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\chila\Downloads\fiat500_VehicleSelection_Dataset.csv")
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.611
1	2	pop	51	1186	32500	1	45.666359	12.241
2	3	sport	74	4658	142228	1	45.503300	11.417
3	4	lounge	51	2739	160000	1	40.633171	17.634
4	5	pop	73	3074	106880	1	41.903221	12.495
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704
1534	1535	lounge	74	3835	112000	1	45.845692	8.666
1535	1536	pop	51	2223	60457	1	45.481541	9.413
1536	1537	lounge	51	2557	80750	1	45.000702	7.682
1537	1538	pop	51	1766	54276	1	40.323410	17.568

1538 rows × 9 columns

In [3]:

```
df=df[['engine_power','age_in_days']]
df.columns=['ep','aid']
```

In [4]:

```
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   aid     1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [5]:

```
df.describe()
```

Out[5]:

	ep	aid
count	1538.000000	1538.000000
mean	51.904421	1650.980494
std	3.988023	1289.522278
min	51.000000	366.000000
25%	51.000000	670.000000
50%	51.000000	1035.000000
75%	51.000000	2616.000000
max	77.000000	4658.000000

In [6]:

```
df.head(20)
```

Out[6]:

	ep	aid
0	51	882
1	51	1186
2	74	4658
3	51	2739
4	73	3074
5	74	3623
6	51	731
7	51	1521
8	73	4049
9	51	3653
10	51	790
11	51	366
12	51	456
13	51	3835
14	51	1035
15	51	1096
16	73	4200
17	51	2223
18	51	2861
19	51	425

In [7]:

```
df.tail(20)
```

Out[7]:

	ep	aid
1518	51	397
1519	51	670
1520	51	1035
1521	51	3774
1522	51	366
1523	51	2251
1524	51	2192
1525	51	790
1526	51	1705
1527	51	517
1528	51	2861
1529	51	731
1530	51	670
1531	73	4505
1532	51	1917
1533	51	3712
1534	74	3835
1535	51	2223
1536	51	2557
1537	51	1766

In [8]:

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

```
C:\Users\chila\AppData\Local\Temp\ipykernel_17304\4116506308.py:1: Setting
WithCopyWarning:
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-doc
s/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)
  df.fillna(method='ffill',inplace=True)
```

In [9]:

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

In [10]:

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

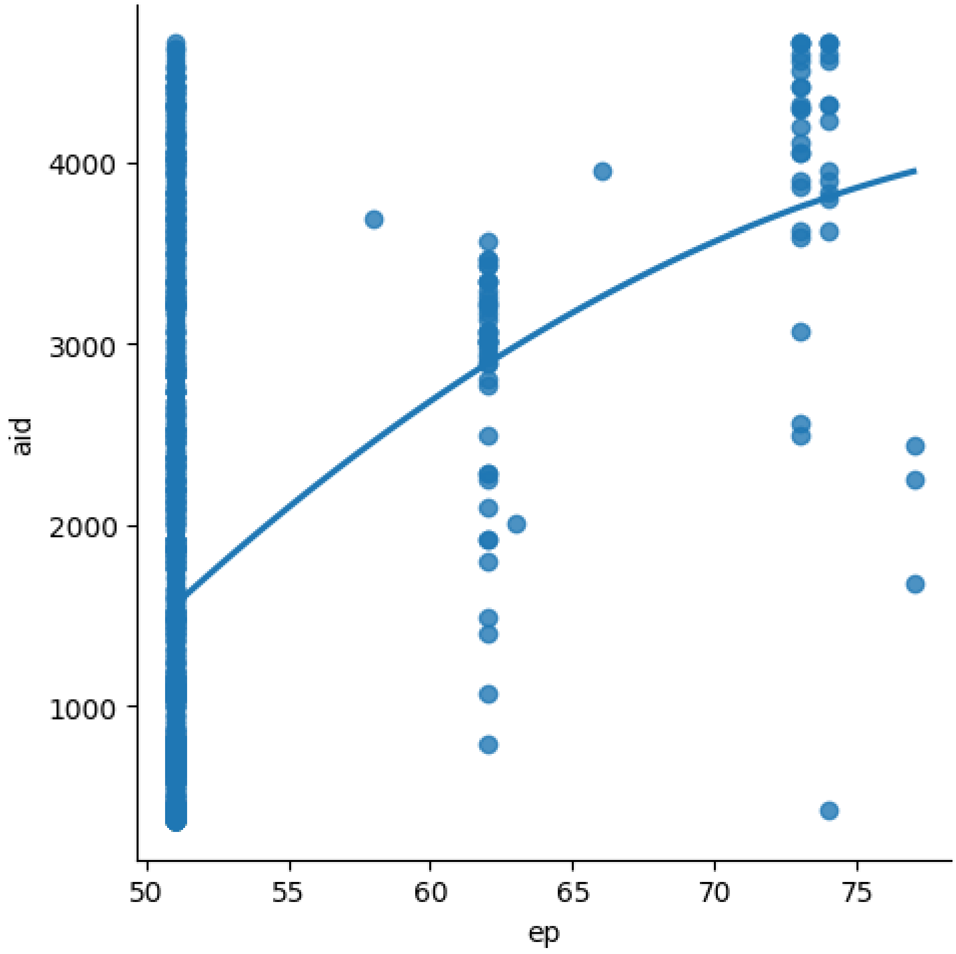
```
C:\Users\chila\AppData\Local\Temp\ipykernel_17304\1379821321.py:1: Setting
WithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/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)
  df.dropna(inplace=True)
```

In [11]:

```
#Exploring the data scatter_plotting the data scatter
sns.lmplot(x = "ep", y = "aid", data = df, order = 2, ci = None)
```

Out[11]:

```
<seaborn.axisgrid.FacetGrid at 0x1ba4f74b460>
```



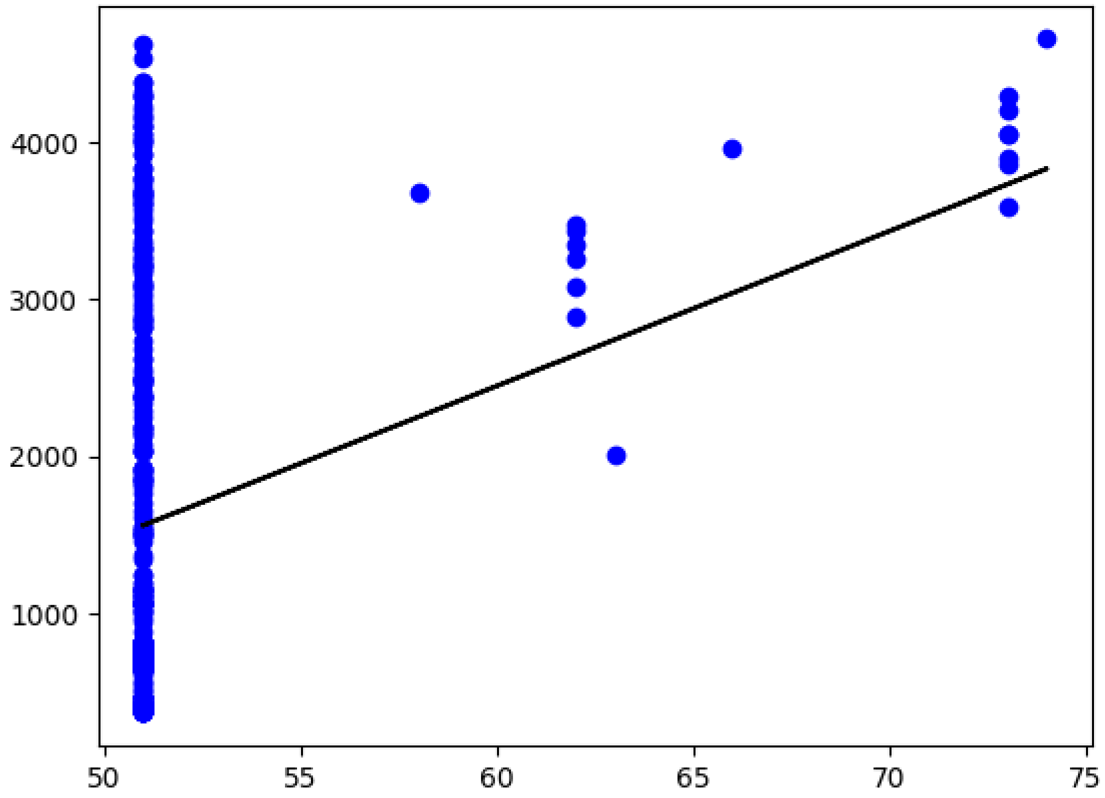
In [12]:

```
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))
```

```
0.10971550279341524
```

In [13]:

```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

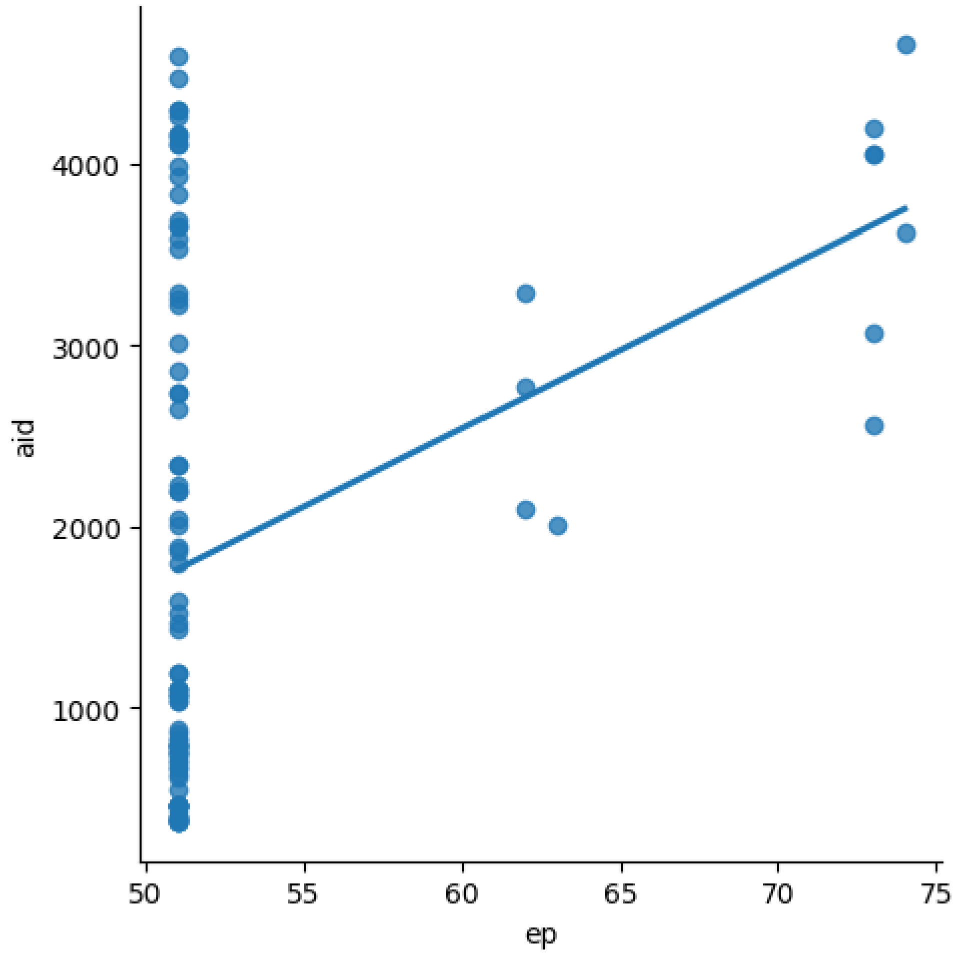


In [14]:

```
df100=df[:][:100]
sns.lmplot(x='ep',y='aid',data=df100,order=1,ci=None)
```

Out[14]:

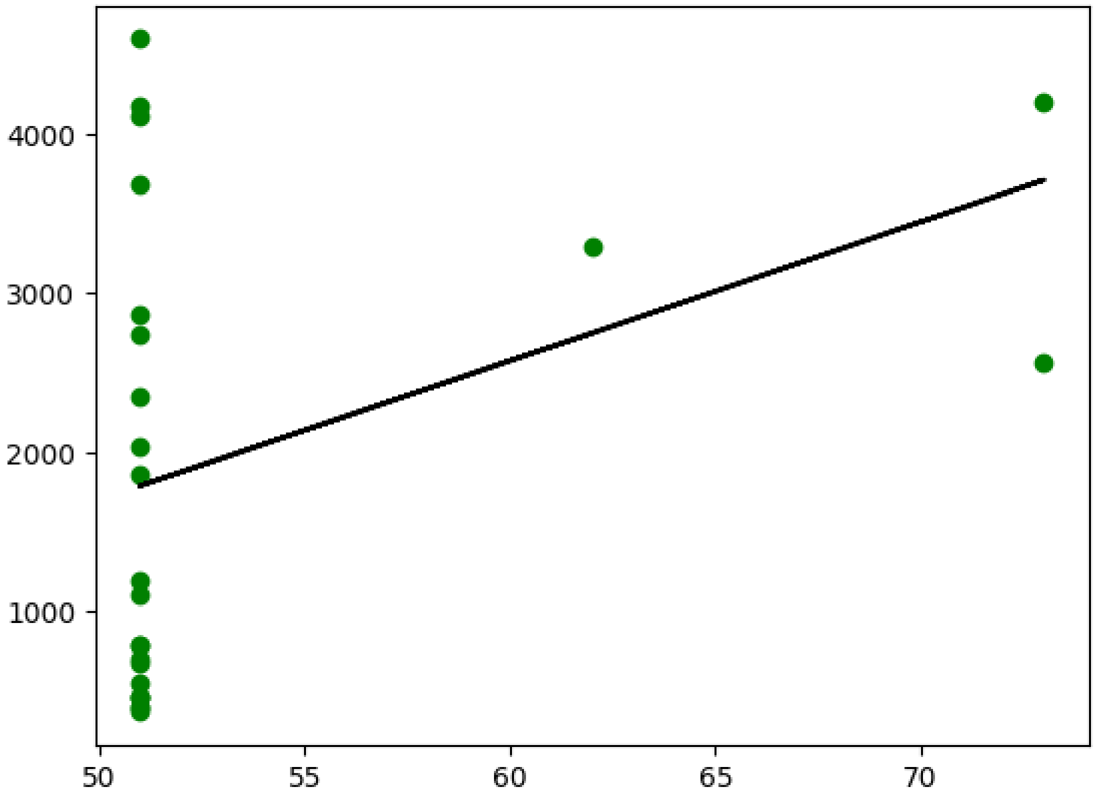
```
<seaborn.axisgrid.FacetGrid at 0x1ba45463a60>
```



In [15]:

```
df100.fillna(method='ffill',inplace=True)
X=np.array(df100['ep']).reshape(-1,1)
y=np.array(df100['aid']).reshape(-1,1)
df100.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))
print("Regression: ",regr.score(x_test,y_test))
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='g')
plt.plot(x_test,y_pred,color='k')
plt.show()
```

```
0.12893418004674928
Regression:  0.12893418004674928
```



In [16]:

```
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:  0.12893418004674928
```

# Conclusion:

Dataset we have taken is poor for linear model but with the smaller data works well with linear model