

Problem statement

Data collection

Importing libraries

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

Importing dataset

```
In [2]: data=pd.read_csv(r"C:\Users\user\Downloads\fiat500_VehicleSelection_Dataset (2).csv")
data
```

```
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	long
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



head

```
In [3]: # to display first 8 dataset values
da=data.head(8)
da
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price	Unnamed: 9
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	
5	6.0	pop	74.0	3623.0	70225.0	1.0	45.000702	7.68227005	7900	
6	7.0	lounge	51.0	731.0	11600.0	1.0	44.907242	8.611559868	10750	
7	8.0	lounge	51.0	1521.0	49076.0	1.0	41.903221	12.49565029	9190	

info

In [4]:

```
# to identify missing values
data.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

describe

In [5]:

```
# to display summary of the dataset
data.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
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

	ID	engine_power	age_in_days	km	previous_owners	lat	Unnamed: 9
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

columns

```
In [6]: # to display headings of the dataset
data.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]: a=data.dropna(axis=1,how='any')
a
b=a.head(8)
b
```

```
Out[7]:
```

	lon	price
0	8.611559868	8900
1	12.24188995	8800
2	11.41784	4200
3	17.63460922	6000
4	12.49565029	5700
5	7.68227005	7900
6	8.611559868	10750
7	12.49565029	9190

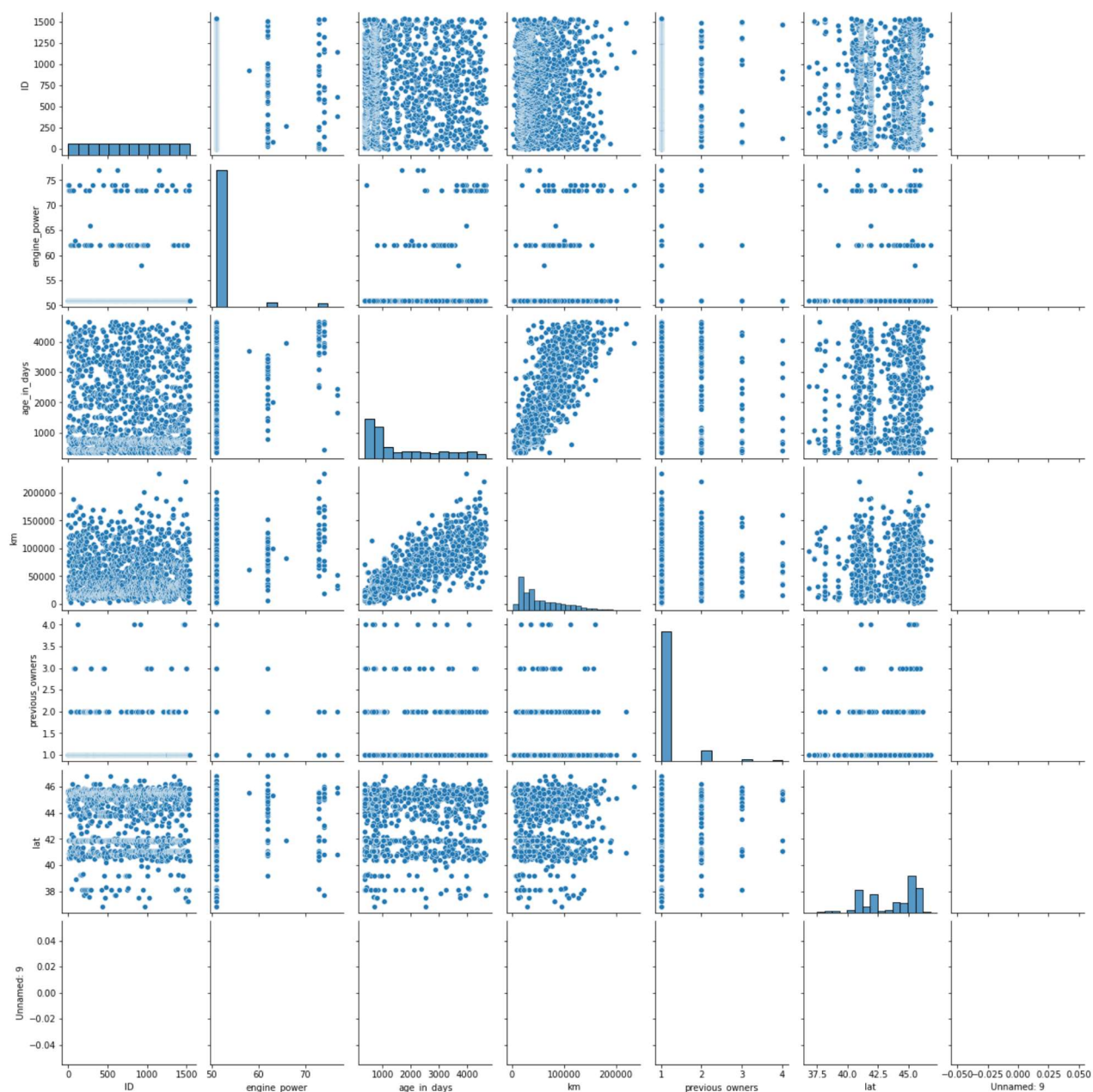
```
In [8]: a.columns
```

```
Out[8]: Index(['lon', 'price'], dtype='object')
```

EDA and Visualization

```
In [9]: sns.pairplot(data)
```

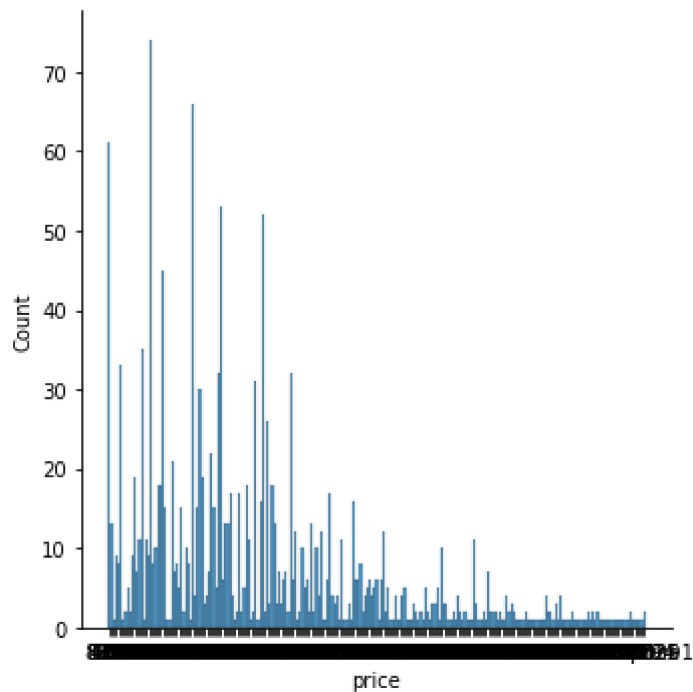
Out[9]: <seaborn.axisgrid.PairGrid at 0x234ac1c6880>



distribution plot

```
In [10]: sns.displot(a["price"])
```

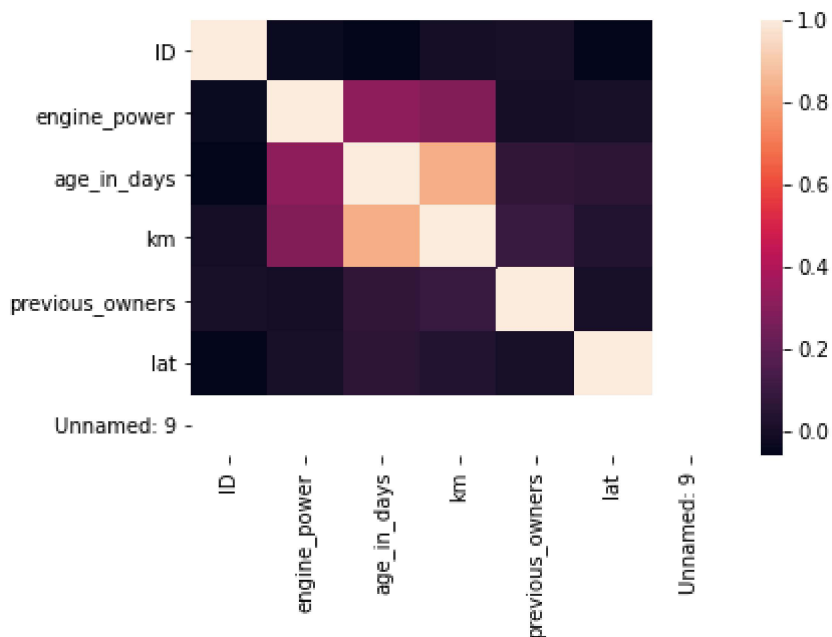
Out[10]: <seaborn.axisgrid.FacetGrid at 0x234b2690880>



correlation

```
In [11]: dat=data[['ID', 'model', 'engine_power', 'age_in_days', 'km', 'previous_owners',
                  'lat', 'lon', 'price', 'Unnamed: 9', 'Unnamed: 10']]
          sns.heatmap(dat.corr())
```

Out[11]: <AxesSubplot:>



To train the model-Model Building

```
In [12]: x=b[['price']]
          y=b[['price']]
```

```
In [13]: # to split my dataset into training and test data
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 [14]: from sklearn.linear_model import LinearRegression
lr= LinearRegression()
lr.fit(x_train,y_train)
```

Out[14]: LinearRegression()

```
In [15]: print(lr.intercept_)
```

-1.8189894035458565e-12

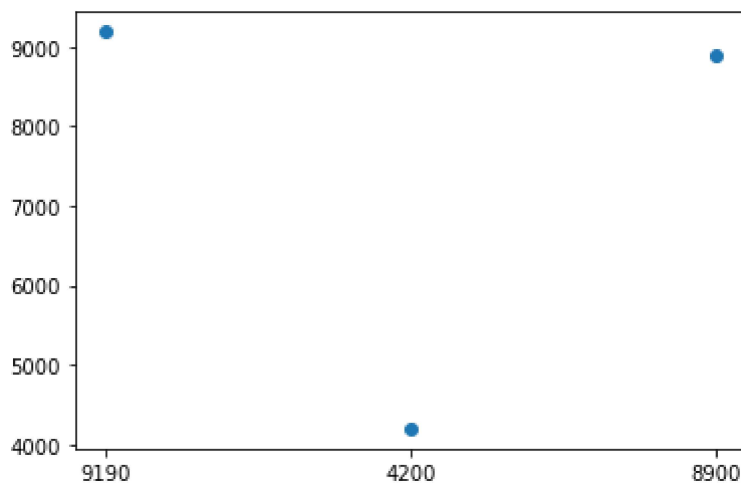
```
In [16]: coeff=pd.DataFrame(lr.coef_,x.columns,columns=['Co-efficient'])
coeff
```

Out[16]:

	Co-efficient
price	1.0

```
In [17]: prediction=lr.predict(x_test)
plt.scatter(y_test,prediction)
```

Out[17]: <matplotlib.collections.PathCollection at 0x234b5188460>



```
In [18]: print(lr.score(x_test,y_test))
```

1.0