Data Analysis

In [1]:
```
import pandas as r
d=r.read_csv("/home/placement/Downloads/fiat500") #reading the file into the jupyter
```

In [2]:
```
#method is used to prints information about the DataFrame
d.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1538 entries, 0 to 1537
Data columns (total 9 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   ID               1538 non-null   int64  
 1   model            1538 non-null   object 
 2   engine_power     1538 non-null   int64  
 3   age_in_days      1538 non-null   int64  
 4   km               1538 non-null   int64  
 5   previous_owners  1538 non-null   int64  
 6   lat              1538 non-null   float64
 7   lon              1538 non-null   float64
 8   price            1538 non-null   int64  
dtypes: float64(2), int64(6), object(1)
memory usage: 108.3+ KB
```

In [3]:
```
#This command is to describe the data present in the DataFrame in statistically
d.describe()
```

Out[3]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
count	1538.000000	1538.000000	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	11.563428	8576.003901
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

In [4]:
```
#The head() method returns a specified number of rows, string from the top
d.head()
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	lounge	51	882	25000	1	44.907242	8.611560	8900
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
2	3	sport	74	4658	142228	1	45.503300	11.417840	4200
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6000
4	5	pop	73	3074	106880	1	41.903221	12.495650	5700

In [5]:
```
#The tail() method returns a specified number of rows, string from the bottom
d.tail()
```

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7900

In [6]:
```
#This will return the values present in the specified column
d['price']
```

Out[6]:
```
0       8900
1       8800
2       4200
3       6000
4       5700
        ...
1533    5200
1534    4600
1535    7500
1536    5990
1537    7900
Name: price, Length: 1538, dtype: int64
```

In [7]:
```
#It returns the unique elements in the specified Column in a array
d['previous_owners'].unique()
```

Out[7]:
```
array([1, 2, 3, 4])
```

In [8]:
```
#This method will return the Columns names that present in the dataframe in a list.
list(d.columns)
```

Out[8]:
```
['ID',
 'model',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

In [9]:
```
'''groupby count method is used to count the values in each group by
ignoring the missing values or NaN values in the data frame.'''

d.groupby(['previous_owners']).count()
```

Out[9]:

previous_owners	ID	model	engine_power	age_in_days	km	lat	lon	price
1	1389	1389	1389	1389	1389	1389	1389	1389
2	117	117	117	117	117	117	117	117
3	23	23	23	23	23	23	23	23
4	9	9	9	9	9	9	9	9

In [10]:
```
#This method is to drop a columns from the dataframe.
data1=d.drop(['lat','ID'],axis=1)
data1
```

Out[10]:

	model	engine_power	age_in_days	km	previous_owners	lon	price
0	lounge	51	882	25000	1	8.611560	8900
1	pop	51	1186	32500	1	12.241890	8800
2	sport	74	4658	142228	1	11.417840	4200
3	lounge	51	2739	160000	1	17.634609	6000
4	pop	73	3074	106880	1	12.495650	5700
...	...	...	...	...	...	...	...
1533	sport	51	3712	115280	1	7.704920	5200
1534	lounge	74	3835	112000	1	8.666870	4600
1535	pop	51	2223	60457	1	9.413480	7500
1536	lounge	51	2557	80750	1	7.682270	5990
1537	pop	51	1766	54276	1	17.568270	7900

1538 rows × 7 columns

In [11]:
```
#This sum() is used to sum the values present in a column.
d['engine_power'].sum()
```

Out[11]: 79829

In [12]:
```
#loc[] is used to retrieve the group of rows and columns by labels
data2=d.loc[(d.model=='pop') & (d.engine_power==51)]
data2
```

Out[12]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1	2	pop	51	1186	32500	1	45.666359	12.241890	8800
10	11	pop	51	790	43286	1	40.871429	14.438960	8950
13	14	pop	51	3835	120000	1	40.531590	17.436159	4800
17	18	pop	51	2223	96848	1	43.782372	11.254990	7990
26	27	pop	51	3592	124000	1	40.966179	17.116480	6800
...	...	...	...	...	...	...	...	...	...
1524	1525	pop	51	2192	53300	1	40.609531	14.980930	7900
1527	1528	pop	51	517	3000	1	40.748241	14.528350	9999
1532	1533	pop	51	1917	52008	1	45.548000	11.549470	9900
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

333 rows × 9 columns

In [13]:
```
#To ignore warnings
import warnings
warnings.filterwarnings("ignore")
#This is to change a specified value in a column
d['model']=d['model'].map({'lounge':1,'pop':2,'sport':3})
d
```

Out[13]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
0	1	1	51	882	25000	1	44.907242	8.611560	8900
1	2	2	51	1186	32500	1	45.666359	12.241890	8800
2	3	3	74	4658	142228	1	45.503300	11.417840	4200
3	4	1	51	2739	160000	1	40.633171	17.634609	6000
4	5	2	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	3	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	1	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	2	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	1	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	2	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [14]:
```
#This method is used to change column name.
d=d.rename(columns={'previous_owners':'Previous Owners',})
d
```

Out[14]:

	ID	model	engine_power	age_in_days	km	Previous Owners	lat	lon	price
0	1	1	51	882	25000	1	44.907242	8.611560	8900
1	2	2	51	1186	32500	1	45.666359	12.241890	8800
2	3	3	74	4658	142228	1	45.503300	11.417840	4200
3	4	1	51	2739	160000	1	40.633171	17.634609	6000
4	5	2	73	3074	106880	1	41.903221	12.495650	5700
...	...	...	...	...	...	...	...	...	...
1533	1534	3	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	1	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	2	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	1	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	2	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

In [ ]:
```
# Note: The correlation between strings give NaN values. so removing strings from dataframe
#data1=d.drop(['model','engine_power'],axis=1)
```

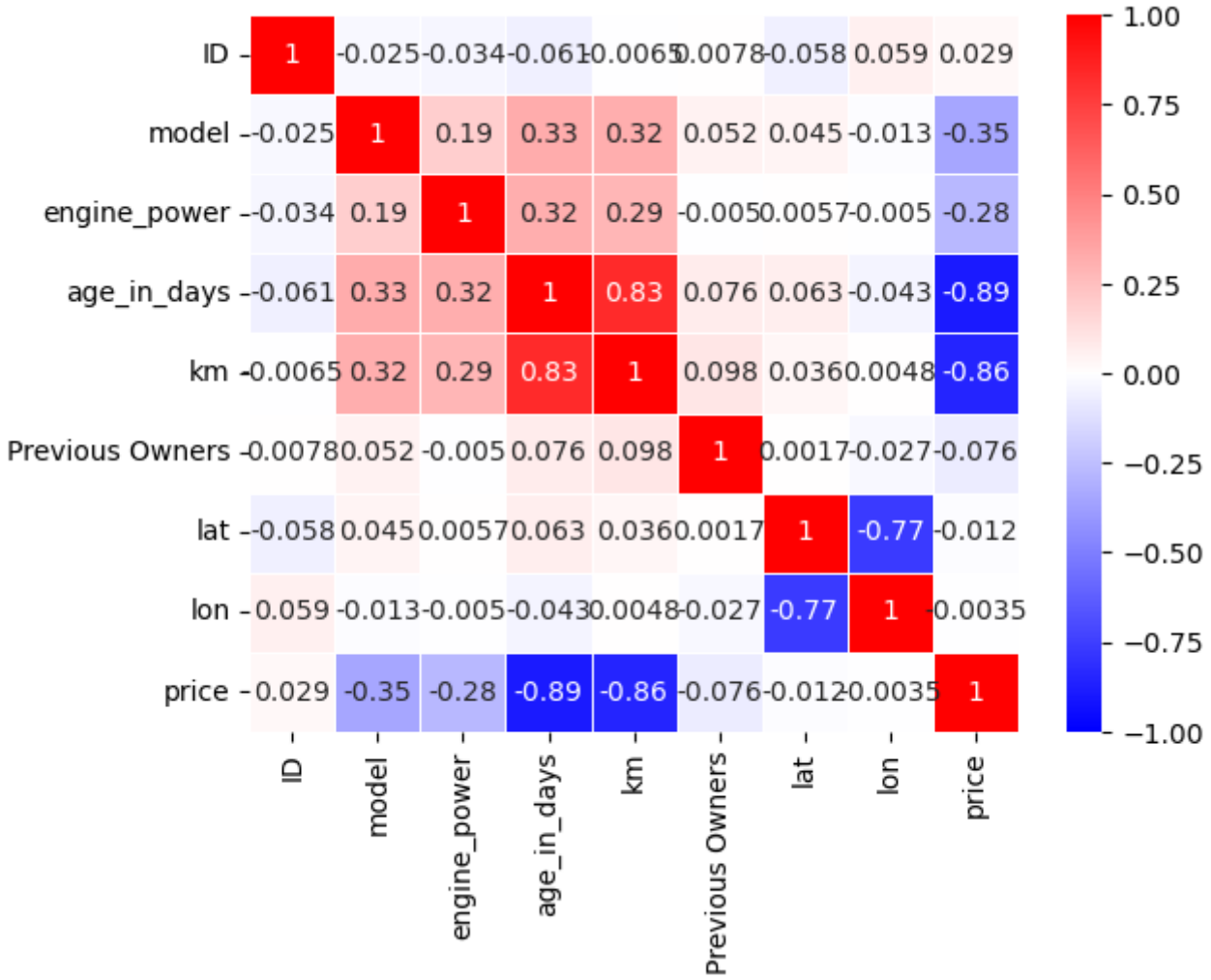
In [16]:
```
#This function is to find correlation
d3=d.corr()
d3
```

Out[16]:

	ID	model	engine_power	age_in_days	km	Previous Owners	lat	lon	price
ID	1.000000	-0.024740	-0.034059	-0.060753	-0.006537	0.007803	-0.058207	0.058941	0.028516
model	-0.024740	1.000000	0.189906	0.326508	0.319580	0.052480	0.044901	-0.013200	-0.349885
engine_power	-0.034059	0.189906	1.000000	0.319190	0.285495	-0.005030	0.005721	-0.005032	-0.277235
age_in_days	-0.060753	0.326508	0.319190	1.000000	0.833890	0.075775	0.062982	-0.042667	-0.893328
km	-0.006537	0.319580	0.285495	0.833890	1.000000	0.097539	0.035519	0.004839	-0.859373
Previous Owners	0.007803	0.052480	-0.005030	0.075775	0.097539	1.000000	0.001697	-0.026836	-0.076274
lat	-0.058207	0.044901	0.005721	0.062982	0.035519	0.001697	1.000000	-0.766646	-0.011733
lon	0.058941	-0.013200	-0.005032	-0.042667	0.004839	-0.026836	-0.766646	1.000000	-0.003541
price	0.028516	-0.349885	-0.277235	-0.893328	-0.859373	-0.076274	-0.011733	-0.003541	1.000000

In [17]:
```
#This create a heatmap
import seaborn as sns
sns.heatmap(d3,vmax=1,vmin=-1,annot=True,linewidth=.5,cmap='bwr')
```

Out[17]: <Axes: >



In [18]:
```
#This create a graph
d3.plot()
```

Out[18]: <Axes: >