Problem Statement:

A real estate agent want to help to predict the house price for regions in USA.He gave us the dataset to work on to use Linear Regression modelCreate a Model that helps him to estimate of what the house would sell for

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",low_memory=False)[0:1500]
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
...	...	...	...	...	...	...	...	...	
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 × 11 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1500 non-null   float64
 1   model            1500 non-null   object 
 2   engine_power     1500 non-null   float64
 3   age_in_days      1500 non-null   float64
 4   km               1500 non-null   float64
 5   previous_owners  1500 non-null   float64
 6   lat              1500 non-null   float64
 7   lon              1500 non-null   object 
 8   price            1500 non-null   object 
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      object 
dtypes: float64(7), object(4)
memory usage: 129.0+ KB
```

In [4]:

```
df.head()
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Ur
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 Pre-Processing

In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1500 entries, 0 to 1499
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1500 non-null   float64
 1   model            1500 non-null   object 
 2   engine_power     1500 non-null   float64
 3   age_in_days      1500 non-null   float64
 4   km               1500 non-null   float64
 5   previous_owners  1500 non-null   float64
 6   lat              1500 non-null   float64
 7   lon              1500 non-null   object 
 8   price            1500 non-null   object 
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      object 
dtypes: float64(7), object(4)
memory usage: 129.0+ KB
```

In [6]:

```
df.describe()
```

Out[6]:

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
count	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	1500.000000	0.0
mean	750.500000	51.875333	1641.629333	53074.900000	1.126667	43.545904	NaN
std	433.157015	3.911606	1288.091104	39955.013731	0.421197	2.112907	NaN
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	NaN

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
25%	375.750000	51.000000	670.000000	20000.000000	1.000000	41.802990	NaN
50%	750.500000	51.000000	1035.000000	38720.000000	1.000000	44.360376	NaN
75%	1125.250000	51.000000	2616.000000	78170.250000	1.000000	45.467960	NaN
max	1500.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	NaN

In [7]:

```
df.dropna(axis='columns')
```

Out[7]:

	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
...	...	...	...	...	...	...	...	...	
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 [8]:

```
a = df.dropna(axis='columns')
a.columns
```

Out[8]:

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

In [9]:

```
df.columns
```

Out[9]:

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

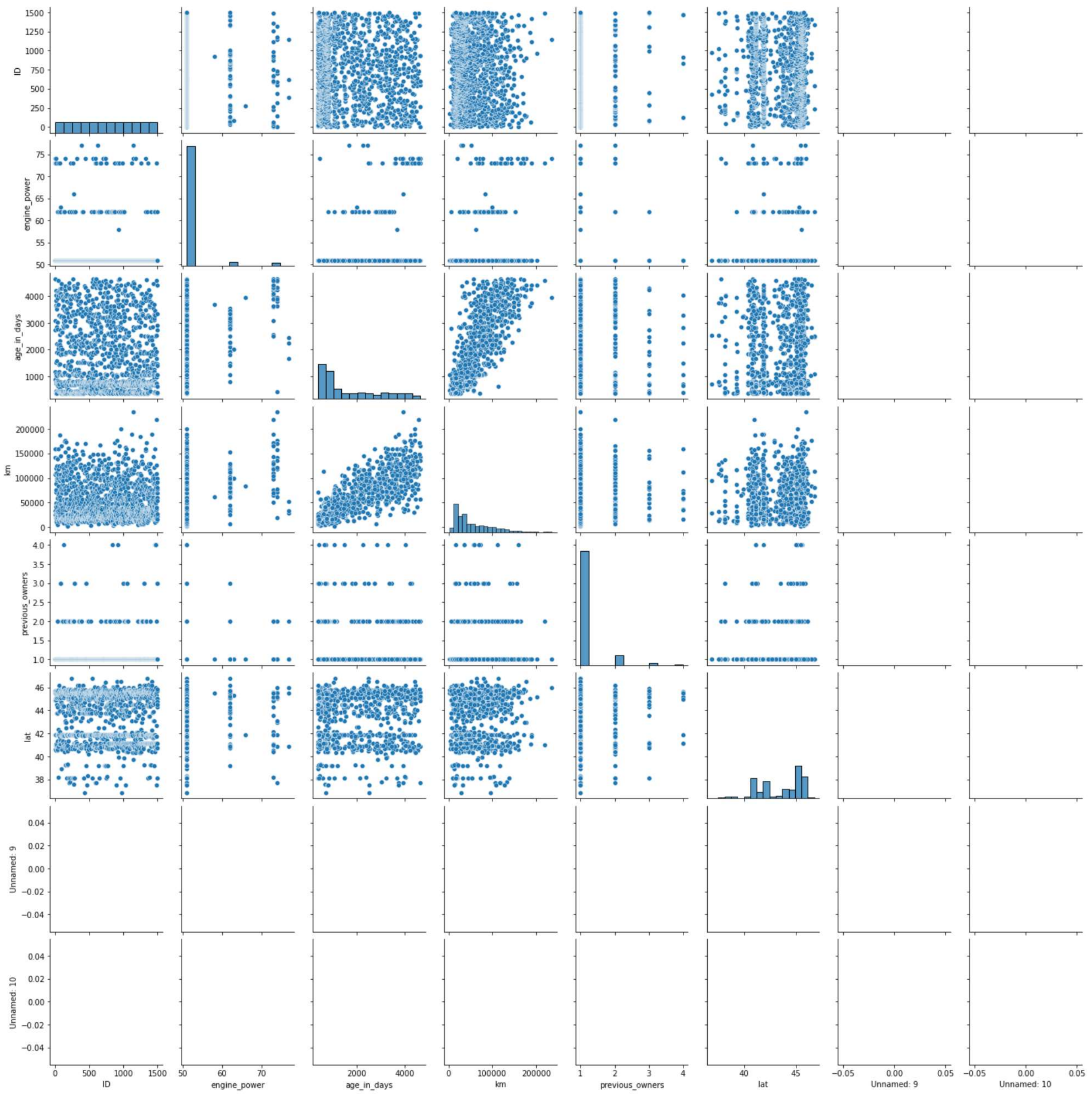
EDA and VISUALIZATION

In [10]:

```
sns.pairplot(df)
```

Out[10]:

```
<seaborn.axisgrid.PairGrid at 0x27cbd3a5490>
```



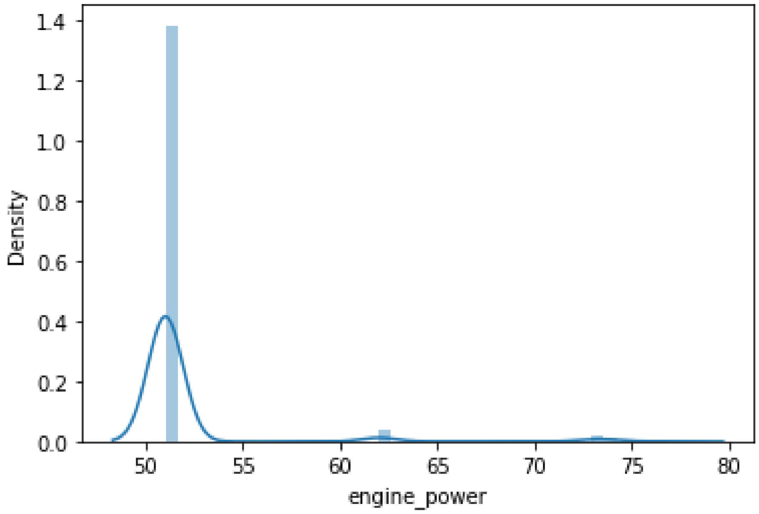
In [11]:

```
sns.distplot(df['engine_power'])
```

```
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[11]:

```
<AxesSubplot:xlabel='engine_power', ylabel='Density'>
```



In [12]:

```
df1=df[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price']]
```

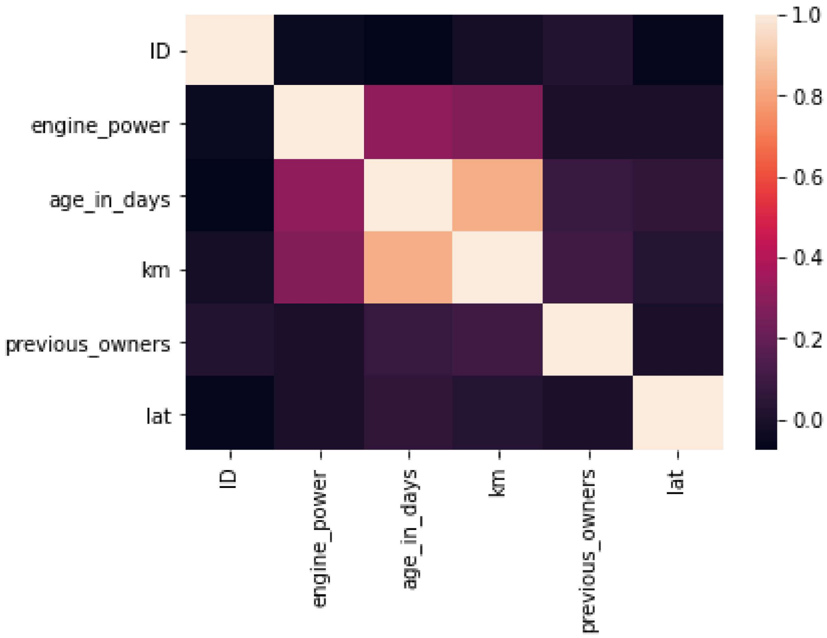
Plot Using Heat Map

In [13]:

```
sns.heatmap(df1.corr())
```

Out[13]:

```
<AxesSubplot:>
```



To Train The Model-Model Building

we are going to train Linera Regression Model;We need to split out data into two variables x and y where x is independent variable(input) and y is dependent on x(output) we could ignore address column as it required for our model

In [14]:

```
x=df1[['ID', 'previous_owners','lat']]
y=df1['engine_power']
```

To Split my dataset into training and test data

In [15]:

```
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 [16]:

```
from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[16]:

```
LinearRegression()
```

In [17]:

```
lr.intercept_
```

Out[17]:

```
51.12162591438447
```

In [18]:

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

Out[18]:

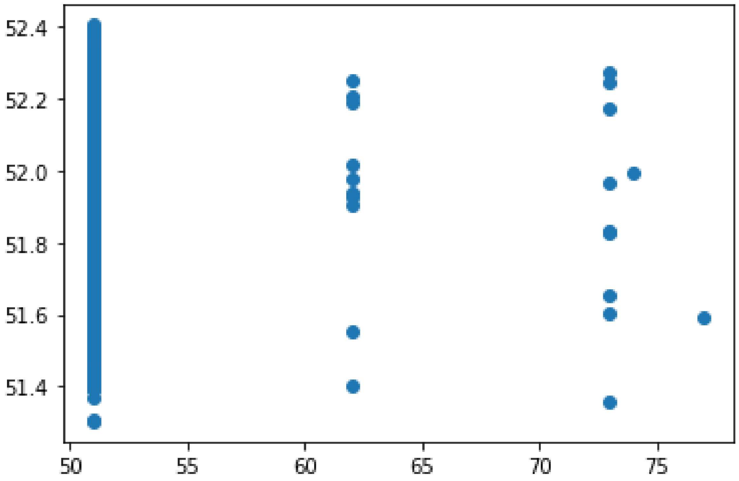
	Co-efficient
ID	-0.000597
previous_owners	-0.033231
lat	0.029109

In [19]:

```
prediction = lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[19]:

```
<matplotlib.collections.PathCollection at 0x27cc19dc970>
```



In [20]:

```
lr.score(x_test,y_test)
```

Out[20]:

```
-0.005933017337634627
```