In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
s=pd.read_csv(r"C:\Users\user\Downloads\fiat500_VehicleSelection_Dataset - fiat500_Vehi
s
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pric
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	890
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	880
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	420
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	600
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	570
...	...	...	...	...	...	...	...	...	.
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonpric
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	N(
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	

1549 rows × 11 columns

In [3]:
```
s=s.head(100)
s
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700
...	...	...	...	...	...	...	...	...	...
95	96.0	sport	51.0	4292.0	165600.0	1.0	44.715408	11.30830002	5950
96	97.0	pop	51.0	1066.0	28000.0	1.0	41.769051	12.66281033	8500
97	98.0	sport	51.0	2009.0	86000.0	2.0	40.633171	17.63460922	7800
98	99.0	lounge	51.0	456.0	18592.0	2.0	45.393600	10.48223972	10900
99	100.0	pop	51.0	731.0	41558.0	2.0	45.571220	9.159139633	8790

100 rows × 11 columns

In [4]:
```
s.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               100 non-null    float64
 1   model            100 non-null    object 
 2   engine_power     100 non-null    float64
 3   age_in_days      100 non-null    float64
 4   km               100 non-null    float64
 5   previous_owners  100 non-null    float64
 6   lat              100 non-null    float64
 7   lon              100 non-null    object 
 8   price            100 non-null    object 
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      object 
dtypes: float64(7), object(4)
memory usage: 8.7+ KB
```

In [5]:
```
s.describe()
```

Out[5]:

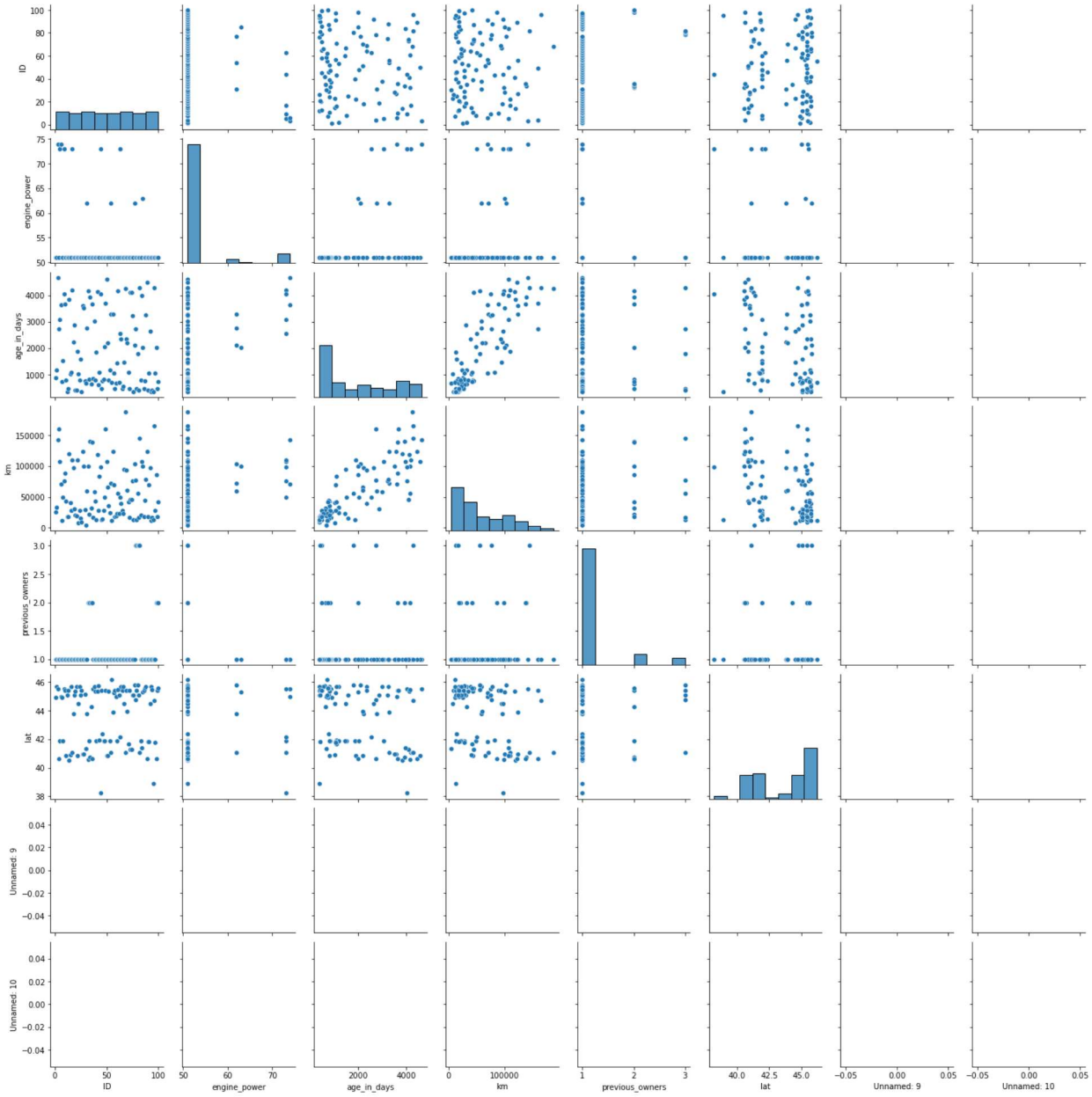
	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	0.0
mean	50.500000	53.010000	1935.300000	58812.180000	1.180000	43.612648	NaN
std	29.011492	6.014284	1414.251278	44728.034639	0.500101	2.083451	NaN
min	1.000000	51.000000	366.000000	4000.000000	1.000000	38.218128	NaN
25%	25.750000	51.000000	723.500000	19781.750000	1.000000	41.744165	NaN
50%	50.500000	51.000000	1446.000000	44032.000000	1.000000	44.831066	NaN
75%	75.250000	51.000000	3265.500000	95075.750000	1.000000	45.396568	NaN
max	100.000000	74.000000	4658.000000	188000.000000	3.000000	46.176498	NaN

In [6]:
```
s.columns
```

Out[6]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10'],
      dtype='object')
```

In [7]:
```
sns.pairplot(s)
```

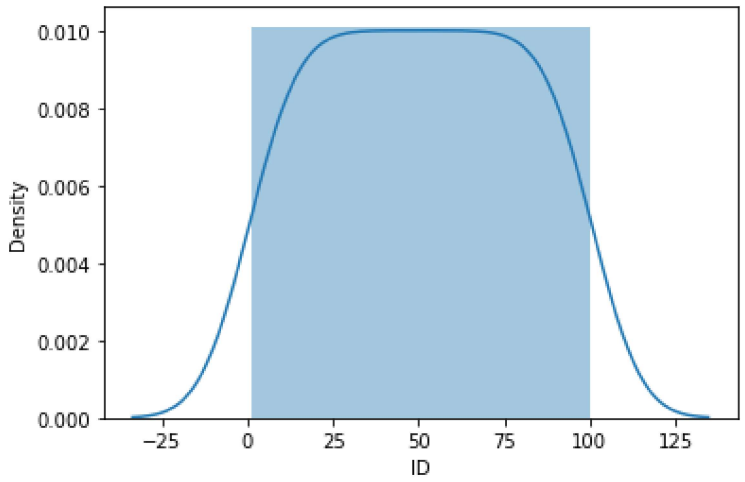
Out[7]:
```
<seaborn.axisgrid.PairGrid at 0x23f7d4b0d00>
```



In [8]:
```
sns.distplot(s['ID'])
```

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[8]: <AxesSubplot:xlabel='ID', ylabel='Density'>



In [9]:
```
s1=s[['ID','engine_power','age_in_days','km', 'previous_owners','lat']]
s
```

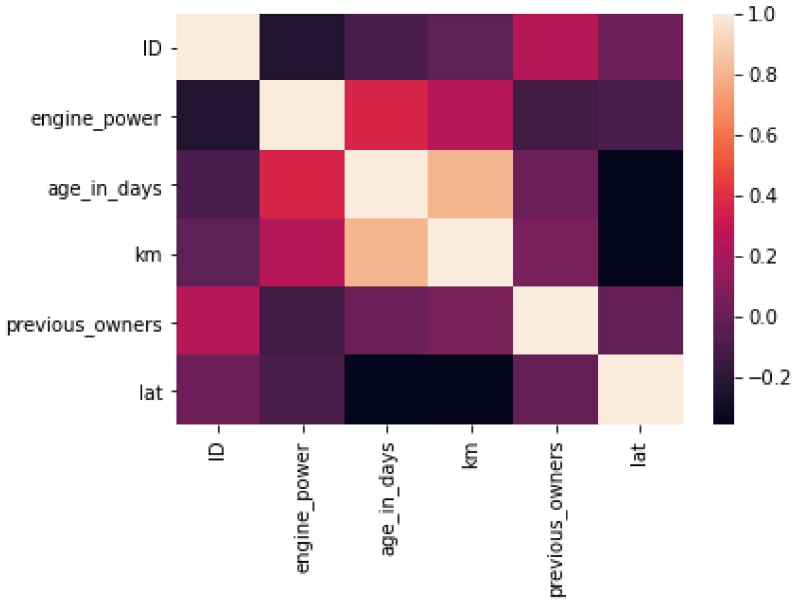
Out[9]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700
...	...	...	...	...	...	...	...	...	...
95	96.0	sport	51.0	4292.0	165600.0	1.0	44.715408	11.30830002	5950
96	97.0	pop	51.0	1066.0	28000.0	1.0	41.769051	12.66281033	8500
97	98.0	sport	51.0	2009.0	86000.0	2.0	40.633171	17.63460922	7800
98	99.0	lounge	51.0	456.0	18592.0	2.0	45.393600	10.48223972	10900
99	100.0	pop	51.0	731.0	41558.0	2.0	45.571220	9.159139633	8790

100 rows × 11 columns

In [10]:
```
sns.heatmap(s1.corr())
```

Out[10]: <AxesSubplot:>



In [11]:
```
x=s1[['ID','engine_power','age_in_days','km', 'previous_owners']]
y=s1['lat']
```

In [12]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3)
```

In [13]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[13]: LinearRegression()

In [14]:
```
lr.intercept_
```

Out[14]: 43.2678987066511

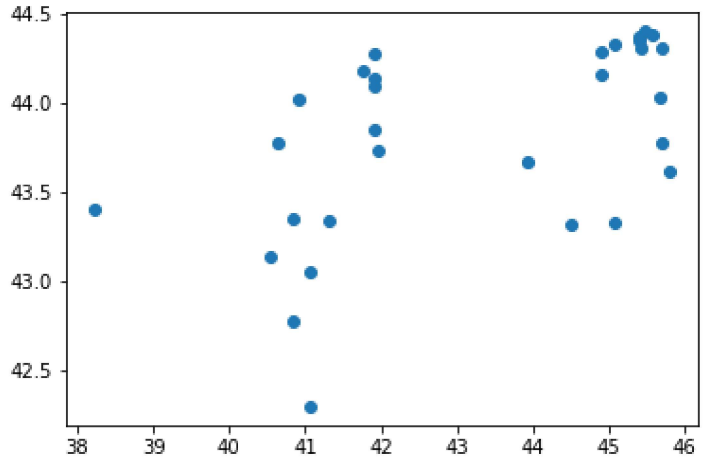
In [15]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
ID	0.000883
engine_power	0.020334
age_in_days	-0.000227
km	-0.000007
previous_owners	0.222827

In [16]:
```
prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[16]:
```
<matplotlib.collections.PathCollection at 0x23f029a6640>
```



In [17]:
```
print(lr.score(x_test,y_test))
```

```
0.12296921970112817
```

In [18]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [19]:
```
from sklearn.linear_model import Ridge,Lasso
```

In [20]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[20]:
```
0.12875739618610493
```

In [21]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
la.score(x_test,y_test)
```

Out[21]:
```
0.13620008884793888
```

In [22]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[22]:
```
ElasticNet()
```

In [23]:
```
print(en.coef_)
```

```
[ 1.61908958e-04  3.35157761e-03 -1.99105789e-04 -7.34835293e-06
  0.00000000e+00]
```

In [24]:
```
print(en.intercept_)
```

```
44.4381561922279
```

In [25]:
```
print(en.predict(x_test))
```

```
[44.1078325  43.47009904 42.70228996 44.20038411 42.39021634 43.0532036
 43.94691425 43.59299185 44.43230767 44.41850201 44.24992843 44.17434836
 44.40956066 42.91581813 44.34452376 43.16362743 44.38482212 44.39570136
 44.16707319 43.37901517 44.13444953 44.37943279 44.20679117 43.7879155
 43.86089837 44.31202571 43.42881631 43.73730341 43.42495091 43.48321938]
```

In [26]:
```
print(en.score(x_test,y_test))
```

```
0.13860005355756688
```

In [27]:
```
from sklearn import metrics
```

In [28]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 1.8175597359755356
```

In [29]:
```
print("Mean squared Error",metrics.mean_squared_error(y_test,prediction))
```

```
Mean squared Error 4.196729812262524
```

In [30]:
```
print("Root Mean squared Error",np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
Root Mean squared Error 2.0485921537149663
```

In [ ]:
```

```