In [1]:
```
#step-1:problem statement
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 [4]:
```
#step-2:reading the data set
df=pd.read_csv(r"C:\Users\chinta pavani\Documents\fiat500_VehicleSelection_
df
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	
0	1	lounge	51	882	25000	1	44.907242	8.61
1	2	pop	51	1186	32500	1	45.666359	12.24
2	3	sport	74	4658	142228	1	45.503300	11.41
3	4	lounge	51	2739	160000	1	40.633171	17.63
4	5	pop	73	3074	106880	1	41.903221	12.49
...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.70
1534	1535	lounge	74	3835	112000	1	45.845692	8.66
1535	1536	pop	51	2223	60457	1	45.481541	9.41
1536	1537	lounge	51	2557	80750	1	45.000702	7.68
1537	1538	pop	51	1766	54276	1	40.323410	17.56

1538 rows × 9 columns

In [5]:
```
df=df[['engine_power','age_in_days']]
df.columns=['Eng','Age']
```

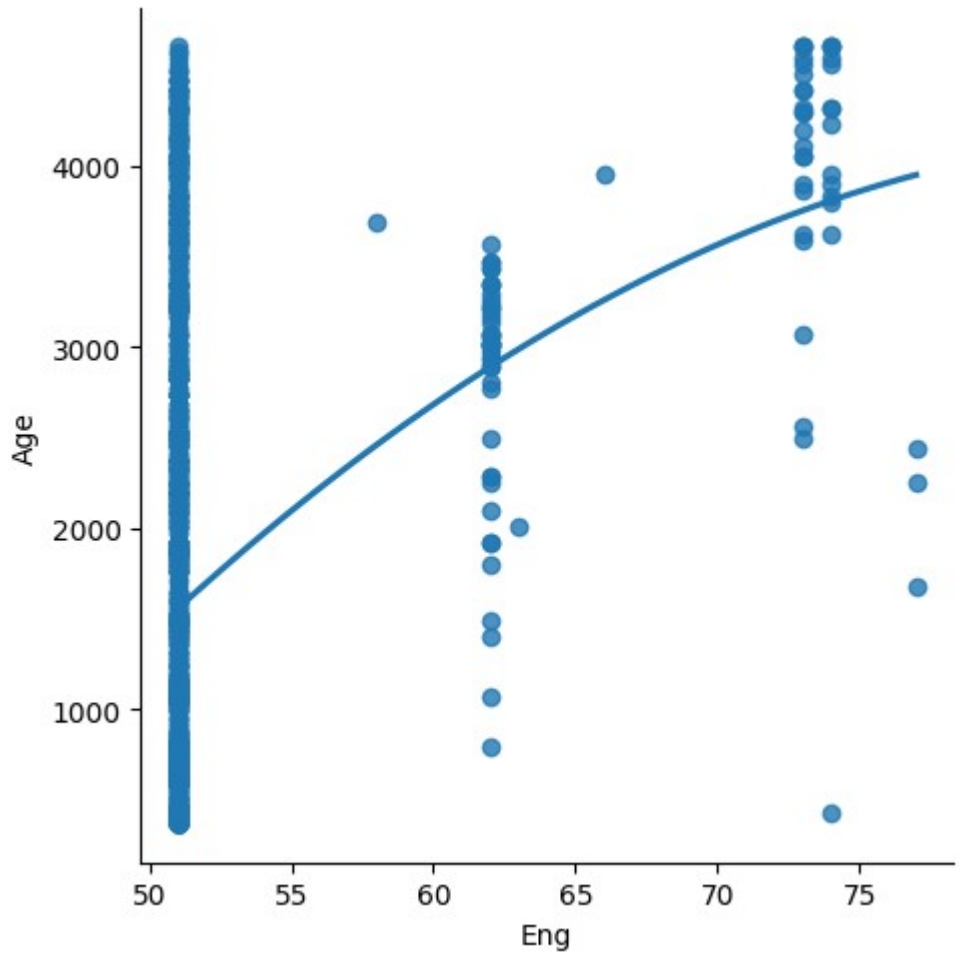
In [6]:
```
df.head(10)
```

Out[6]:

	Eng	Age
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

In [7]:
```
#step-3:exploring the data scatter-plotting the data scatter
sns.lmplot(x="Eng",y="Age",data=df,order=2,ci=None)
```

Out[7]: <seaborn.axisgrid.FacetGrid at 0x25136eb0a10>



In [8]:
```
df.describe()
```

Out[8]:

	Eng	Age
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 [9]:
```
df.info()
```

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

In [22]:
```
#step-4:data cleaning-eliminating nan or missing input
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_28804\4116506308.py:
1: SettingWithCopyWarning:
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-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.fillna(method='ffill',inplace=True)
```

In [11]:
```
#step-5:training our model
x=np.array(df['Eng']).reshape(-1,1)
y=np.array(df['Age']).reshape(-1,1)
```

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

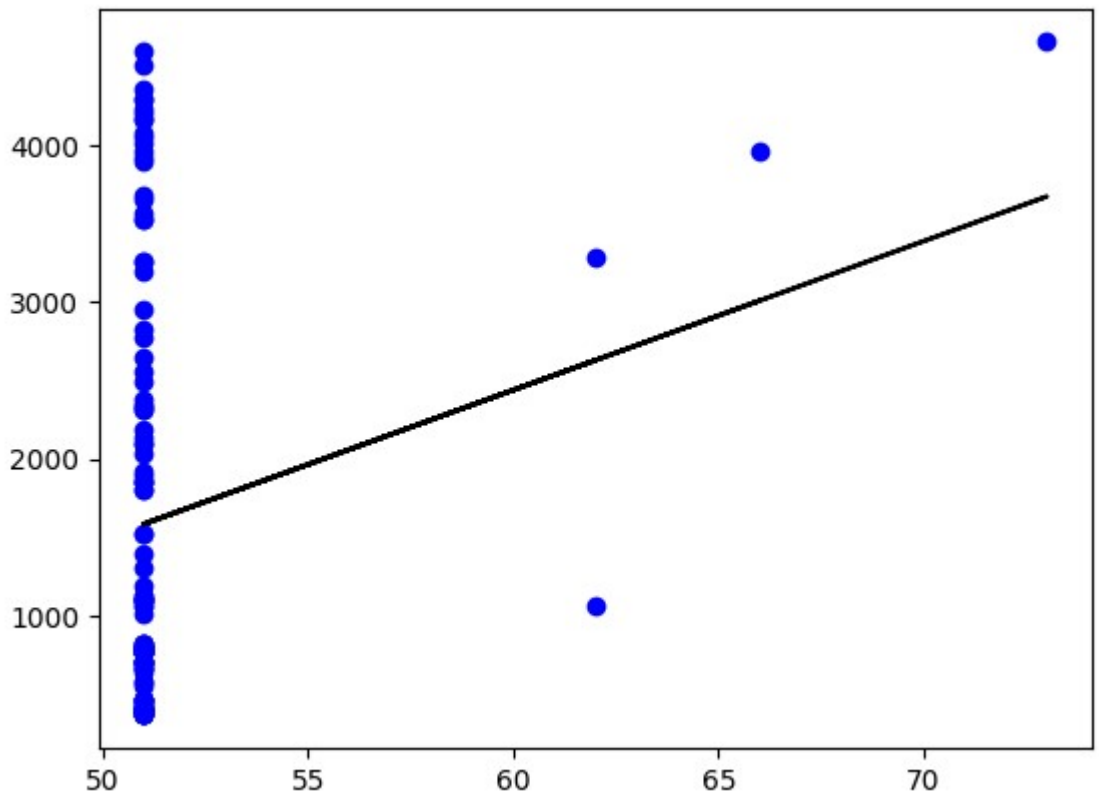
```
C:\Users\chinta pavani\AppData\Local\Temp\ipykernel_28804\1379821321.py:
1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-do
cs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (http
s://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returni
ng-a-view-versus-a-copy)
  df.dropna(inplace=True)
```

In [13]:
```
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))
```

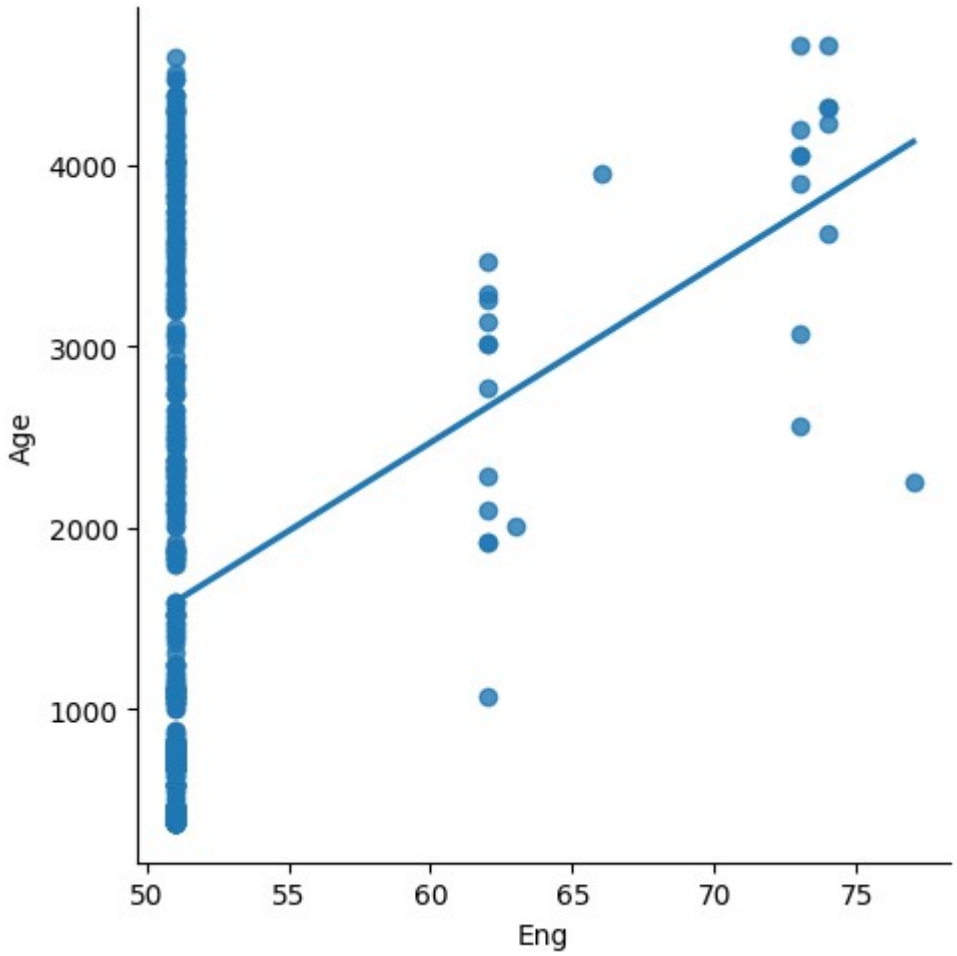
```
0.10844042252426467
```

In [21]:
```
#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
```



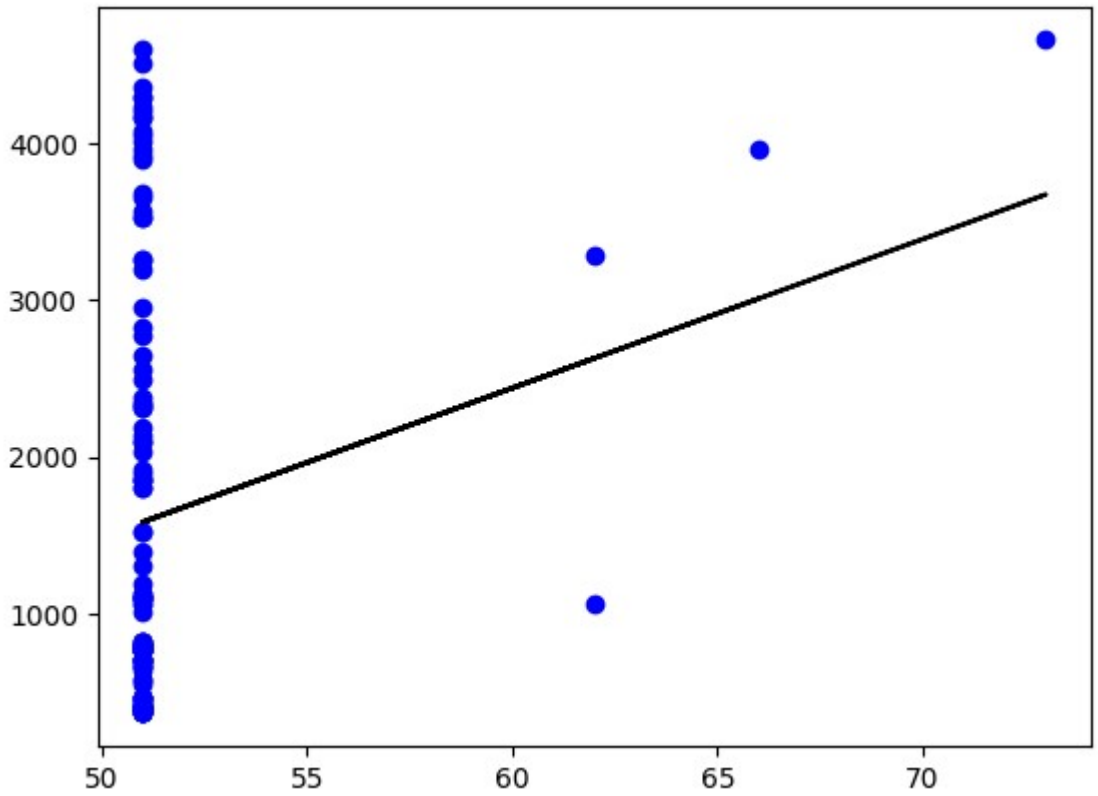
In [20]:
```
#step-7:working with a smallest dataset
df500=df[:][:500]
sns.lmplot(x="Eng",y="Age",data=df500,order=1,ci=None)
```

Out[20]: <seaborn.axisgrid.FacetGrid at 0x25136eb2090>



In [18]:
```
df500.fillna(method='ffill',inplace=True)
x=np.array(df500['Eng']).reshape(-1,1)
y=np.array(df500['Age']).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: 0.05674189944888408
```



In [19]:
```
#step-8:evaluation of model
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: 0.05674189944888408
```

In [ ]:
```
#step-9:conclusion
#dataset we have taken is poor for linear model but with the smaller data
```