In [45]:

```
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 [46]:

```
df=pd.read_csv(r"C:\Users\MY HOME\Downloads\fiat500_VehicleSelection_Dataset.csv")
df
```

Out[46]:

	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 [47]:

```
#taking selected columns from dataset
df=df[['age_in_days','km']]

df
```

Out[47]:

```
      age_in_days      km
   0          882   25000
   1         1186   32500
   2         4658  142228
   3         2739  160000
   4         3074  106880
 ...          ...     ...
1533         3712  115280
1534         3835  112000
1535         2223   60457
1536         2557   80750
1537         1766   54276

1538 rows × 2 columns
```

In [48]:

```
#Renamin columns for easier process(OPTIONAL)
df.columns=['age_in_days','km']
df
```

Out[48]:

```
      age_in_days      km
   0          882   25000
   1         1186   32500
   2         4658  142228
   3         2739  160000
   4         3074  106880
 ...          ...     ...
1533         3712  115280
1534         3835  112000
1535         2223   60457
1536         2557   80750
1537         1766   54276

1538 rows × 2 columns
```

In [49]:

```
df.head(10)
```

Out[49]:

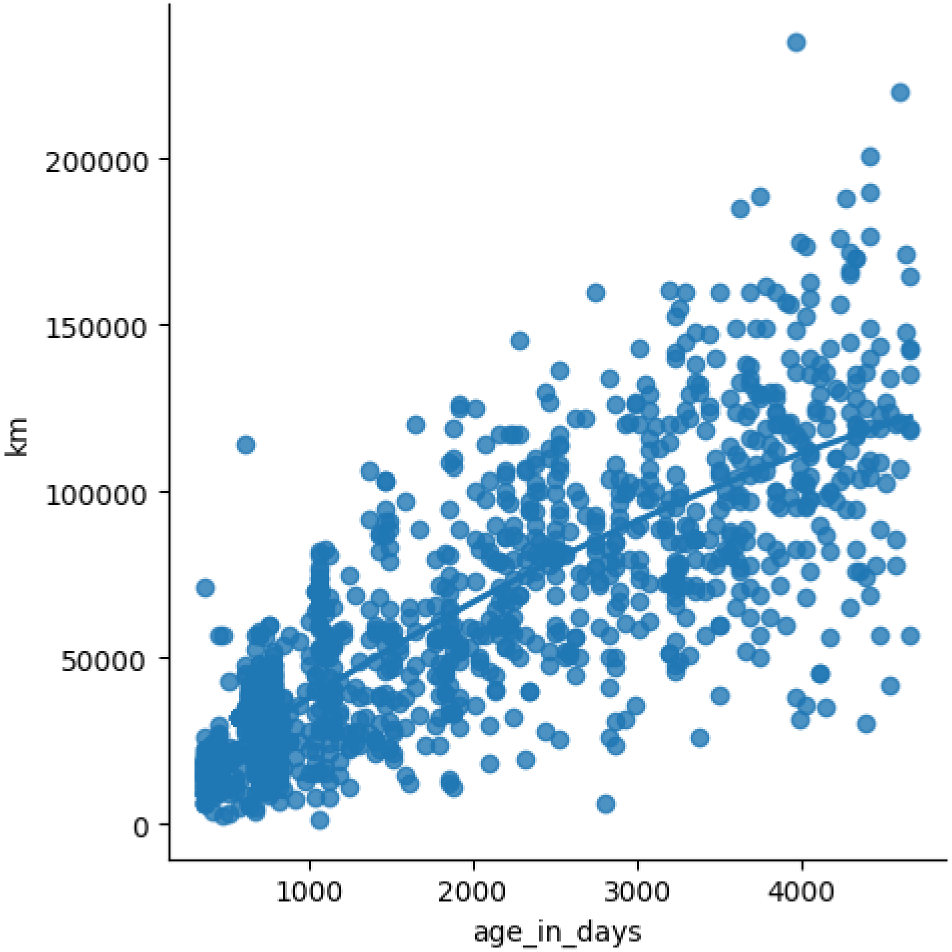
	age_in_days	km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
5	3623	70225
6	731	11600
7	1521	49076
8	4049	76000
9	3653	89000

In [ ]:

```
###step 3: Exploring to data scatter-pltting the data
```

In [50]:

```
sns.lmplot(x="age_in_days",y="km",order=2,data=df,ci=None)
plt.show()
```



In [51]:

```
df.describe()
```

Out[51]:

	age_in_days	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 [52]:

```
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_in_days  1538 non-null   int64
 1   km           1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```

In [53]:

```
##step 4:Data cleaning-Eliminating Nan/missing values
```

In [54]:

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

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_7796\2729279820.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)
```

Out[54]:

	age_in_days	km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
...	...	...
1533	3712	115280
1534	3835	112000
1535	2223	60457
1536	2557	80750
1537	1766	54276

1538 rows × 2 columns

step 5:Training our model

In [55]:

```
#Separating data into independent & dependent variables
#Now each dataframe contains only one coloumn
x=np.array(df['age_in_days']).reshape(-1,1)
y=np.array(df['km']).reshape(-1,1)
#Dropping any rows with Nan values
df.dropna(inplace=True)
df
```

```
C:\Users\MY HOME\AppData\Local\Temp\ipykernel_7796\49978593.py:6: 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#ret
urning-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)
```

Out[55]:

	age_in_days	km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
...	...	...
1533	3712	115280
1534	3835	112000
1535	2223	60457
1536	2557	80750
1537	1766	54276

1538 rows × 2 columns

In [56]:

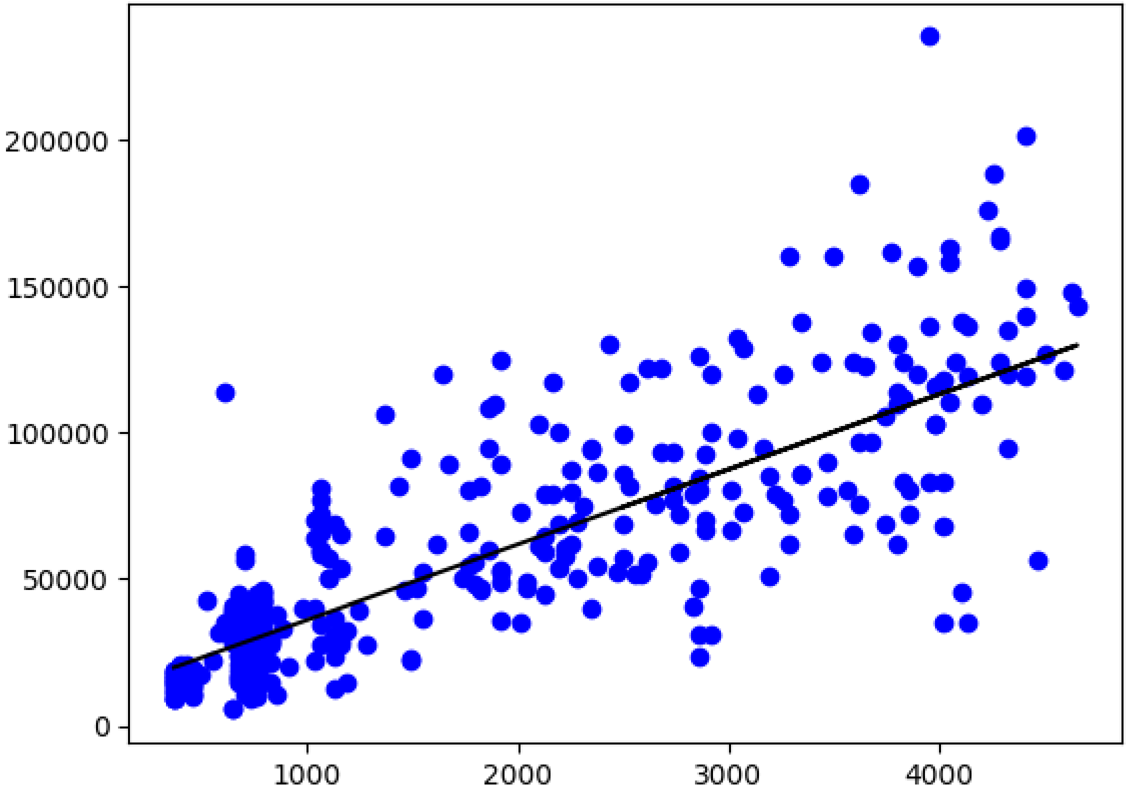
```
#Splitting the data into training and testing data
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25)
regr=LinearRegression()
regr.fit(x_train,y_train)
print(regr.score(x_test,y_test))
```

```
0.6740218819186333
```

###step 6:Exploring our results

In [57]:

```
#Data scatter to predict the values
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 [58]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#Train the model
model=LinearRegression()
model.fit(x_train,y_train)
#evaluate 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: 0.6740218819186333
```

###step 7:Working with the smaller dataset

In [65]:

```
#selecting the first 500 rows
df500=df[:][:400]
df500
```

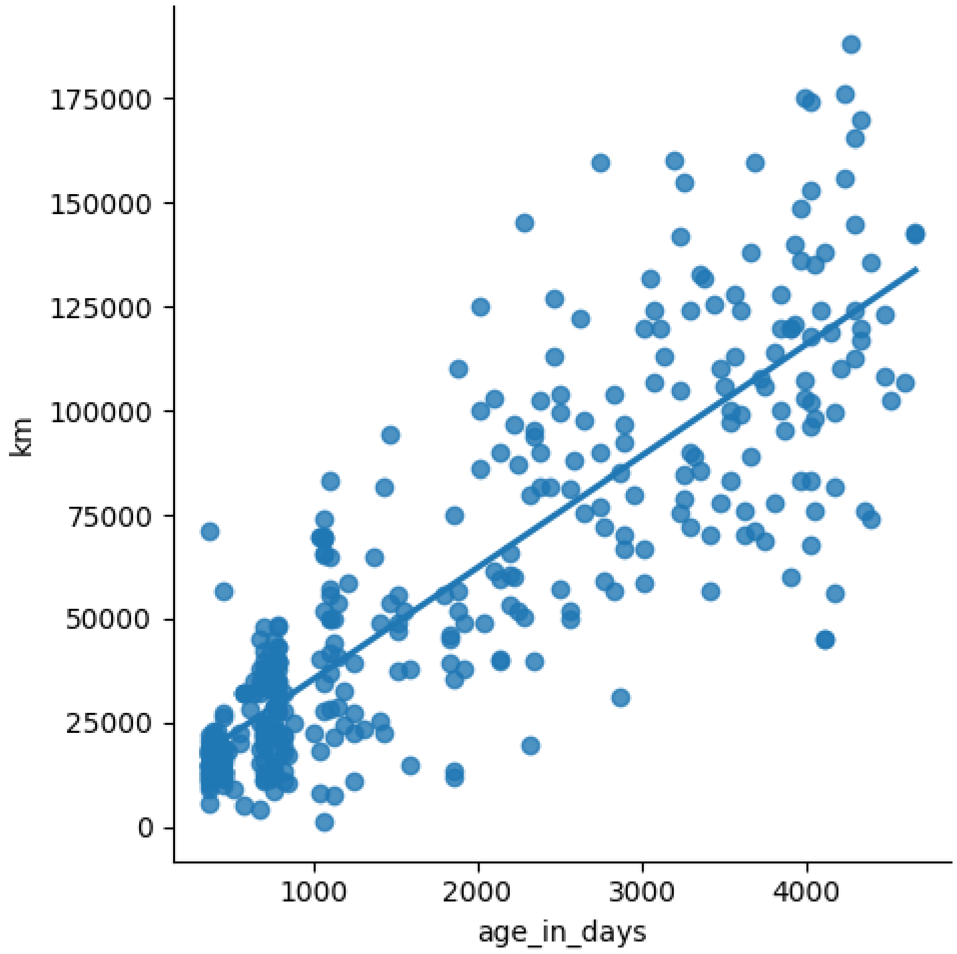
Out[65]:

```
     age_in_days      km
  0          882   25000
  1         1186   32500
  2         4658  142228
  3         2739  160000
  4         3074  106880
...          ...     ...
395          366   18818
396          821   10800
397          578   32057
398         1035   69900
399         3258  155000

400 rows × 2 columns
```

In [66]:

```
sns.lmplot(x="age_in_days",y="km",data=df500,order=1,ci=None)
plt.show()
```



In [67]:

```
df500.fillna(method='ffill',inplace=True)
df500
```

Out[67]:

	age_in_days	km
0	882	25000
1	1186	32500
2	4658	142228
3	2739	160000
4	3074	106880
...	...	...
395	366	18818
396	821	10800
397	578	32057
398	1035	69900
399	3258	155000

400 rows × 2 columns

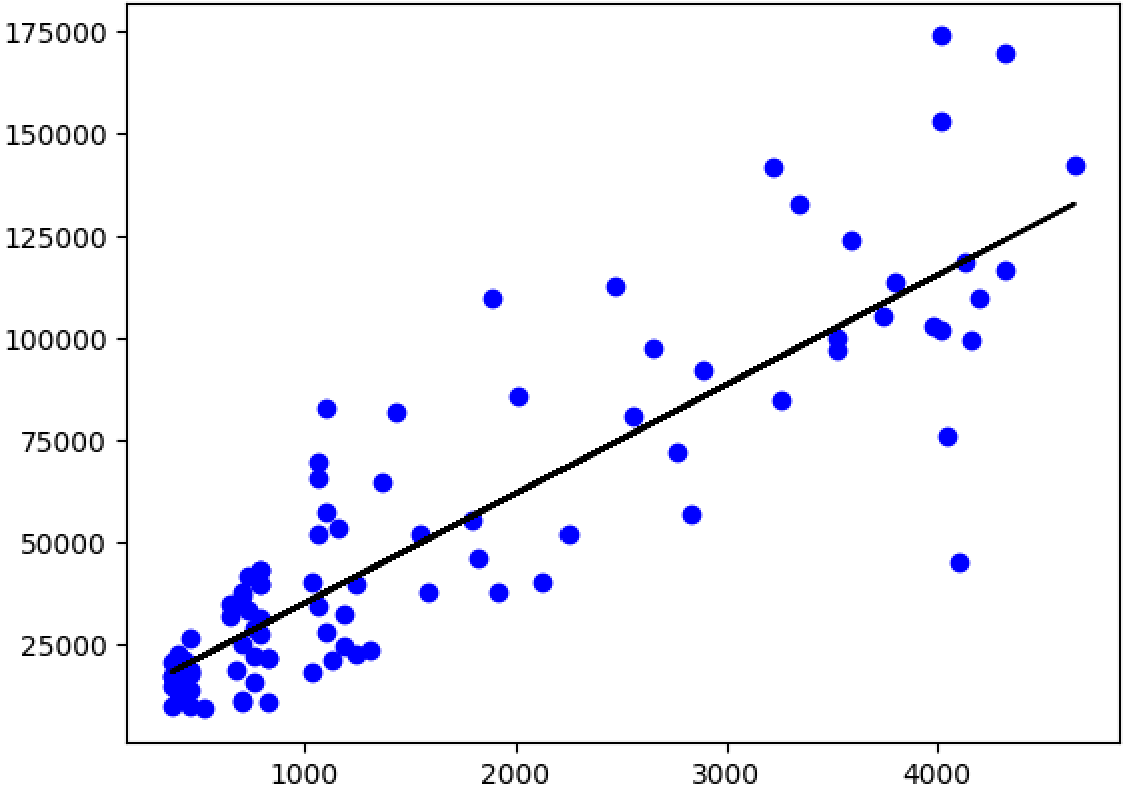
In [68]:

```
x=np.array(df500['age_in_days']).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))
```

```
regression: 0.7722892542047648
```

In [69]:

```
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 [70]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
#Train the model
model=LinearRegression()
model.fit(x_train,y_train)
#evaluate 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: 0.7722892542047648
```

conclusion:

the dataset we have taken is acceptable.but,maybe it cannot be a best fit .