In [1]:

```
import pandas as pd
```

In [2]:

```
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [3]:

```
data
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lo
0	1	lounge	51	882	25000	1	44.907242	8.61156
1	2	pop	51	1186	32500	1	45.666359	12.24189
2	3	sport	74	4658	142228	1	45.503300	11.41784
3	4	lounge	51	2739	160000	1	40.633171	17.63460
4	5	pop	73	3074	106880	1	41.903221	12.49565
...	...	...	...	...	...	...	...	.
1533	1534	sport	51	3712	115280	1	45.069679	7.70492
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687
1535	1536	pop	51	2223	60457	1	45.481541	9.41348
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227
1537	1538	pop	51	1766	54276	1	40.323410	17.56827

1538 rows × 9 columns

In [5]:

```
data1=data.loc[(data.previous_owners==1)]
data1
```

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lo
0	1	lounge	51	882	25000	1	44.907242	8.61156
1	2	pop	51	1186	32500	1	45.666359	12.24189
2	3	sport	74	4658	142228	1	45.503300	11.41784
3	4	lounge	51	2739	160000	1	40.633171	17.63460
4	5	pop	73	3074	106880	1	41.903221	12.49565
...	...	...	...	...	...	...	...	.
1533	1534	sport	51	3712	115280	1	45.069679	7.70492
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687
1535	1536	pop	51	2223	60457	1	45.481541	9.41348
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227
1537	1538	pop	51	1766	54276	1	40.323410	17.56827

1389 rows × 9 columns

In [6]:

```
data1=data1.drop(['ID','lat','lon'],axis=1)
data1
```

Out[6]:

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

1389 rows × 6 columns

In [7]:

```
data1=pd.get_dummies(data1)
data1
```

Out[7]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop
0	51	882	25000	1	8900	1	0
1	51	1186	32500	1	8800	0	1
2	74	4658	142228	1	4200	0	0
3	51	2739	160000	1	6000	1	0
4	73	3074	106880	1	5700	0	1
...	...	...	...	...	...	...	...
1533	51	3712	115280	1	5200	0	0
1534	74	3835	112000	1	4600	1	0
1535	51	2223	60457	1	7500	0	1
1536	51	2557	80750	1	5990	1	0
1537	51	1766	54276	1	7900	0	1

1389 rows × 8 columns

In [10]:

```
y=data1['price']
X=data1.drop(['price'],axis=1)
```

In [11]:

```
y
```

Out[11]:

```
0       8900
1       8800
2       4200
3       6000
4       5700
        ...
1533    5200
1534    4600
1535    7500
1536    5990
1537    7900
Name: price, Length: 1389, dtype: int64
```

In [12]:

```
X
```

Out[12]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	moc
0	51	882	25000	1	1	0	
1	51	1186	32500	1	0	1	
2	74	4658	142228	1	0	0	
3	51	2739	160000	1	1	0	
4	73	3074	106880	1	0	1	
...	...	...	...	...	...	...	
1533	51	3712	115280	1	0	0	
1534	74	3835	112000	1	1	0	
1535	51	2223	60457	1	0	1	
1536	51	2557	80750	1	1	0	
1537	51	1766	54276	1	0	1	

1389 rows × 7 columns

In [13]:

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.1,random_state=42)
```

In [14]:

```
X_train
```

Out[14]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model
956	51	790	26210	1	1	0	
1411	51	1461	46108	1	1	0	
333	51	456	26526	1	1	0	
1452	51	1247	75000	1	1	0	
1369	51	701	36500	1	1	0	
...	...	...	...	...	...	...	
1201	51	790	50740	1	0	1	
1239	51	4383	107600	1	0	1	
1432	51	701	42095	1	1	0	
951	51	3684	78000	1	1	0	
1235	51	1613	45000	1	1	0	

1250 rows × 7 columns

In [15]:

```
y_train
```

Out[15]:

```
956     8750
1411    8000
333     9980
1452    8000
1369    9990
        ...
1201    8300
1239    3950
1432    8900
951     6500
1235    8800
Name: price, Length: 1250, dtype: int64
```

In [16]:

```
y_test
```

Out[16]:

```
625      5400
187      5399
279      4900
734     10500
315      9300
        ...
1507     9950
806      9700
1090    10400
436      7950
937      7100
Name: price, Length: 139, dtype: int64
```

In [18]:

```
from sklearn.model_selection import GridSearchCV
```

In [19]:

```
from sklearn.linear_model import ElasticNet

elastic = ElasticNet()

parameters = {'alpha': [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20]}

elastic_regressor = GridSearchCV(elastic, parameters)

elastic_regressor.fit(X_train, y_train)
```

```
check the scale of the features or consider increasing regularisatio
n. Duality gap: 2.841e+08, tolerance: 3.711e+05
  model = cd_fast.enet_coordinate_descent(
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linea
r_model/_coordinate_descent.py:631: ConvergenceWarning: Objective di
d not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisatio
n. Duality gap: 2.997e+08, tolerance: 3.576e+05
  model = cd_fast.enet_coordinate_descent(
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linea
r_model/_coordinate_descent.py:631: ConvergenceWarning: Objective di
d not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisatio
n. Duality gap: 2.860e+08, tolerance: 3.519e+05
  model = cd_fast.enet_coordinate_descent(
/home/placement/anaconda3/lib/python3.10/site-packages/sklearn/linea
r_model/_coordinate_descent.py:631: ConvergenceWarning: Objective di
d not converge. You might want to increase the number of iterations,
check the scale of the features or consider increasing regularisatio
```

In [ ]: