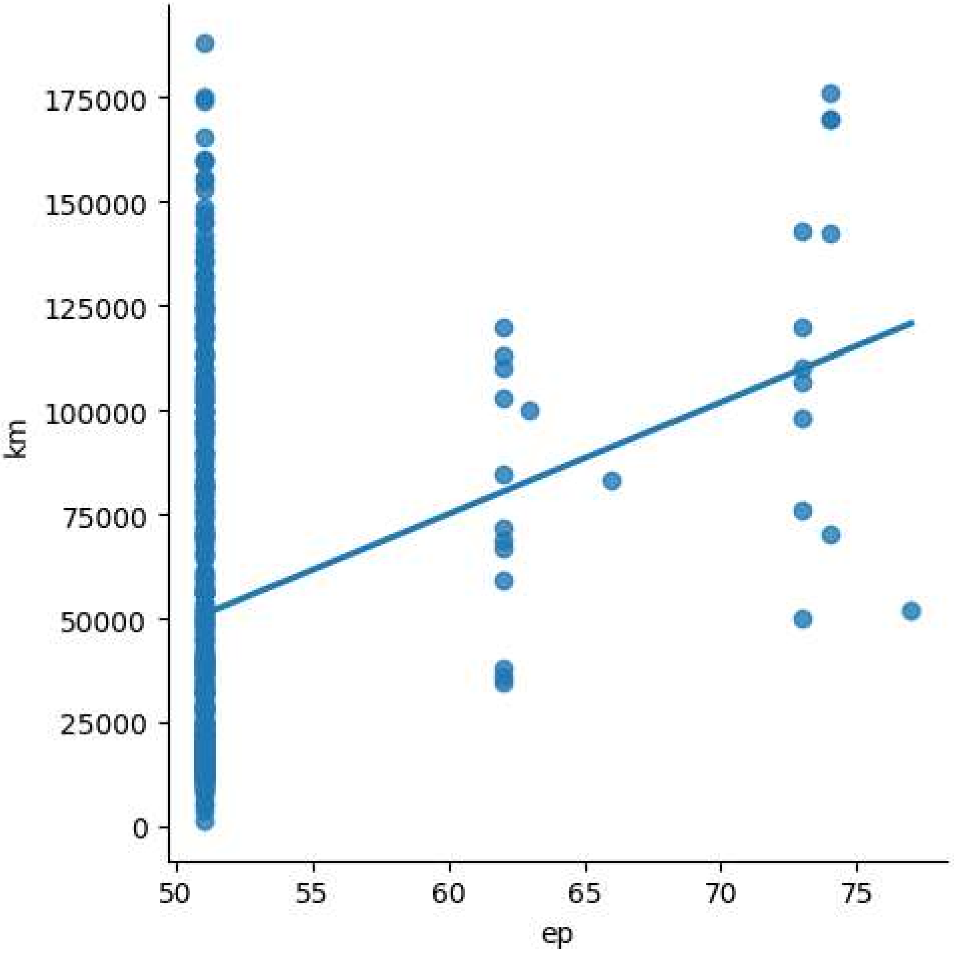


In [28]:

```
df500 = df[:][:500]
# selecting the 1st 500 rows of the data
sns.lmplot(x = "ep", y ="km", data = df500, order = 1, ci = None)
```

Out[28]:

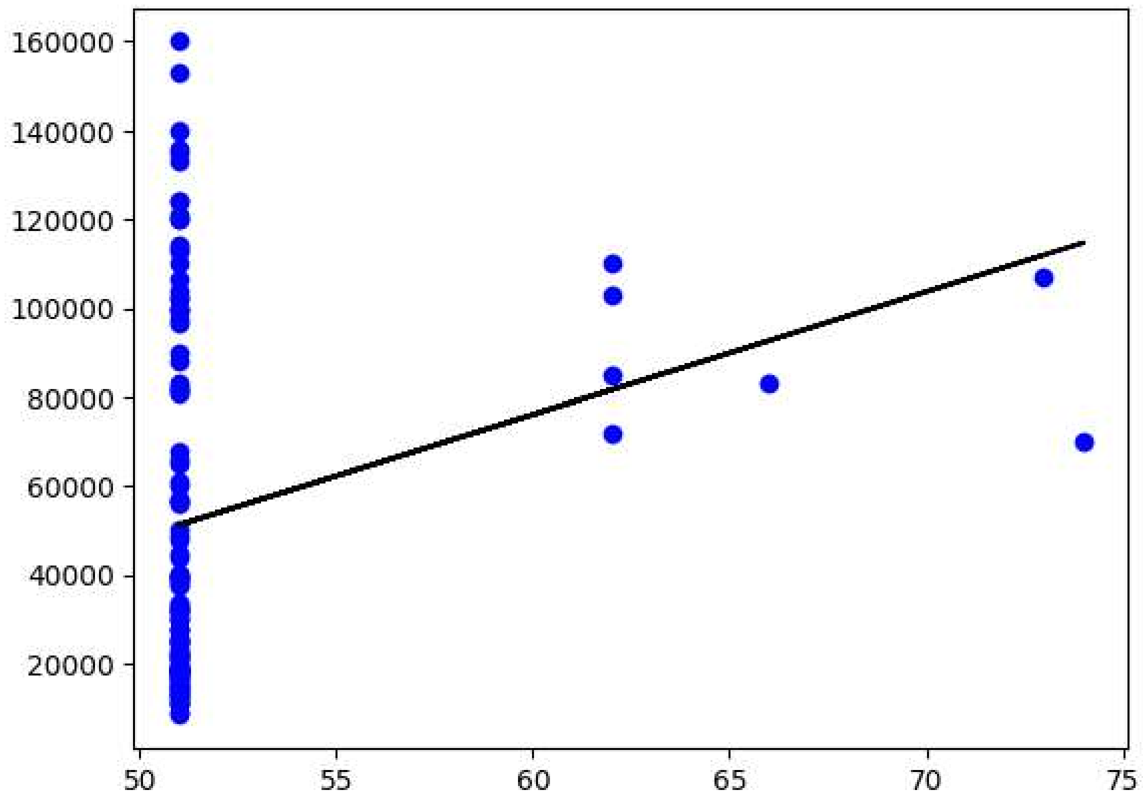
```
<seaborn.axisgrid.FacetGrid at 0x228b6769870>
```



In [29]:

```
df500.fillna(method ='ffill', inplace = True)
X = np.array(df500['ep']).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)
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:  0.04327592860944873
```



In [30]:

```
# 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)
y_pred=model.predict(X_test)
r2=r2_score(y_test,y_pred)
print("R2 score: ",r2)
# Evaluate the model on the test set
```

```
R2 score:  0.04327592860944873
```

In [ ]: