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("fiat.csv")
df
```

Out[2]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	p
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5
...	...	...	...	...	...	...	...	...	
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length	
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat	lonp
1546	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Null values	
1547	NaN	NaN	NaN	NaN	NaN	NaN	NaN	find	
1548	NaN	NaN	NaN	NaN	NaN	NaN	NaN	search	

1549 rows × 11 columns

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

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	U
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	

DATA CLEANING AND DATA PREPROCESSING

In [4]:
```
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 [5]:
```
df.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: g
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	0.0
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	NaN
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	NaN
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	NaN
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	NaN
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	NaN
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	NaN
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	NaN

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]:
```
df1=df[0:1500]
```

In [8]:
```
df1=df1.dropna(axis=1)
df1
```

Out[8]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	p
0	1.0	lounge	51.0	882.0	25000.0	1.0	44.907242	8.611559868	8
1	2.0	pop	51.0	1186.0	32500.0	1.0	45.666359	12.24188995	8

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	p
2	3.0	sport	74.0	4658.0	142228.0	1.0	45.503300	11.41784	4
3	4.0	lounge	51.0	2739.0	160000.0	1.0	40.633171	17.63460922	6
4	5.0	pop	73.0	3074.0	106880.0	1.0	41.903221	12.49565029	5
...	...	...	...	...	...	...	...	...	
1495	1496.0	pop	62.0	3347.0	80000.0	3.0	44.283878	11.88813972	7
1496	1497.0	pop	51.0	1461.0	91055.0	3.0	44.508839	11.46907997	7
1497	1498.0	lounge	51.0	397.0	15840.0	3.0	38.122070	13.36112022	10
1498	1499.0	sport	51.0	1400.0	60000.0	1.0	45.802021	9.187789917	10
1499	1500.0	pop	51.0	1066.0	53100.0	1.0	38.122070	13.36112022	8

1500 rows × 9 columns

In [9]:
```
df1.columns
```

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

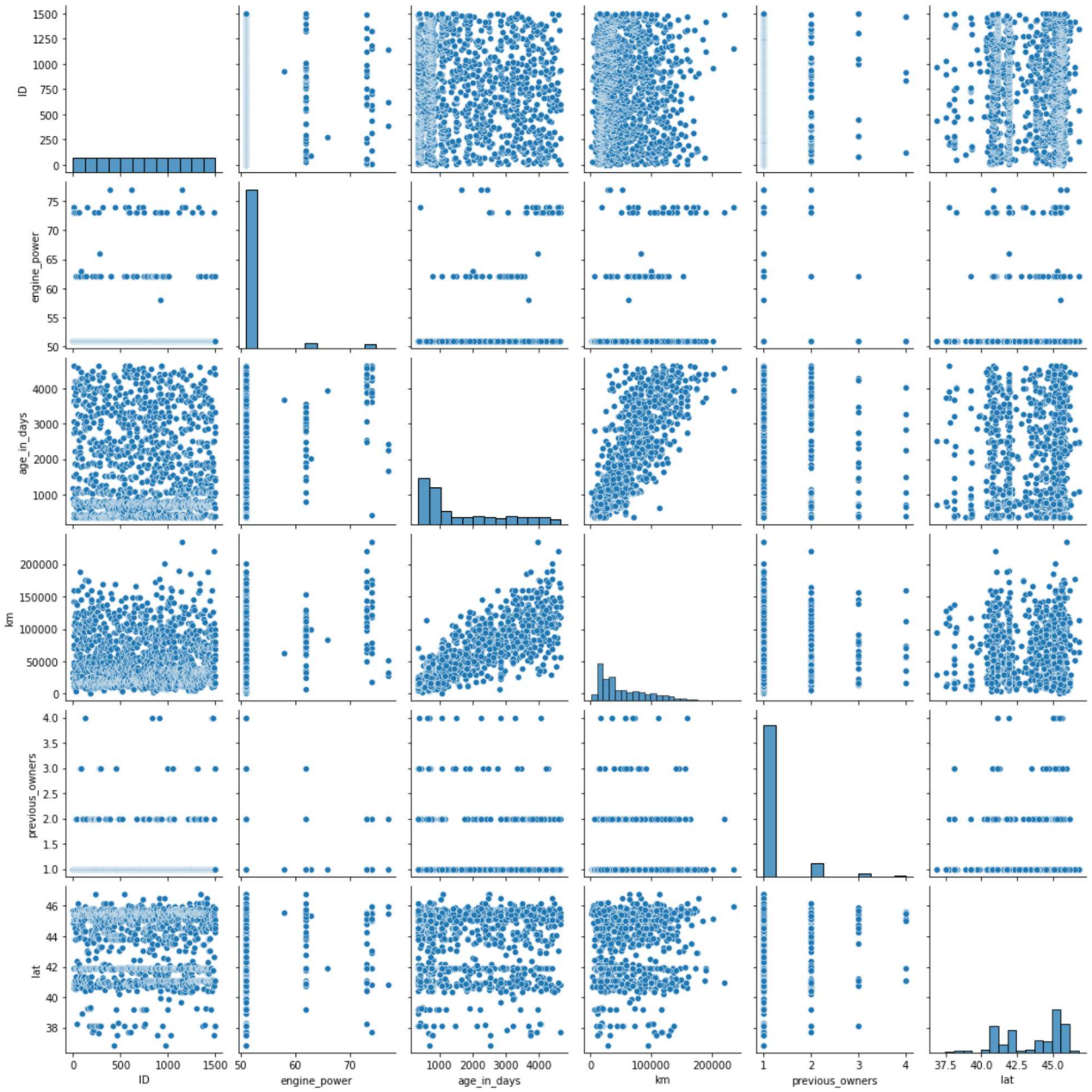
In [ ]:
```

```

In [10]:
```
df1=df1[['ID',  'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat']]
```

EDA AND VISUALIZATION

In [11]:
```
sns.pairplot(df1)
```

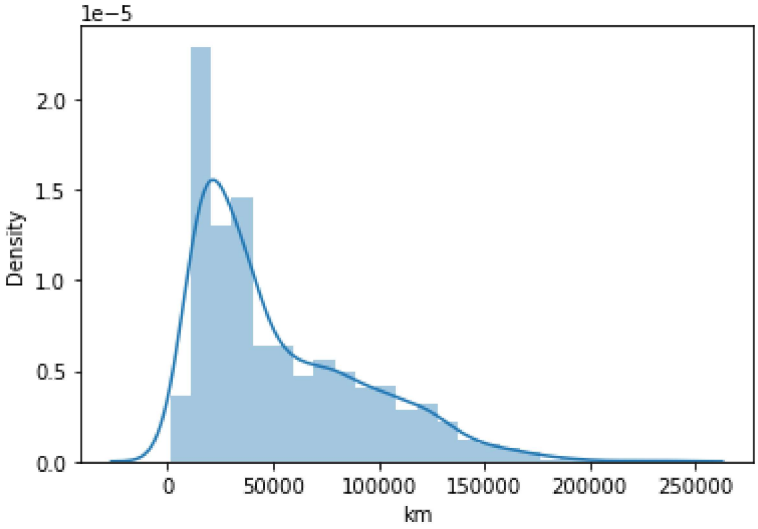
Out[11]:
```
<seaborn.axisgrid.PairGrid at 0x1ea8831fdc0>
```



In [12]:
```
sns.distplot(df1['km'])
```

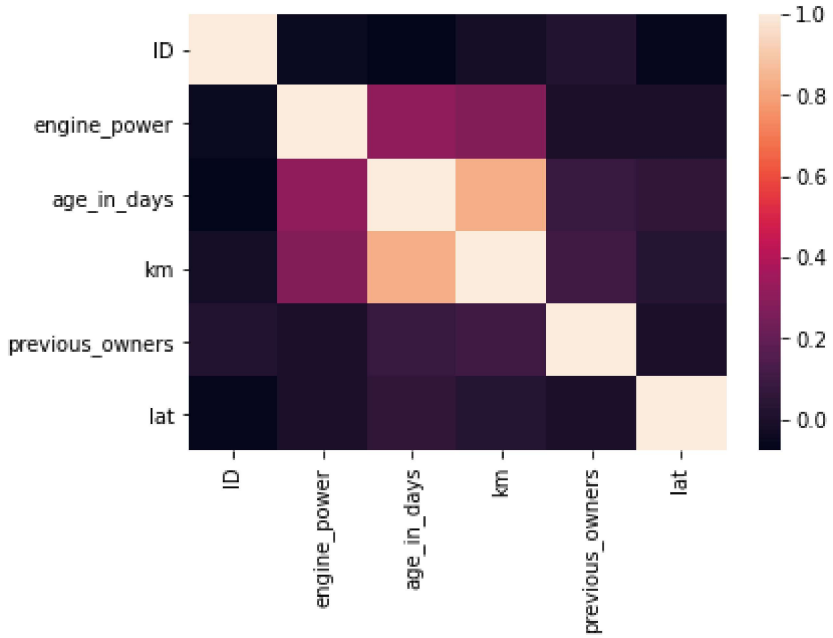
```
C:\ProgramData\Anaconda3\lib\site-packages\seaborn\distributions.py:2557: FutureWarning:
`distplot` is a deprecated function and will be removed in a future version. Please adap
t your code to use either `displot` (a figure-level function with similar flexibility) o
r `histplot` (an axes-level function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

Out[12]: <AxesSubplot:xlabel='km', ylabel='Density'>



In [13]:
```
sns.heatmap(df1.corr())
```

Out[13]: <AxesSubplot:>



TO TRAIN THE MODEL AND MODEL BULDING

In [14]:
```
df1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 6 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1500 non-null   float64
 1   engine_power     1500 non-null   float64
 2   age_in_days      1500 non-null   float64
 3   km               1500 non-null   float64
 4   previous_owners  1500 non-null   float64
 5   lat              1500 non-null   float64
```

```
dtypes: float64(6)
memory usage: 70.4 KB
```

In [15]:
```
x=df1[['ID', 'age_in_days', 'km','previous_owners',
       'lat']]
y=df1['engine_power']
```

In [16]:
```
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 [17]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[17]:
```
LinearRegression()
```

In [18]:
```
lr.intercept_
```

Out[18]:
```
51.60217922847731
```

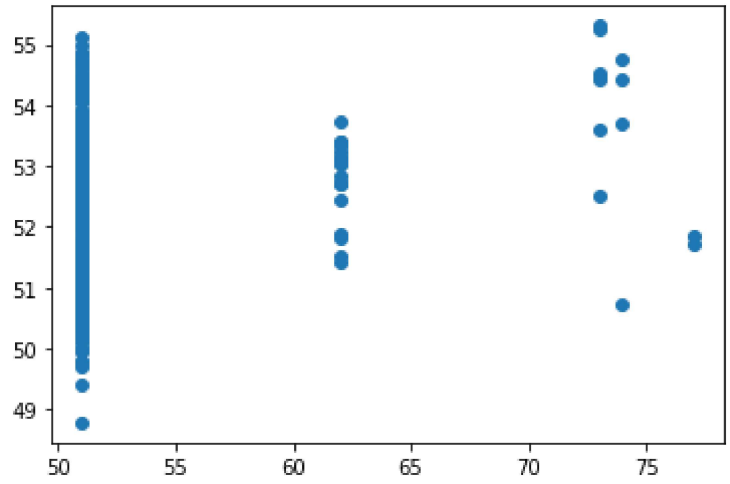
In [19]:
```
coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[19]:

	Co-efficient
ID	-0.000350
age_in_days	0.000495
km	0.000016
previous_owners	-0.526662
lat	-0.014408

In [20]:
```
prediction =lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[20]:
```
<matplotlib.collections.PathCollection at 0x1ea8a9f43a0>
```



ACCURACY

In [21]:
```
lr.score(x_test,y_test)
```

Out[21]: 0.07764114555086932

In [22]:
```
lr.score(x_train,y_train)
```

Out[22]: 0.10784251892262575

In [23]:
```
from sklearn.linear_model import Ridge,Lasso
rr=Ridge(alpha=10)
rr.fit(x_train,y_train)
```

Out[23]: Ridge(alpha=10)

In [24]:
```
rr.score(x_train,y_train)
```

Out[24]: 0.10783228490409225

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

Out[25]: 0.07808516360156248

In [26]:
```
la=Lasso(alpha=10)
la.fit(x_train,y_train)
```

Out[26]: Lasso(alpha=10)

In [27]:
```
la.score(x_train,y_train)
```

Out[27]: 0.10452162198614401

In [28]:
```
la.score(x_test,y_test)
```

Out[28]: 0.08166043800156653

In [29]:
```
from sklearn.linear_model import ElasticNet
en=ElasticNet()
en.fit(x_train,y_train)
```

Out[29]: ElasticNet()

In [30]:
```
en.coef_
```

Out[30]:
```
array([-3.63018942e-04,  4.91031568e-04,  1.53966734e-05, -0.00000000e+00,
       -0.00000000e+00])
```

In [31]:
```
en.intercept_
```

Out[31]: 50.434317399244414

In [32]:
```
prediction=en.predict(x_test)
```

In [33]:
```
en.score(x_test,y_test)
```

Out[33]: 0.08212949691371874

In [34]:
```
from sklearn import metrics
print(metrics.mean_absolute_error(y_test,prediction))
print(metrics.mean_squared_error(y_test,prediction))
print(np.sqrt(metrics.mean_squared_error(y_test,prediction)))
```

```
1.8253996025393957
17.31164506918215
4.160726507376103
```