Problem Statement

Linear Regression

Import Libraries

In [2]:
```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [3]:
```python
a=pd.read_csv("Ren.csv")
a
```

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
...	...	...	...	...	...	...	...	...
1544	NaN	NaN	NaN	NaN	NaN	NaN	NaN	length
1545	NaN	NaN	NaN	NaN	NaN	NaN	NaN	concat
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

To display top 10 rows

In [4]:
```python
c=a.head(15)
c
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pr
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

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	pr
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
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.68227005	7
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.611559868	10
7	8.0	lounge	51.0	1521.0	49076.0	1.0	41.903221	12.49565029	9
8	9.0	sport	73.0	4049.0	76000.0	1.0	45.548000	11.54946995	5
9	10.0	sport	51.0	3653.0	89000.0	1.0	45.438301	10.99170017	6
10	11.0	pop	51.0	790.0	43286.0	1.0	40.871429	14.43896008	8
11	12.0	lounge	51.0	366.0	17500.0	1.0	45.069679	7.704919815	10
12	13.0	lounge	51.0	456.0	18450.0	1.0	45.426571	11.78812981	9
13	14.0	pop	51.0	3835.0	120000.0	1.0	40.531590	17.43615913	4
14	15.0	lounge	51.0	1035.0	40500.0	1.0	40.911362	14.21119976	9

To find Missing values

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   ID               15 non-null     float64
 1   model            15 non-null     object
 2   engine_power     15 non-null     float64
 3   age_in_days      15 non-null     float64
 4   km               15 non-null     float64
 5   previous_owners  15 non-null     float64
 6   lat              15 non-null     float64
 7   lon              15 non-null     object
 8   price            15 non-null     object
 9   Unnamed: 9       0 non-null      float64
 10  Unnamed: 10      0 non-null      object
dtypes: float64(7), object(4)
memory usage: 1.4+ KB
```

To display summary of statistics

In [6]:
```
a.describe()
```

Out[6]:

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnar
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	

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnar
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	

To display column heading

In [7]:
```
a.columns
```

Out[7]:
```
Index(['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10'],
      dtype='object')
```

Pairplot

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

Out[8]:

	lon	price
0	8.611559868	8900
1	12.24188995	8800
2	11.41784	4200
3	17.63460922	6000
4	12.49565029	5700
...	...	...
1544	length	5
1545	concat	lonprice
1546	Null values	NO
1547	find	1
1548	search	1

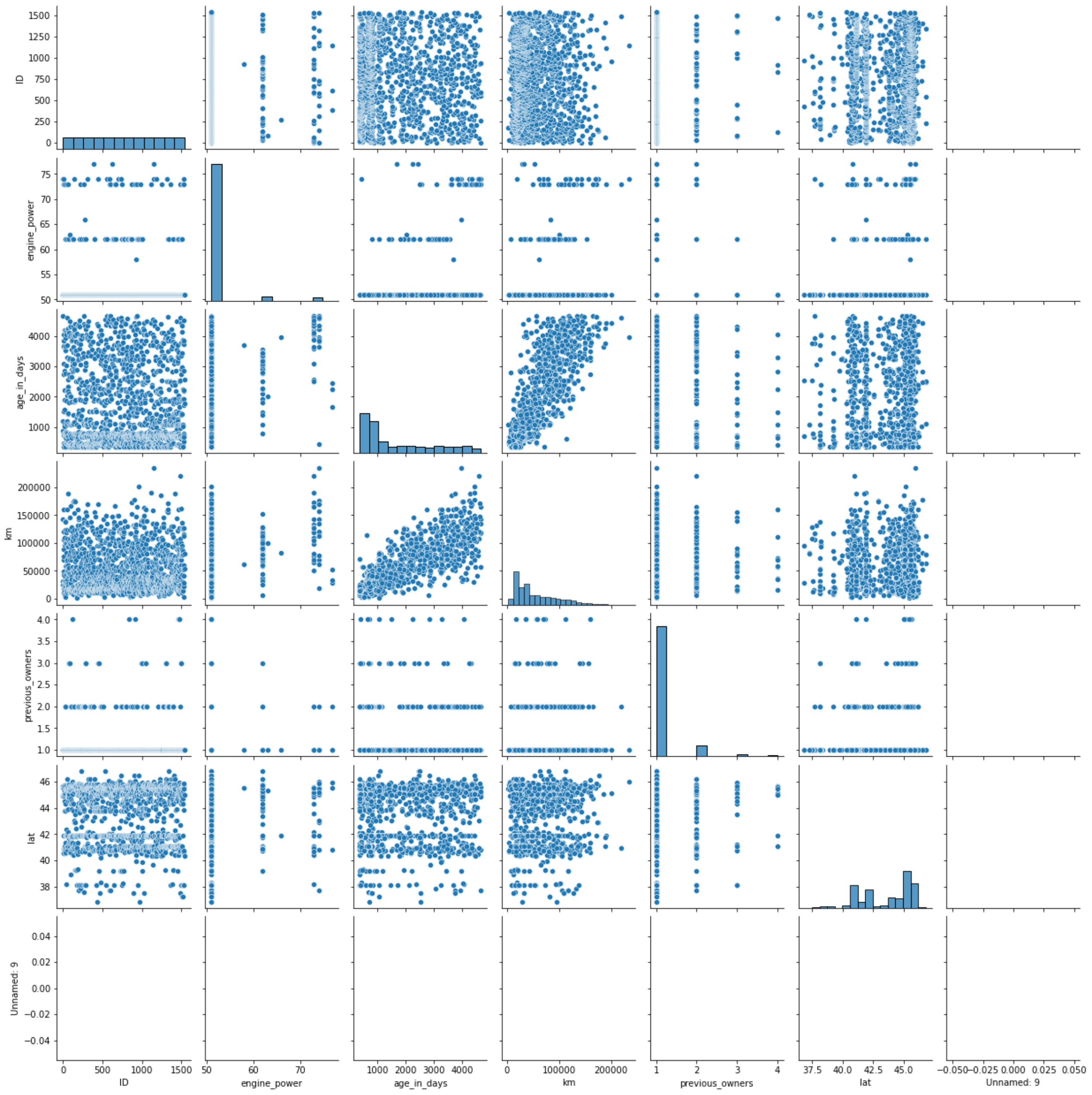
1549 rows × 2 columns

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

Out[9]:
```
Index(['lon', 'price'], dtype='object')
```

In [10]:
```
sns.pairplot(a)
```

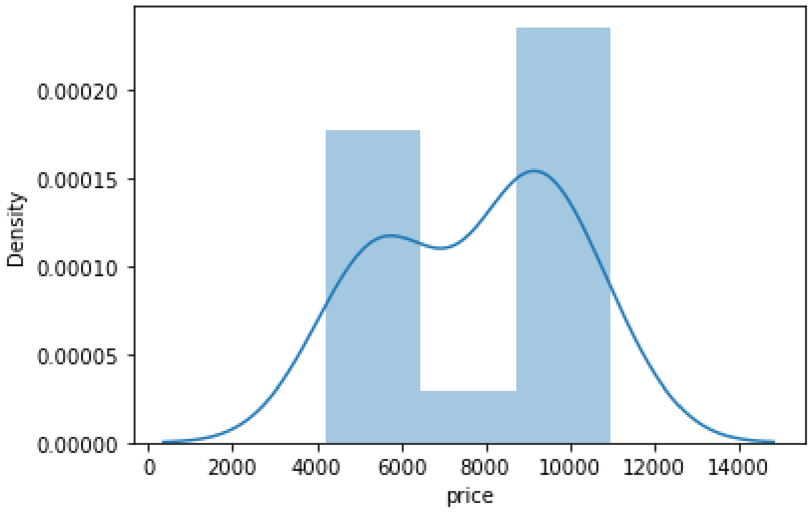
```
Out[10]: <seaborn.axisgrid.PairGrid at 0x26035f81d30>
```



Distribution Plot

```
In [12]: sns.distplot(c['price'])
```

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

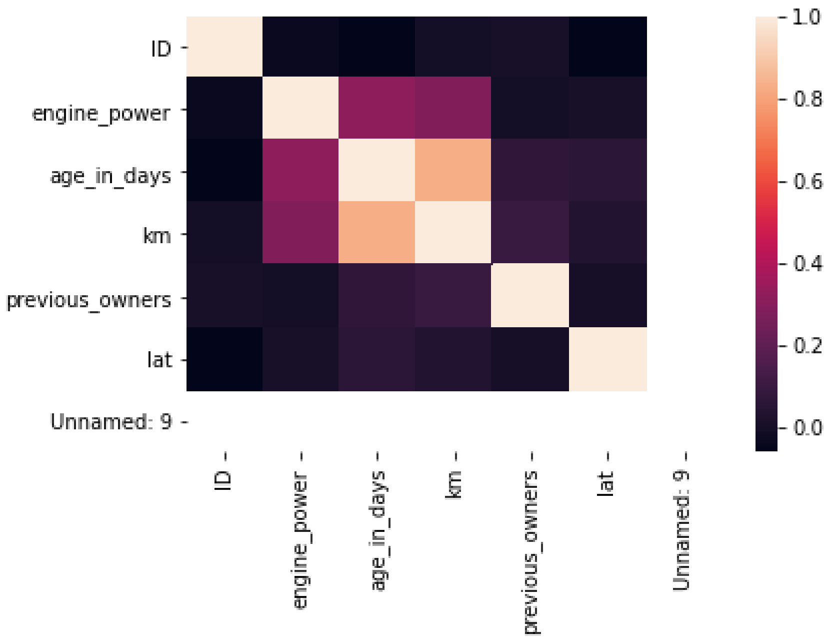


Correlation

Train the model - Model Building

In [14]:
```
b=a[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
       'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10']]
sns.heatmap(b.corr())
```

Out[14]:
```
<AxesSubplot:>
```



In [20]:
```
g=c[['price']]
h=c['price']
```

To split dataset into training end test

In [21]:
```
from sklearn.model_selection import train_test_split
g_train,g_test,h_train,h_test=train_test_split(g,h,test_size=0.6)
```

To run the model

In [22]:
```
from sklearn.linear_model import LinearRegression
```

In [23]:
```
lr=LinearRegression()
lr.fit(g_train,h_train)
```

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

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

```
9.094947017729282e-13
```

Coeffecient

In [25]:
```
coeff=pd.DataFrame(lr.coef_,g.columns,columns=['Co-effecient'])
coeff
```

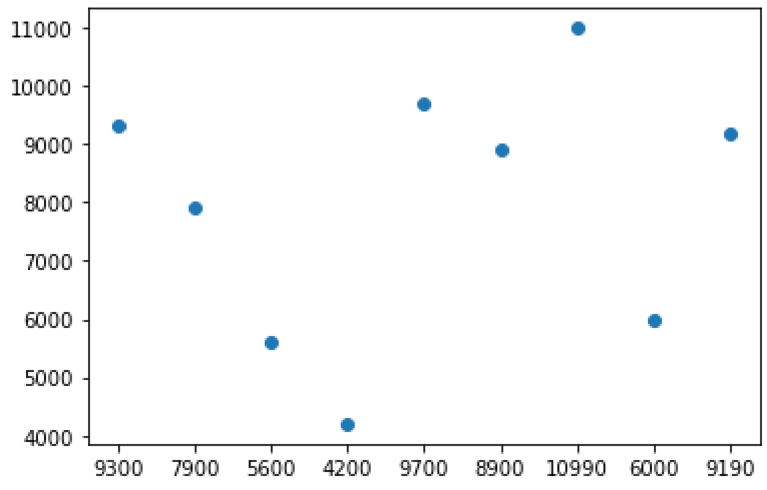
Out[25]:

	Co-effecient
price	1.0

Best Fit line

In [26]:
```
prediction=lr.predict(g_test)
plt.scatter(h_test,prediction)
```

Out[26]:
```
<matplotlib.collections.PathCollection at 0x2603a024d60>
```



To find score

In [27]:
```
print(lr.score(g_test,h_test))
```

```
1.0
```

Import Lasso and ridge

In [28]:
```
from sklearn.linear_model import Ridge,Lasso
```

Ridge

In [29]:
```
ri=Ridge(alpha=5)
ri.fit(g_train,h_train)
```

Out[29]:
```
Ridge(alpha=5)
```

In [30]:
```
ri.score(g_test,h_test)
```

Out[30]:
```
0.9999999999999644
```

In [31]:
```
ri.score(g_train,h_train)
```

Out[31]:
```
0.999999999999966
```

Lasso

In [32]:
```
l=Lasso(alpha=6)
l.fit(g_train,h_train)
```

Out[32]:
```
Lasso(alpha=6)
```

In [33]:
```
l.score(g_test,h_test)
```

Out[33]:
```
0.9999999999981501
```

In [34]:
```
ri.score(g_train,h_train)
```

Out[34]:
```
0.999999999999966
```