Linear Regression- Vehicle selection ¶

In [18]:

```
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 [3]:

```
#Step-2: Reading the Dataset
df=pd.read_csv(r"C:\Users\sudheer\Downloads\fiat500_VehicleSelection_Dataset (1).csv
df
```

Out[3]:

	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 [4]:

```
df=df[['age_in_days','price']]
df.columns=['age_in_days','price']
```

In [5]:

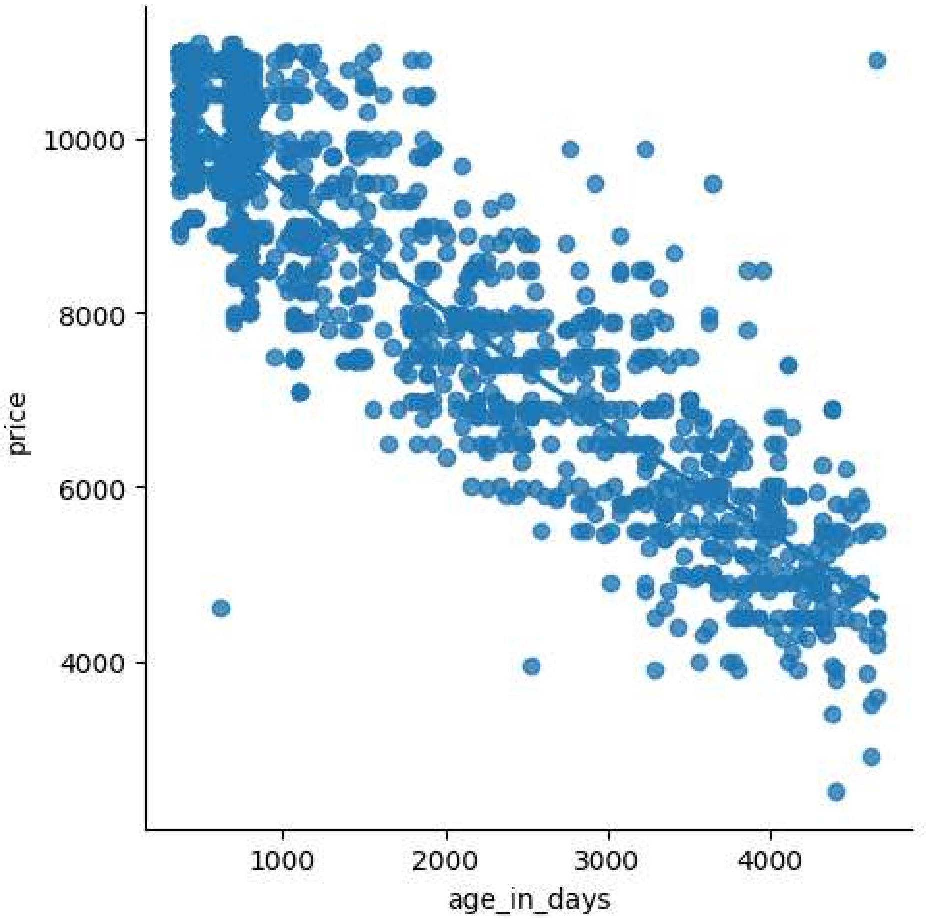
```
df.head()
```

Out[5]:

	age_in_days	price
0	882	8900
1	1186	8800
2	4658	4200
3	2739	6000
4	3074	5700

In [7]:
```
#Step-3: Exploring the Data Scatter - plotting the data scatter
sns.lmplot(x="age_in_days",y="price", data = df, order = 2, ci = None)
df.describe()
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   price        1538 non-null   int64
dtypes: int64(2)
memory usage: 24.2 KB
```



In [8]:
```
#Step-4: Data cleaning - Eliminating NaN OR missing input numbers
df.fillna(method ='ffill', inplace = True)
```

```
C:\Users\sudheer\AppData\Local\Temp\ipykernel_24120\3221840372.py:2: SettingWithCop
yWarning:
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)
```

In [10]:

```
# Step-5: Training Our Model
X = np.array(df['age_in_days']).reshape(-1, 1)
y = np.array(df['price']).reshape(-1, 1)
#Seperating the data into independent and dependent variables and convert
#Now each dataset contains only one column
```

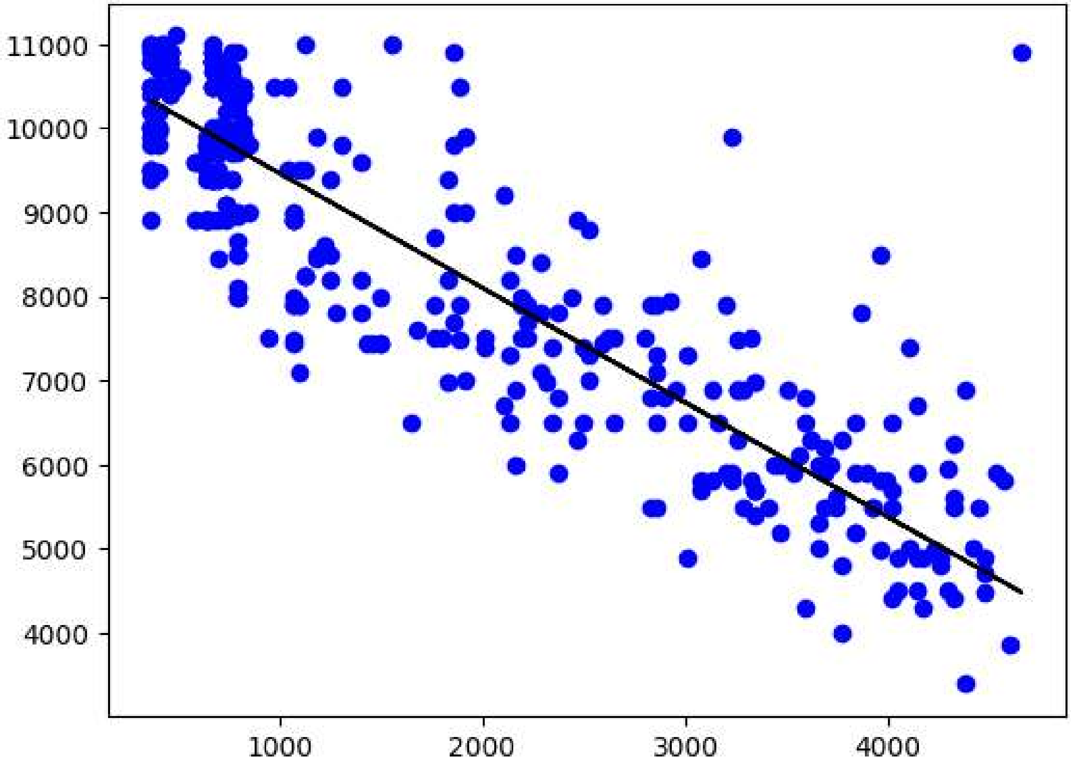
In [11]:

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

```
0.7814006664917448
```

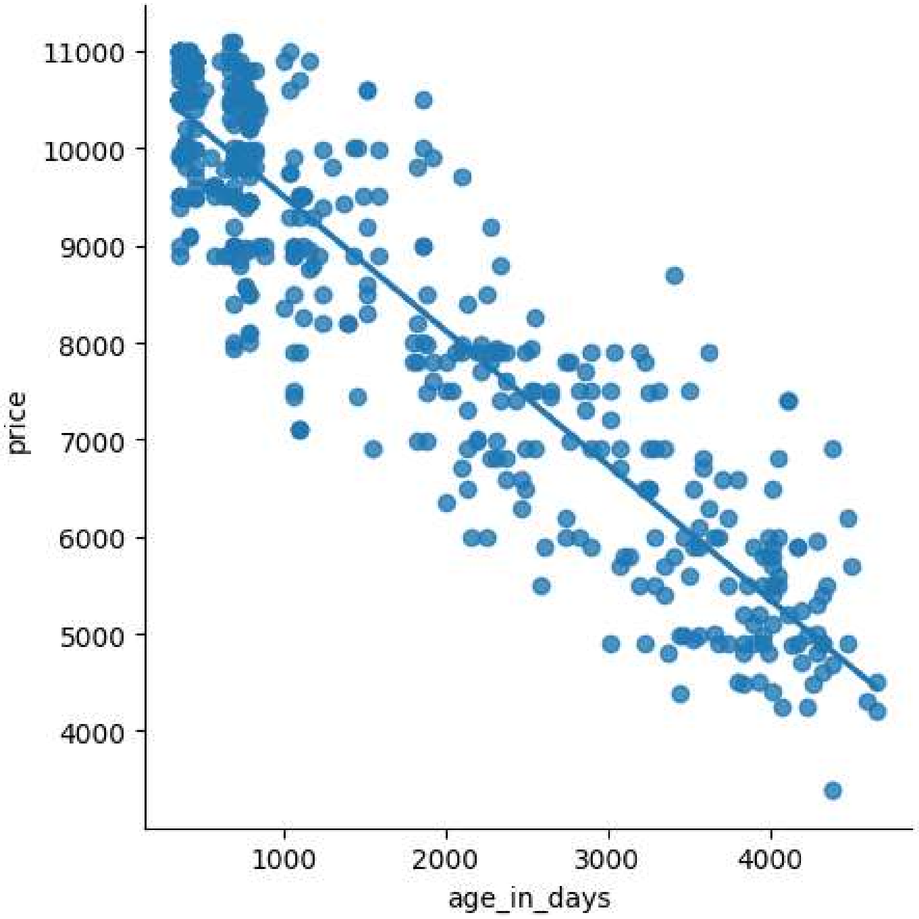
In [12]:

```
#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
```



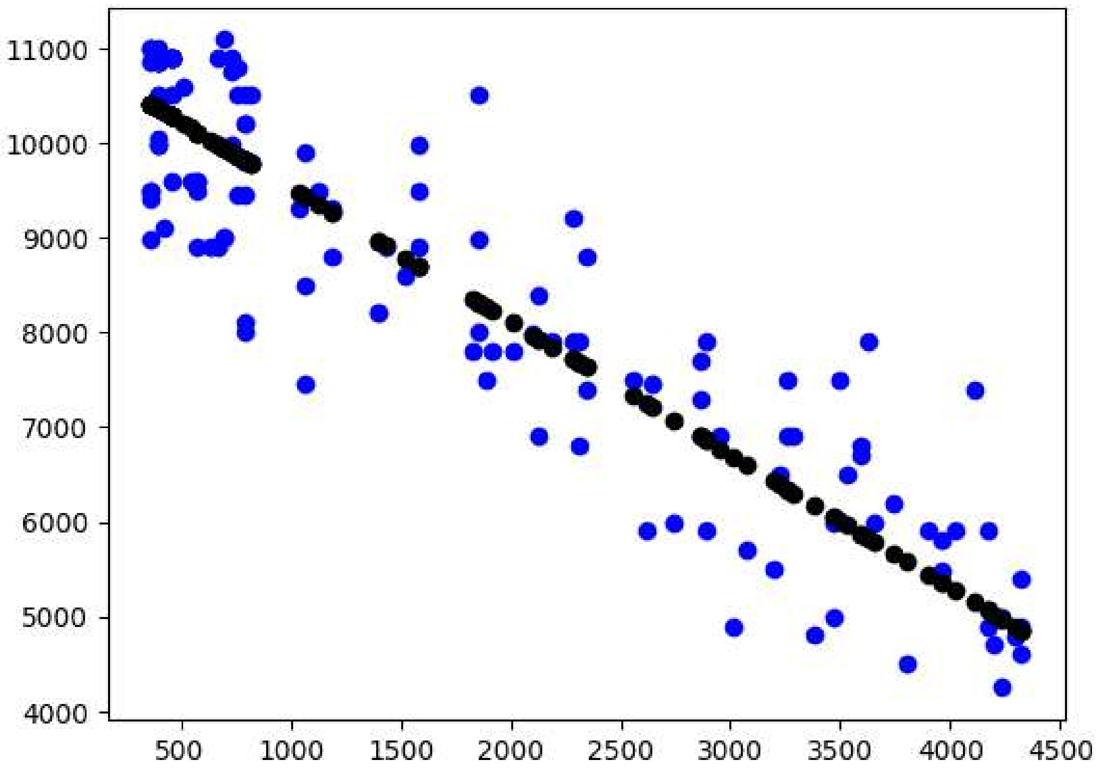
In [13]:
```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of data
sns.lmplot(x = "age_in_days", y ="price", data = df500, order = 1, ci = None)
```

Out[13]:
```
<seaborn.axisgrid.FacetGrid at 0x1d97fb8e8c0>
```



```
In [15]: df500.fillna(method = 'ffill', inplace = True)
         X = np.array(df500['age_in_days']).reshape(-1, 1)
         y = np.array(df500['price']).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.scatter(X_test, y_pred, color = 'k')
         plt.show()
```

```
Regression: 0.8195780898732068
```



In [16]:

```
#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)

#Evaluating 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: 0.8195780898732068
```

Step-9:Conclusion:

Dataset we have taken is poor for Linear Model,but with the smaller data works well with Linear Model.

In [ ]:

```

```