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

In [2]:
```
df=pd.read_csv(r"C:\\Users\\Admin\\Downloads\\fiat500_VehicleSelection_Dataset
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	l
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.6115598
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.241889
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.417
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.634609
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.495650
...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	leng
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	conc
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null valu
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	fi
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	sear

1549 rows × 11 columns

In [3]:
```
df.head()
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029

In [4]:
```
df.describe()
```

Out[4]:

	ID	engine_power	age_in_days	km	previous_owners	lat	U
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	

In [5]:
```
df.info()
```

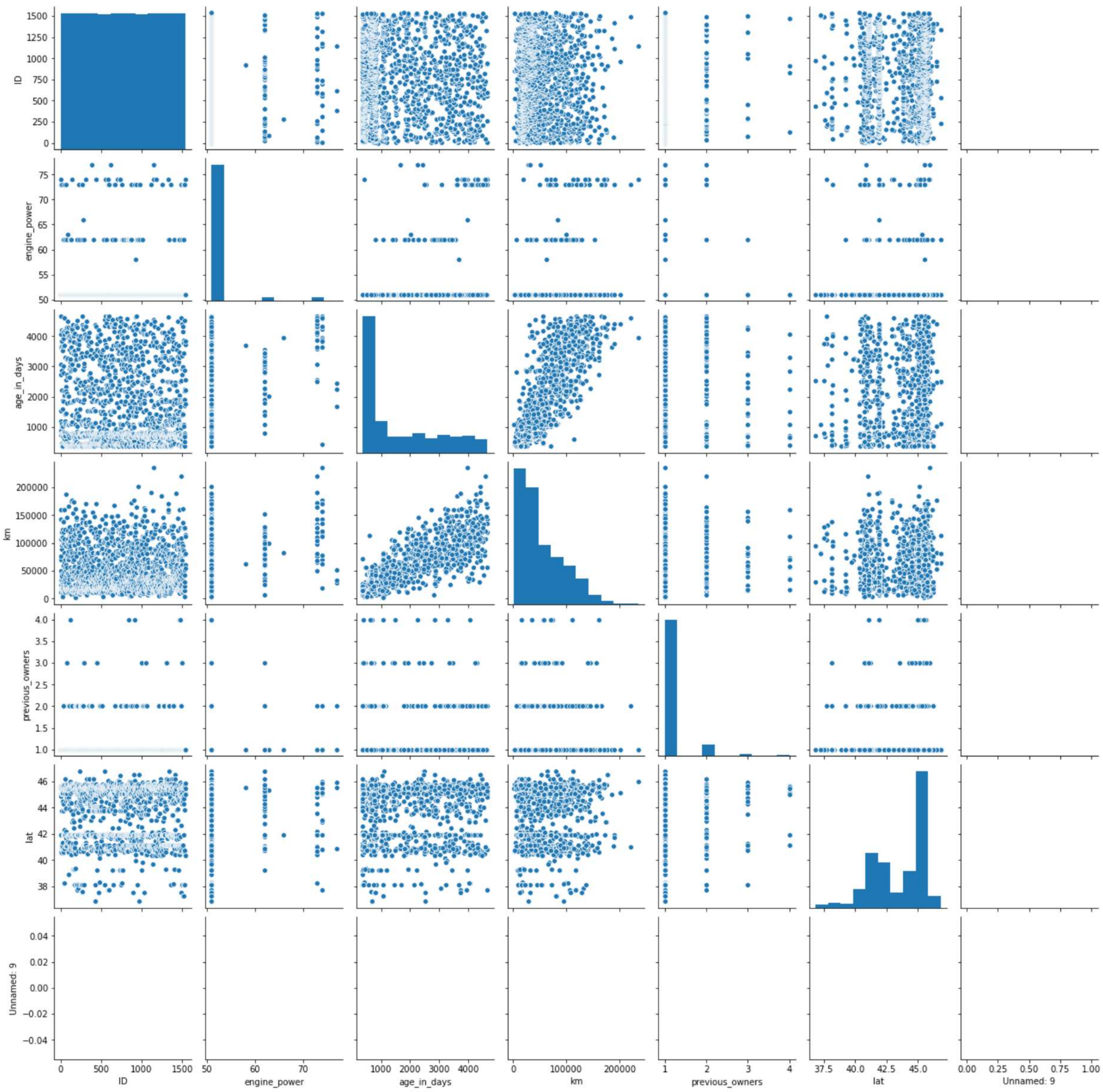
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1549 entries, 0 to 1548
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   float64
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   float64
 3   age_in_days      1538 non-null   float64
 4   km               1538 non-null   float64
 5   previous_owners  1538 non-null   float64
 6   lat              1538 non-null   float64
 7   lon              1549 non-null   object 
 8   price            1549 non-null   object 
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      1 non-null      object 
dtypes: float64(7), object(4)
memory usage: 133.2+ KB
```

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

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

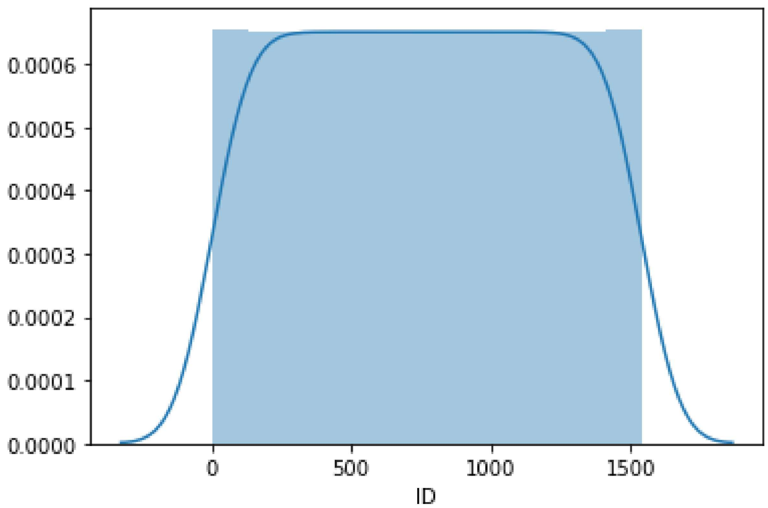
In [7]:
```
sns.pairplot(df)
```

Out[7]: <seaborn.axisgrid.PairGrid at 0x1c73a71ff40>



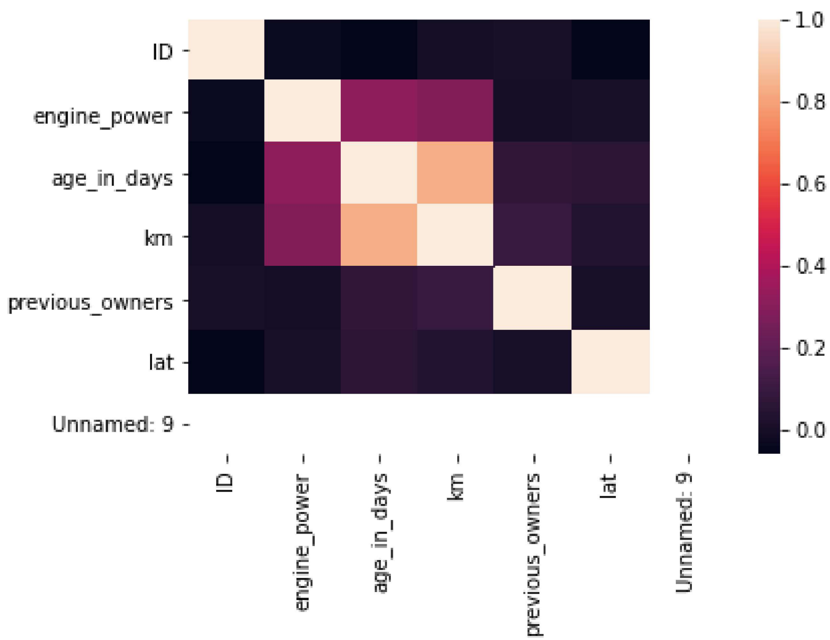
In [8]:
```
sns.distplot(df['ID'])
```

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1c73bdc4b80>



In [9]:
```
sns.heatmap(df.corr())
```

Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x1c73d45f460>



In [10]:
```
df1=df.fillna(value=1)
```

In [11]:
```
x=df1[[ 'age_in_days']]
y=df1['km']
```

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=0.4)
```

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

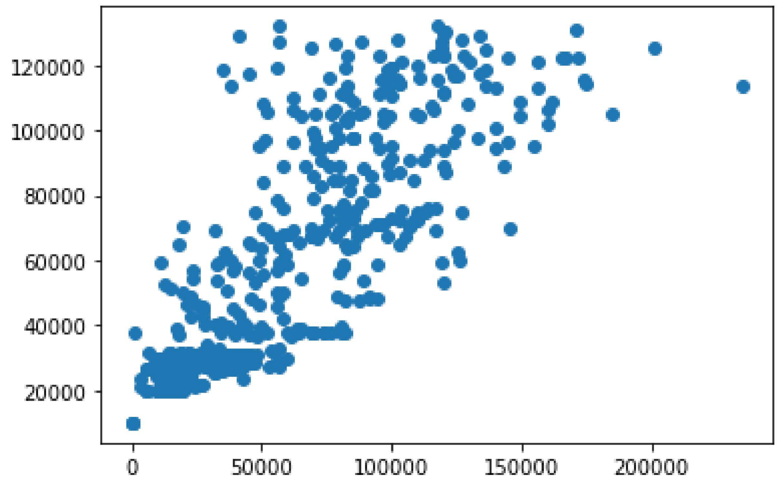
Out[13]: LinearRegression()

In [14]:
```
print(lr.intercept_)
```

10107.496918795696

In [15]:
```
prediction= lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[15]: <matplotlib.collections.PathCollection at 0x1c73df4c130>



In [16]:
```
print(lr.score(x_test,y_test))
```

0.6909047947917997

In [17]:
```
print(lr.score(x_train,y_train))
```

0.7039740472013478

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

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

Out[19]: Ridge(alpha=10)

In [20]:
```
rr.score(x_test,y_test)
```

Out[20]: 0.6909047949621456

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

Out[21]:
```
Lasso(alpha=10)
```

In [22]:
```
la.score(x_test,y_test)
```

Out[22]:
```
0.6909048008397267
```

In [23]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[23]:
```
ElasticNet()
```

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

```
10107.510543888835
```

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

```
[ 43547.71076034  30046.03367764 108047.00147518  30778.68282167
  27638.75791871  26853.77669297  26042.62942638  19684.2814979
 123197.13913193  31589.83008826 127174.37734234 104095.92930563
  22039.22517511 125604.41489086  72225.69154065  27638.75791871
  75417.94852532  38000.51009846  30778.68282167  88134.64438228
 120789.863373    56316.73869902  26042.62942638  29234.88641105
  46739.96774501 110454.27723411 102499.8008133   72225.69154065
  20495.42876449  38000.51009846  28449.90518531  28449.90518531
 116053.80997771  68274.6193711  115242.66271112  30778.68282167
  19684.2814979   88945.79164888  31589.83008826  26853.77669297
  29234.88641105  64271.21511984  84183.57221273  84968.55343847
 100903.67232096  19684.2814979   69007.26851513 119193.73488067
 122412.15790619  21228.07790852  29234.88641105  30778.68282167
  48336.09623735  30046.03367764  19684.2814979   49906.05868882
  19684.2814979   68274.6193711   21228.07790852  21228.07790852
  47551.11501161  53124.48171435  67463.47210451  85779.70070507
  28449.90518531  69818.41578172  26042.62942638  54720.61020669
  65867.34361217  58724.01445795  36404.38160612  19684.2814979
  74632.96729958 125604.41489086  20495.42876449  22039.22517511
```

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

```
0.6909048030066923
```

Evaluation

In [27]:
```
from sklearn import metrics
print("Mean Absolute Error",metrics.mean_absolute_error(y_test,prediction))
print("Mean Squared Error",metrics.mean_squared_error(y_test,prediction))
print("Root Mean Squared Error:",np.sqrt(metrics.mean_squared_error(y_test,pred
```

```
Mean Absolute Error 15844.679730165151
Mean Squared Error 507836061.28438807
Root Mean Squared Error: 22535.218243549098
```

In [ ]: