In [ ]:

```
```

LineaRegression-vehicle selection

In [19]:

```
# Step-1 :Importing all the required libraries
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 [24]:

```
df=pd.read_csv(r"C:\Users\jyothi reddy\Downloads\fiat500_VehicleSelection_Dataset (1).csv"
df
```

Out[24]:

	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

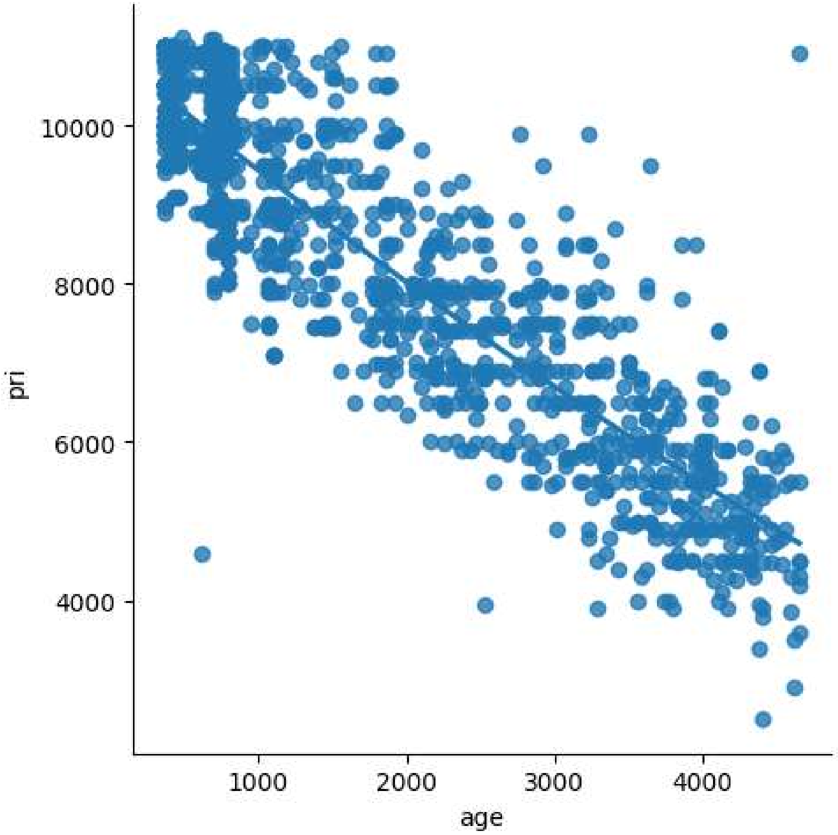
1538 rows × 9 columns

In [25]:

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

```
In [26]: sns.lmplot(x="age",y="pri", data = df, order = 2, ci = None)
```

```
Out[26]: <seaborn.axisgrid.FacetGrid at 0x133c1803790>
```



```
In [27]: df.head(10)
```

Out[27]:

	age	at
0	882	8900
1	1186	8800
2	4658	4200
3	2739	6000
4	3074	5700
5	3623	7900
6	731	10750
7	1521	9190
8	4049	5600
9	3653	6000

In [28]:

```
df.describe()
```

Out[28]:

	age	at
count	1538.000000	1538.000000
mean	1650.980494	8576.003901
std	1289.522278	1939.958641
min	366.000000	2500.000000
25%	670.000000	7122.500000
50%	1035.000000	9000.000000
75%	2616.000000	10000.000000
max	4658.000000	11100.000000

In [29]:

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

In [30]:

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

```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_13408\48824337.py:1: SettingWithCopyWa
rning:
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_g
uide/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 [31]:

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

In [32]:

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

```
C:\Users\jyothi reddy\AppData\Local\Temp\ipykernel_13408\1791587065.py:1: SettingWithCopy
Warning:
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_g
uide/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)
```

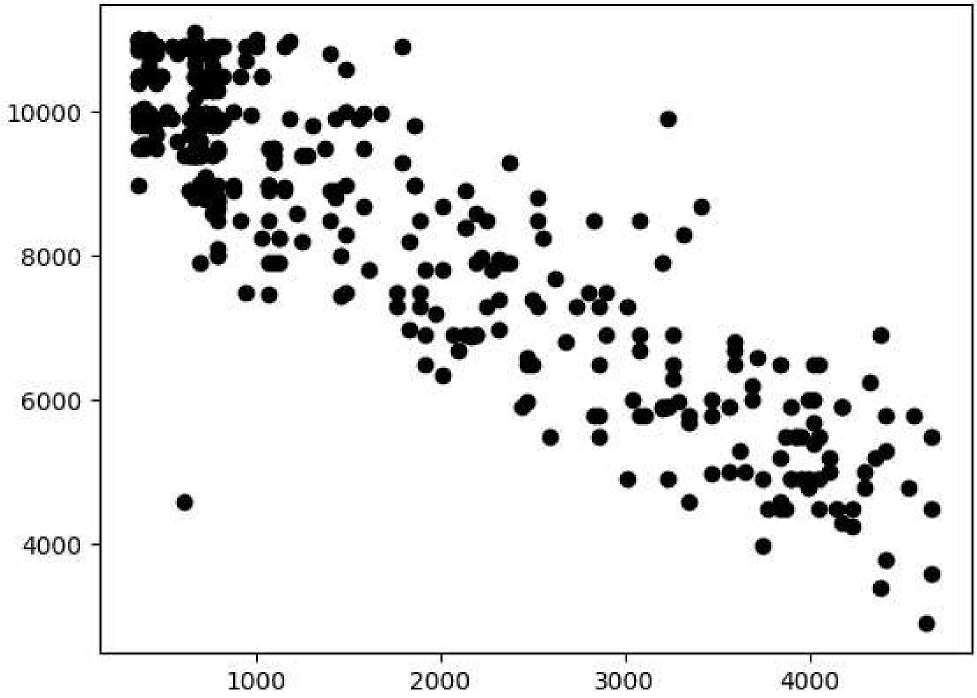
In [33]:

```
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))
```

```
0.7988385840550731
```

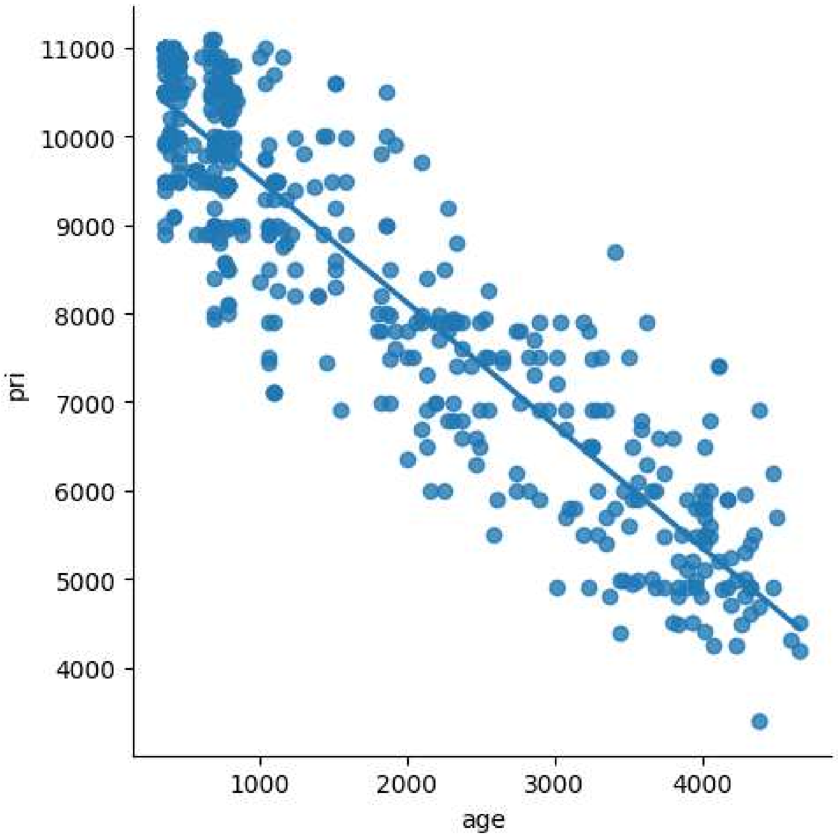
In [34]:

```
#step-6: Exploring Our Results
y_pred = regr.predict(X_test)
plt.scatter(X_test, y_test, color = 'k')
plt.show()
# Data scatter of predicted values
```



In [35]:
```
# Step-7: Working with a smaller Dataset
df500 = df[:][:500]
# Selecting the 1st 500 rows of the data
sns.lmplot(x ="age", y ="pri", data = df500, order = 1, ci = None)
```

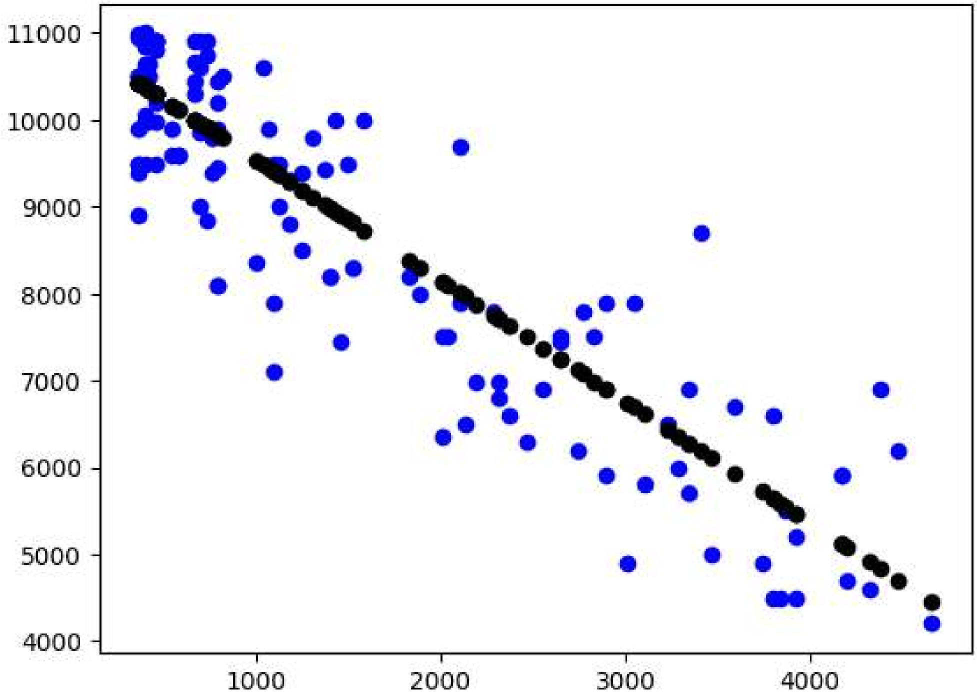
Out[35]:
```
<seaborn.axisgrid.FacetGrid at 0x133c3e7fd50>
```



In [36]:

```
df500.fillna(method = 'ffill', inplace = True)
X = np.array(df500['age']).reshape(-1, 1)
y = np.array(df500['pri']).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: 0.8055415735693708
```



In [37]:

```
#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: 0.8055415735693708
```

Step 9-conclusion: Data set we have taken is poor for linear model but with the smaller data works well with Linear model

In [ ]: