In [1]:
```
import pandas as pd
```

In [2]:
```
data=pd.read_csv("/home/placement/Desktop/reddy/fiat500.csv")
```

In [3]:
```
data.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]:
```
data.tail(10)
```

Out[4]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1528	1529	lounge	51	2861	126000	1	43.841980	10.51531	5500
1529	1530	lounge	51	731	22551	1	38.122070	13.36112	9900
1530	1531	lounge	51	670	29000	1	45.764648	8.99450	10800
1531	1532	sport	73	4505	127000	1	45.528511	9.59323	4750
1532	1533	pop	51	1917	52008	1	45.548000	11.54947	9900
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 [5]:
```
data1=data.drop(['ID','lat','lon'],axis=1)
```

In [6]:
```
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

1538 rows × 6 columns

In [7]:
```
data1['model']=data1['model'].map({'lounge':1,'pop':2,'sport':3})
```

In [8]:
```
data1
```

Out[8]:

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

1538 rows × 6 columns

In [10]:
```
data1=pd.get_dummies(data1)
```

In [11]:
```
data1
```

Out[11]:

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

1538 rows × 6 columns

In [12]:
```
y=data1['price']
x=data1.drop('price',axis=1)
```

In [13]:
```
y
```

Out[13]:
```
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 [14]:
```
x
```

Out[14]:

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

1538 rows × 5 columns

In [15]:
```
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)
```

In [16]:
```
x_test.head(5)
```

Out[16]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1

In [17]:
```
x_train.shape
```

Out[17]:
```
(1030, 5)
```

In [18]:
```
y_train.shape
```

Out[18]:
```
(1030,)
```

In [19]:
```
x_train.head()
```

Out[19]:

	model	engine_power	age_in_days	km	previous_owners
527	1	51	425	13111	1
129	1	51	1127	21400	1
602	2	51	2039	57039	1
331	1	51	1155	40700	1
323	1	51	425	16783	1

In [20]:
```
y_train.head()
```

Out[20]:
```
527    9990
129    9500
602    7590
331    8750
323    9100
Name: price, dtype: int64
```

In [21]:
```
x_test.head()
```

Out[21]:

	model	engine_power	age_in_days	km	previous_owners
481	2	51	3197	120000	2
76	2	62	2101	103000	1
1502	1	51	670	32473	1
669	1	51	913	29000	1
1409	1	51	762	18800	1

In [22]:
```
y_test.head()
```

Out[22]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

#linear regression

In [23]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
ypred=reg.predict(x_test)
```

In [25]:
```
ypred
```

Out[25]:
```
array([ 5994.51703157,  7263.58726658,  9841.90754881,  9699.31627673,
       10014.19892635,  9630.58715835,  9649.4499026 , 10092.9819664 ,
        9879.19498711,  9329.19347948, 10407.2964056 ,  7716.91706011,
        7682.89152522,  6673.95810983,  9639.42618839, 10346.53679153,
        9366.53363673,  7707.90063494,  4727.33552438, 10428.17092937,
       10359.87663878, 10364.84674179,  7680.16157493,  9927.58506055,
        7127.7284177 ,  9097.51161986,  4929.31229715,  6940.60225317,
        7794.35120591,  9600.43942019,  7319.85877519,  5224.05298205,
        5559.52039134,  5201.35403287,  8960.11762682,  5659.72968338,
        9915.79926869,  8255.93615893,  6270.40332834,  8556.73835062,
        9749.72882426,  6873.76758364,  8951.72659758, 10301.95669828,
        8674.89268564, 10301.93257222,  9165.73586068,  8846.92420399,
        7044.68964545,  9052.4031418 ,  9390.75738772, 10267.3912561 ,
       10046.90924744,  6855.71260655,  9761.93338967,  9450.05744337,
        9274.98388541, 10416.00474283,  9771.10646661,  7302.96566423,
       10082.61483093,  6996.96553454,  9829.40534825,  7134.21944391,
        6407.26222178,  9971.82132188,  9757.01618446,  8614.84049875,
        8437.92452169,  6489.24658616,  7752.65456507,  6626.60510856,
        8329.88998217, 10412.00324329,  7342.77348105,  8543.63624413,
```

In [26]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[26]:
```
0.8383895235218546
```

In [27]:
```
from sklearn.metrics import mean_squared_error as ns
o=ns(y_test,ypred)
o
```

Out[27]:
```
593504.2888137395
```

In [28]:
```
import math
math.sqrt(o)
```

Out[28]:
```
770.3922954013361
```

In [29]:
```
ypred
```

Out[29]:
```
array([ 5994.51703157,  7263.58726658,  9841.90754881,  9699.31627673,
       10014.19892635,  9630.58715835,  9649.4499026 , 10092.9819664 ,
        9879.19498711,  9329.19347948, 10407.2964056 ,  7716.91706011,
        7682.89152522,  6673.95810983,  9639.42618839, 10346.53679153,
        9366.53363673,  7707.90063494,  4727.33552438, 10428.17092937,
       10359.87663878, 10364.84674179,  7680.16157493,  9927.58506055,
        7127.7284177 ,  9097.51161986,  4929.31229715,  6940.60225317,
        7794.35120591,  9600.43942019,  7319.85877519,  5224.05298205,
        5559.52039134,  5201.35403287,  8960.11762682,  5659.72968338,
        9915.79926869,  8255.93615893,  6270.40332834,  8556.73835062,
        9749.72882426,  6873.76758364,  8951.72659758, 10301.95669828,
        8674.89268564, 10301.93257222,  9165.73586068,  8846.92420399,
        7044.68964545,  9052.4031418 ,  9390.75738772, 10267.3912561 ,
       10046.90924744,  6855.71260655,  9761.93338967,  9450.05744337,
        9274.98388541, 10416.00474283,  9771.10646661,  7302.96566423,
       10082.61483093,  6996.96553454,  9829.40534825,  7134.21944391,
        6407.26222178,  9971.82132188,  9757.01618446,  8614.84049875,
        8437.92452169,  6489.24658616,  7752.65456507,  6626.60510856,
        8329.88998217, 10412.00324329,  7342.77348105,  8543.63624413,
```

In [30]:
```
Results=pd.DataFrame(columns=['price','predicted'])
Results['price']=y_test
Results['predicted']=ypred
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(15)
```

Out[30]:

	index	price	predicted	ID
0	481	7900	5994.517032	0
1	76	7900	7263.587267	1
2	1502	9400	9841.907549	2
3	669	8500	9699.316277	3
4	1409	9700	10014.198926	4
5	1414	9900	9630.587158	5
6	1089	9900	9649.449903	6
7	1507	9950	10092.981966	7
8	970	10700	9879.194987	8
9	1198	8999	9329.193479	9
10	1088	9890	10407.296406	10
11	576	7990	7716.917060	11
12	965	7380	7682.891525	12
13	1488	6800	6673.958110	13
14	1432	8900	9639.426188	14

In [31]:
```
Results['price_diff']=Results.apply(lambda row: row.price - row.predicted,axis=1)
```

In [32]:
```
Results
```

Out[32]:

	index	price	predicted	ID	price_diff
0	481	7900	5994.517032	0	1905.482968
1	76	7900	7263.587267	1	636.412733
2	1502	9400	9841.907549	2	-441.907549
3	669	8500	9699.316277	3	-1199.316277
4	1409	9700	10014.198926	4	-314.198926
...	...	...	...	...	...
503	291	10900	10007.364639	503	892.635361
504	596	5699	6390.174715	504	-691.174715
505	1489	9500	10079.478928	505	-579.478928
506	1436	6990	8363.337585	506	-1373.337585
507	575	10900	10344.486077	507	555.513923

508 rows × 5 columns

#ridge regression

In [33]:
```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
#ridge regression
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[33]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [34]:
```
ridge_regressor.best_params_
```

Out[34]:
```
{'alpha': 30}
```

In [35]:
```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

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

Out[36]:
```
590569.9121697355
```

In [37]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[37]: 0.8391885506165899

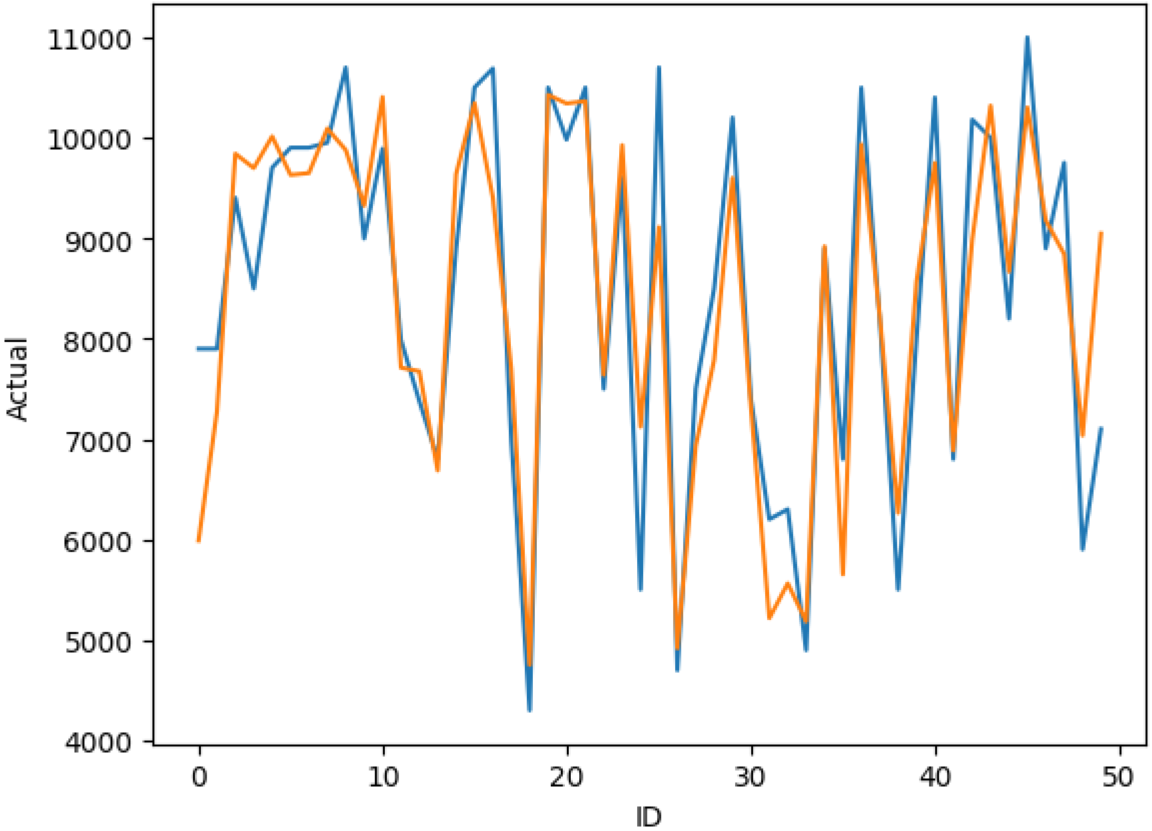
In [38]:
```
Results=pd.DataFrame(columns=['Actual','predicted'])
Results['Actual']=y_test
Results['predicted']=y_pred_ridge
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(10)
```

Out[38]:

	index	Actual	predicted	ID
0	481	7900	5987.682984	0
1	76	7900	7272.490419	1
2	1502	9400	9839.847697	2
3	669	8500	9696.775405	3
4	1409	9700	10012.040862	4
5	1414	9900	9628.286853	5
6	1089	9900	9646.945160	6
7	1507	9950	10090.960592	7
8	970	10700	9877.094341	8
9	1198	8999	9326.088982	9

In [39]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='Actual',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[39]: []



#elastic

In [42]:
```
from sklearn.linear_model import ElasticNet
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[42]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [43]:
```
elastic_regressor.best_params_
```

Out[43]:
```
{'alpha': 0.01}
```

In [47]:
```
elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [48]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[48]:
```
0.8385500526604823
```

In [49]:
```
from sklearn.metrics import mean_squared_error
Elasticnet_Error=mean_squared_error(y_pred_elastic,y_test)
Elasticnet_Error
```

Out[49]:
```
592914.7556700263
```

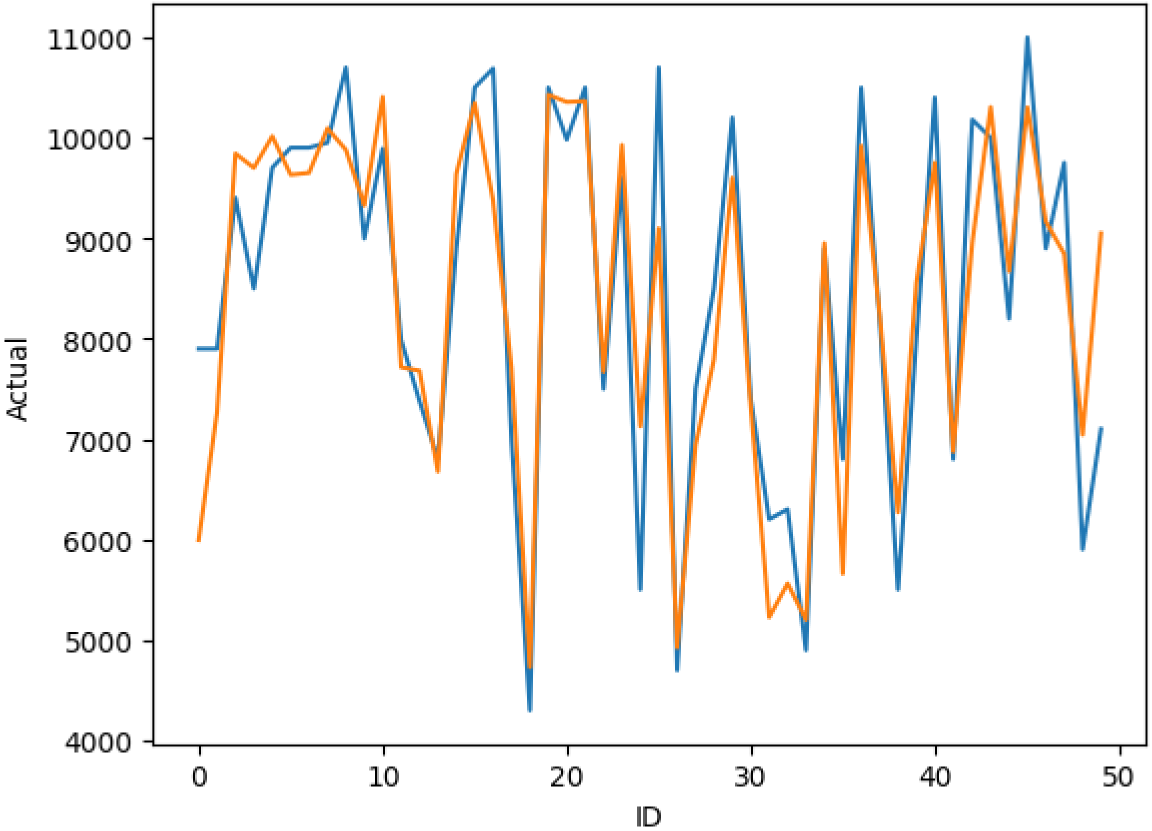
In [50]:
```
Results=pd.DataFrame(columns=['Actual','predicted'])
Results['Actual']=y_test
Results['predicted']=y_pred_elastic
Results=Results.reset_index()
Results['ID']=Results.index
Results.head(10)
```

Out[50]:

	index	Actual	predicted	ID
0	481	7900	5993.053059	0
1	76	7900	7265.275818	1
2	1502	9400	9841.546147	2
3	669	8500	9698.864284	3
4	1409	9700	10013.815854	4
5	1414	9900	9630.182678	5
6	1089	9900	9649.005668	6
7	1507	9950	10092.624034	7
8	970	10700	9878.825124	8
9	1198	8999	9328.638538	9

In [51]:
```
import seaborn as sns#plot
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='Actual',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[51]: []



In [ ]: