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_VehicleSelection_Dataset (1).cs
s
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Unnamed: 9	Unnamed: 10
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900	NaN	NaN
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800	NaN	NaN
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200	NaN	NaN
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000	NaN	NaN
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	5	NaN	NaN
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonprice	NaN	NaN
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	NO	NaN	NaN
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	1	NaN	NaN
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	1	NaN	NaN

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	Unnamed: 9	Unnamed: 10
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8900	NaN	NaN
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8800	NaN	NaN
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4200	NaN	NaN
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6000	NaN	NaN
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5700	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
95	96.0	sport	51.0	4292.0	165600.0	1.0	44.715408	11.30830002	5950	NaN	NaN
96	97.0	pop	51.0	1066.0	28000.0	1.0	41.769051	12.66281033	8500	NaN	NaN
97	98.0	sport	51.0	2009.0	86000.0	2.0	40.633171	17.63460922	7800	NaN	NaN
98	99.0	lounge	51.0	456.0	18592.0	2.0	45.393600	10.48223972	10900	NaN	NaN
99	100.0	pop	51.0	731.0	41558.0	2.0	45.571220	9.159139633	8790	NaN	NaN

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]:

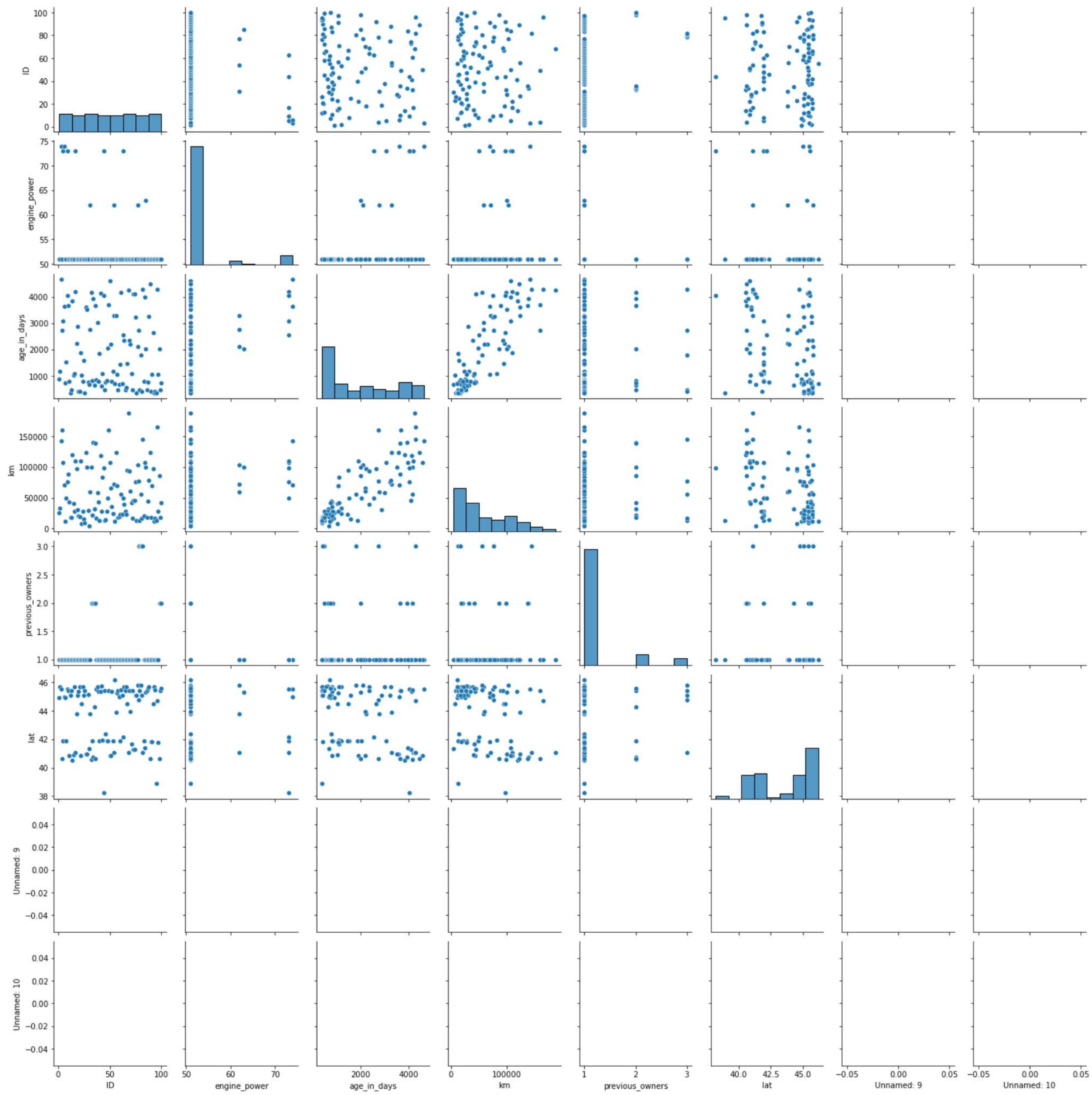
	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
count	100.000000	100.000000	100.000000	100.000000	100.000000	100.000000	0.0
mean	50.500000	53.010000	1935.300000	58812.180000	1.180000	43.612648	NaN
std	29.011492	6.014284	1414.251278	44728.034639	0.500101	2.083451	NaN
min	1.000000	51.000000	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)
```

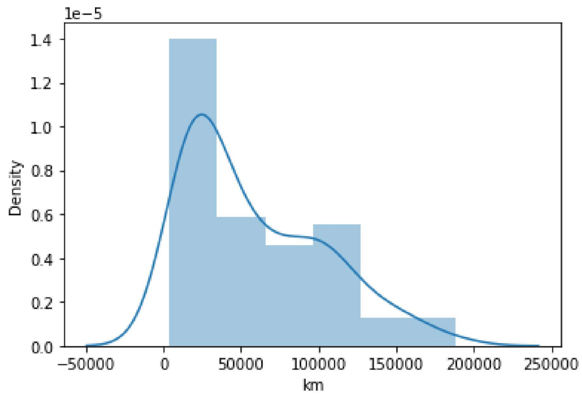
Out[7]: <seaborn.axisgrid.PairGrid at 0x23ce2e66130>



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

```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning: `distplot` is a deprecat
ed function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-lev
el function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

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



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

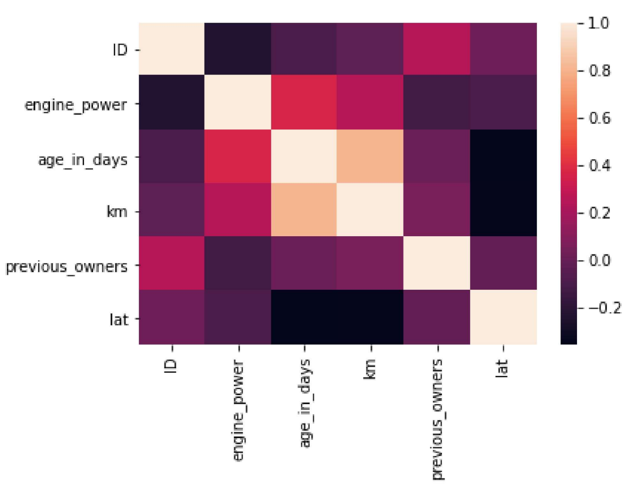
Out[9]:

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

100 rows × 8 columns

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

Out[10]: <AxesSubplot:>



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

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]: 1.8474111129762605e-12

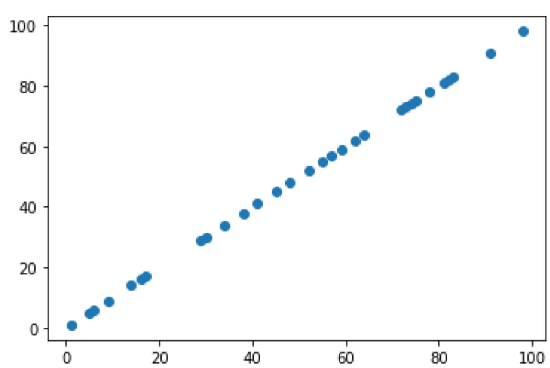
In [15]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[15]:

	Co-efficient
ID	1.000000e+00
engine_power	3.730330e-17
age_in_days	1.086154e-16
km	-3.907403e-17
previous_owners	4.339719e-16
lat	-6.805757e-17
lon	-1.200486e-16
price	2.951709e-17

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

Out[16]: <matplotlib.collections.PathCollection at 0x23cec16a250>



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

1.0

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

In [19]:
```
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
rr.score(x_test,y_test)
```

Out[19]: 0.99999997239915

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

Out[20]: 0.9998637532031336

In [21]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[21]:
```
ElasticNet()
```

In [22]:
```
print(en.coef_)
```

```
[ 9.98825341e-01 -0.00000000e+00 -2.50446829e-06  6.57202154e-08
  0.00000000e+00  0.00000000e+00 -0.00000000e+00  0.00000000e+00]
```

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

```
0.06046292052705127
```

In [24]:
```
print(en.predict(x_test))
```

```
[71.97586074 63.98202867 82.96235786 73.96622689 30.02380803 51.99970231
 14.04229948 41.0097566  45.0072954  61.98736822 72.96740155 90.95549404
  1.05872233 17.03720417 38.01212631 58.99070695 56.99539822  9.04474513
 16.04077678  6.04895648 97.94596676 54.99483709 29.02638871 80.96545491
  5.05391506 77.96705    48.00219295 81.9629211  34.01989028 74.96923415]
```

In [25]:
```
print(en.score(x_test,y_test))
```

```
0.9999986430866786
```

In [26]:
```
from sklearn import metrics
```

In [27]:
```
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
```

```
Mean Absolute Error 1.2809901287861673e-12
```

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

```
Mean squared Error 2.2276939714236516e-24
```

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

```
Root Mean squared Error 1.492546137117259e-12
```

In [ ]: