```
Vehicle Selection (project2)
1)problem statement:Hw best fit the data set
```

In [8]:
```
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 [9]:
```
df=pd.read_csv(r"C:\Users\Jayadeep\Downloads\fiat500_VehicleSelection_Dataset (1).csv")
df
```

Out[9]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [10]:
```
df=df[['engine_power','km']]
df.columns=['engine','Km']
```

In [11]:
```
df.head(10)
```

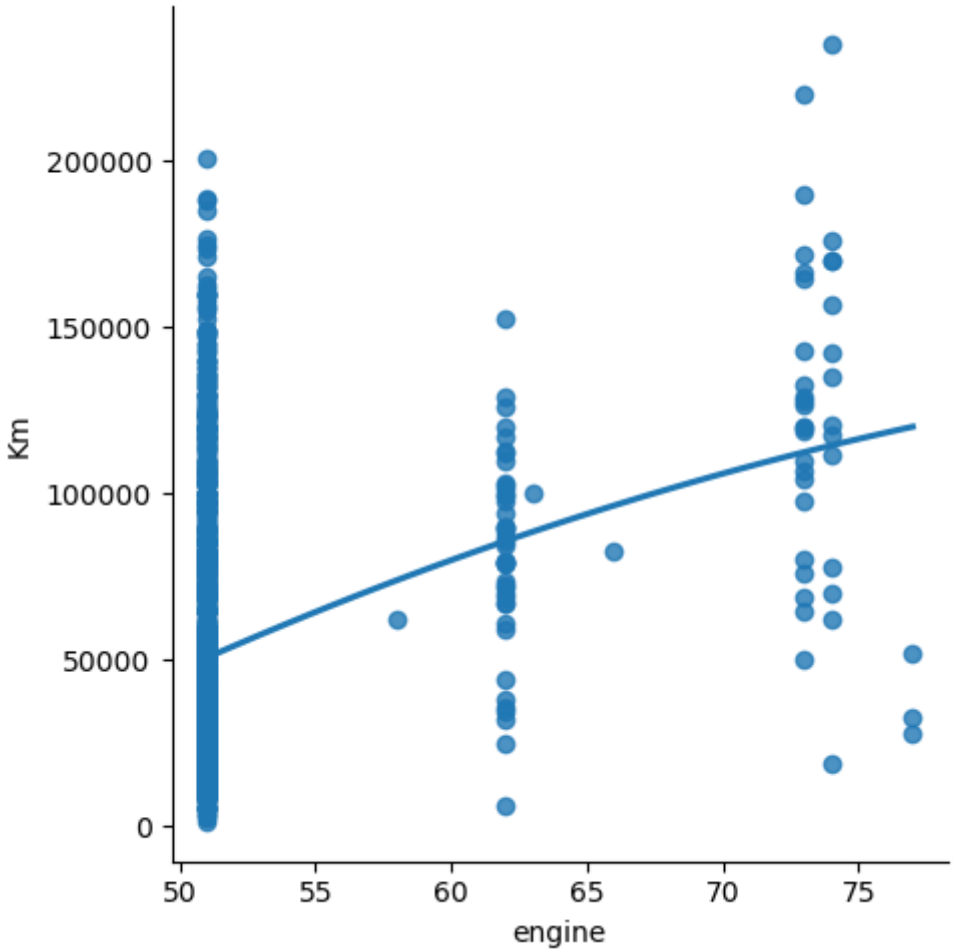
Out[11]:

	engine	Km
0	51	25000
1	51	32500
2	74	142228
3	51	160000
4	73	106880
5	74	70225
6	51	11600
7	51	49076
8	73	76000
9	51	89000

3)Exploing the data scatter-plottting the data scatter

In [12]:
```
sns.lmplot(x='engine',y='Km',data=df,order=2,ci=None)
```

Out[12]:
```
<seaborn.axisgrid.FacetGrid at 0x27c6dd1bdc0>
```



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

Out[13]:

	engine	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 [14]:
```
df.info()
```

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

4)Data cleaning-Eliminating nan or missing i/p numbers

In [15]:
```
df.fillna(method='ffill',inplace=True)
```

```
C:\Users\Jayadeep\AppData\Local\Temp\ipykernel_21024\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-docs/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)
```

5)Training our model

In [16]:
```
x=np.array(df['engine']).reshape(-1,1)
y=np.array(df['Km']).reshape(-1,1)
```

In [17]:
```
df.dropna(inplace=True)
```

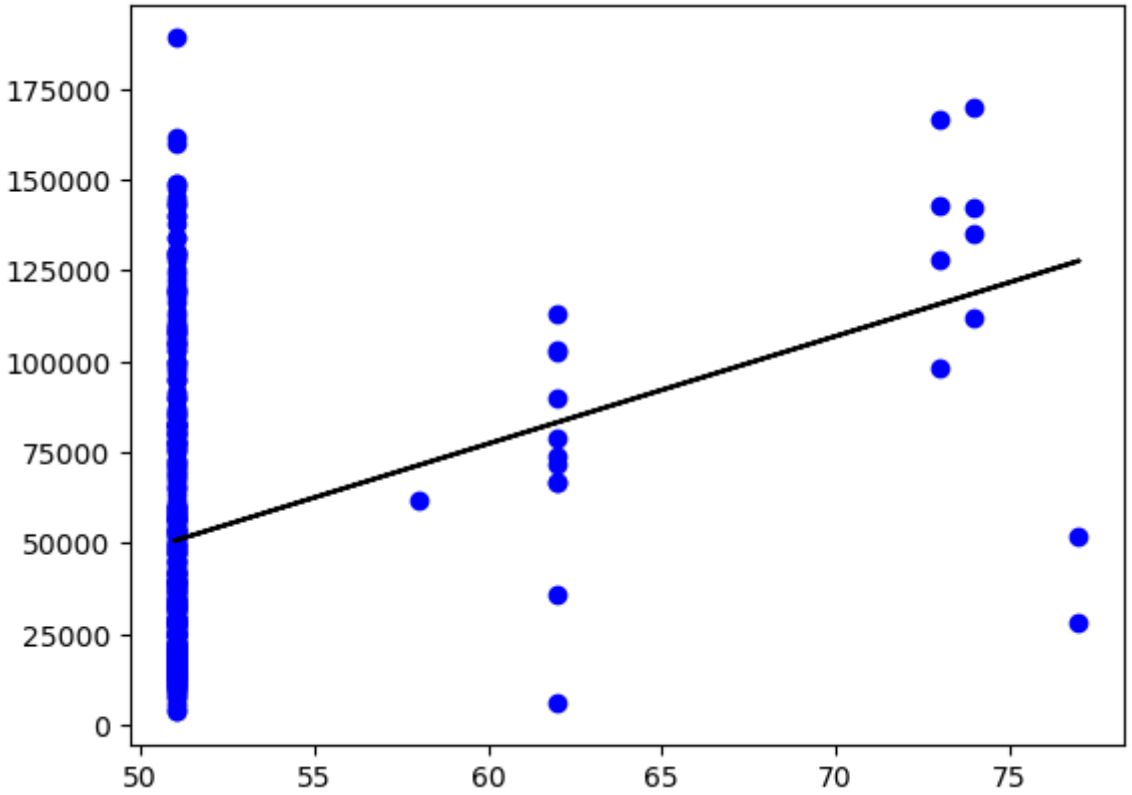
```
C:\Users\Jayadeep\AppData\Local\Temp\ipykernel_21024\1379821321.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/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 [18]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
#splitting data into train and test
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.07159243790417236
```

6)Exploring our results

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()
```



7)working with smaller dataset

In [21]:
```
df500=df[:][:500]
sns.lmplot(x="engine",y="Km",data=df500,order=1,ci=None)
```

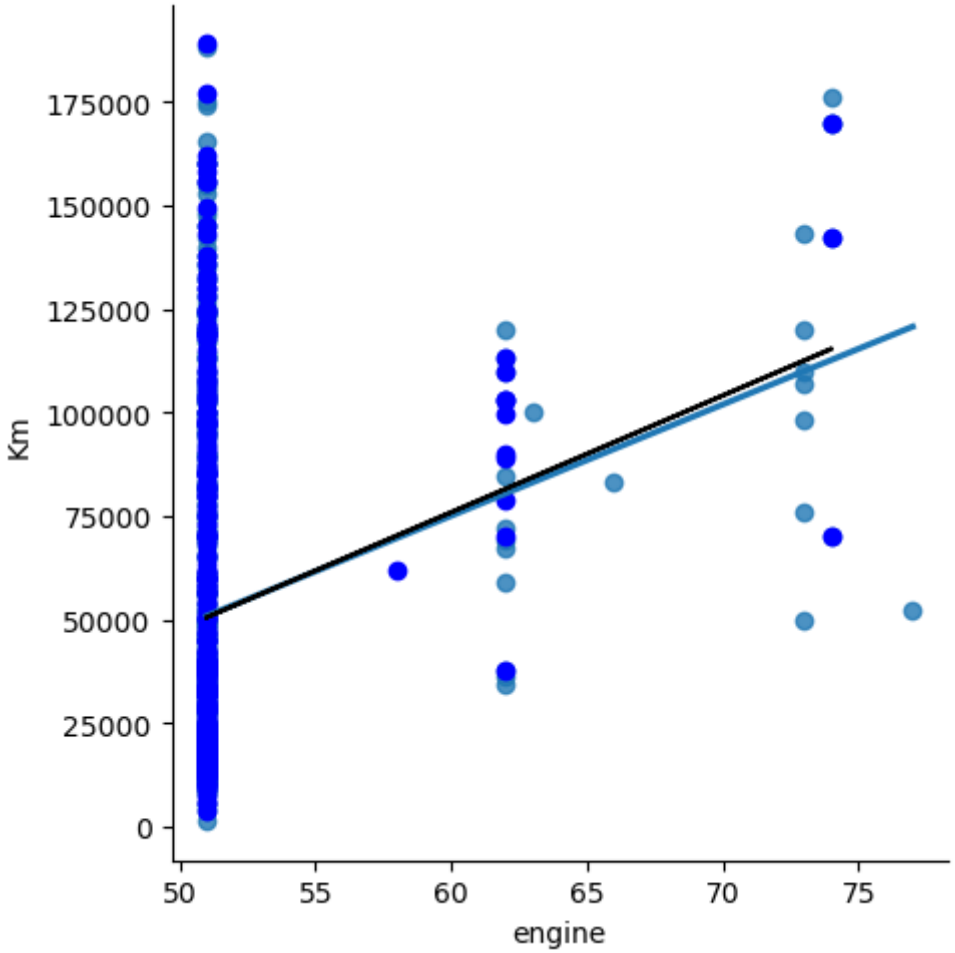
Out[21]:
```
<seaborn.axisgrid.FacetGrid at 0x27c6e72efd0>
```

In [22]:
```
df500.fillna(method='ffill',inplace=True)
```

In [23]:
```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
#splitting data into train and test
regr=LinearRegression()
regr.fit(x_train,y_train)
print('Regression:',regr.score(x_test,y_test))
```

```
Regression: 0.04919109342805206
```

In [24]:
```
y_pred=regr.predict(x_test)
plt.scatter(x_test,y_test,color='b')
plt.plot(x_test,y_pred,color='k')
plt.show()
```



8)Evaluation of model

In [25]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
```

In [26]:
```
#train model
model=LinearRegression()
model.fit(x_train,y_train)
#Evaluation 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: 0.04919109342805206
```

In [ ]:
```
conclusion: Data set  we have taken is poor for this model
```