

```
In [2]: import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [3]: data=pd.read_csv("/home/placement/Desktop/nio/fiat500.csv")
```

```
In [4]: data.describe()
```

Out[4]:

	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 [5]: data.head

```
Out[5]: <bound method NDFrame.head of
0      1 lounge      51      ID      model  engine_power  age_in_days      km  previous_owners  \
1      2      pop      51      1186    32500              1
2      3      sport     74      4658   142228              1
3      4 lounge      51      2739   160000              1
4      5      pop      73      3074   106880              1
...    ...    ...    ...    ...    ...    ...
1533  1534 sport      51      3712   115280              1
1534  1535 lounge     74      3835   112000              1
1535  1536      pop     51      2223    60457              1
1536  1537 lounge     51      2557    80750              1
1537  1538      pop     51      1766    54276              1

      lat      lon  price
0  44.907242  8.611560  8900
1  45.666359  12.241890  8800
2  45.503300  11.417840  4200
3  40.633171  17.634609  6000
4  41.903221  12.495650  5700
...    ...    ...    ...
1533  45.069679  7.704920  5200
1534  45.845692  8.666870  4600
1535  45.481541  9.413480  7500
1536  45.000702  7.682270  5990
1537  40.323410  17.568270  7900

[1538 rows x 9 columns]>
```

In [6]: data.head()

Out[6]:

	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 [7]: data

Out[7]:

	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
...
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5200
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4600
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7500
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5990
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7900

1538 rows × 9 columns

```
In [8]: list(data)
```

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

```
In [9]: data1=data.drop(["lat","lon","ID"],axis=1)
```

```
In [10]: data1
```

```
Out[10]:
```

	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

1538 rows × 6 columns

```
In [11]: data1=pd.get_dummies(data1)
```

```
In [13]: data1
```

```
Out[13]:
```

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

1538 rows × 8 columns

```
In [14]: y=data["price"]  
x=data1.drop(["price"],axis=1)
```

In [15]:

y

Out[15]:

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 [19]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.33,random_state=42)
#splits data into 33% testing and 66% training data
```

In [21]:

```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
ypred=reg.predict(x_test)
```

In [23]:

```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[23]: 0.8415526986865394

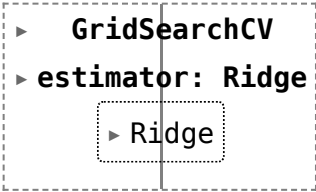
In [28]:

```
from sklearn.metrics import mean_squared_error
mean_squared_error(ypred,y_test)
```

Out[28]: 581887.727391353

```
In [31]: from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
alpha=[1e-15,1e-10,1e-8,1e-4,1e-3,1e-2,1,5,10,20,30]
ridge=Ridge()
parameters={'alpha':alpha}
ridge_regressor=GridSearchCV(ridge,parameters)
ridge_regressor.fit(x_train,y_train)
```

```
Out[31]:
```



```
  ▸ GridSearchCV
  ▸ estimator: Ridge
    ▸ Ridge
```

```
In [32]: ridge_regressor.best_params_
```

```
Out[32]: {'alpha': 30}
```

```
In [34]: ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

```
In [35]: from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

```
Out[35]: 579521.7970897449
```

```
In [36]: from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

```
Out[36]: 0.8421969385523054
```

```
In [41]: from sklearn.linear_model import ElasticNet
import warnings
warnings.filterwarnings("ignore")
from sklearn.model_selection import GridSearchCV

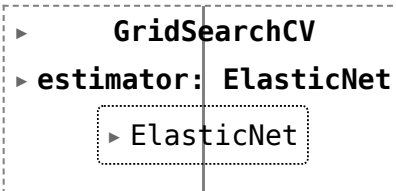
elastic = ElasticNet()

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)
```

```
Out[41]:
```



```
  ▶ GridSearchCV
  ▶ estimator: ElasticNet
    ▶ ElasticNet
```

```
In [42]: elastic_regressor.best_params_
```

```
Out[42]: {'alpha': 0.01}
```

```
In [45]: elastic=ElasticNet(alpha=.01)
elastic.fit(x_train,y_train)#creates elastic const as training data
y_pred_elastic=elastic.predict(x_test)
```

```
In [46]: from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)#EFFICIENCY
```

```
Out[46]: 0.841688021120299
```

```
In [47]: from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error#MEAN SQUARED ERROR
```

```
Out[47]: 581390.7642825295
```


to plot th graph sns plot

```
Results=pd.DataFrame(columns=['actual','predicted'])# CREATING A DATA FRAME AND INSERING COLS
Results['actual']=y_test
Results['predicted']=y_pred_elastic
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(25)
```

```
sns.lineplot(x='ID',y="actual",data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

In []:

In []:

In []:

In []:

In []:

In []: