In [37]:
```
import pandas as pd
data=pd.read_csv("/home/placement/Downloads/fiat500.csv")
```

In [38]:
```
data.describe()
```

Out[38]:

	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 [39]:
```
data.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 [40]:
```
data1=data.loc[(data.previous_owners==1)]
```

In [41]:
```
data1
```

Out[41]:

	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

1389 rows × 9 columns

In [42]:
```
data1=data.drop(['ID','lat','lon'],axis=1)
```

```
In [43]: data1
```

```
Out[43]:
         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 [44]: data1=pd.get_dummies(data)
In [45]: data1
```

```
Out[45]:
        ID  engine_power  age_in_days      km  previous_owners        lat        lon  price  model_lounge  model_pop  model_sport
   0     1            51          882   25000                1  44.907242   8.611560   8900             1          0            0
   1     2            51         1186   32500                1  45.666359  12.241890   8800             0          1            0
   2     3            74         4658  142228                1  45.503300  11.417840   4200             0          0            1
   3     4            51         2739  160000                1  40.633171  17.634609   6000             1          0            0
   4     5            73         3074  106880                1  41.903221  12.495650   5700             0          1            0
 ...   ...           ...          ...     ...              ...        ...        ...    ...           ...        ...          ...
1533  1534            51         3712  115280                1  45.069679   7.704920   5200             0          0            1
1534  1535            74         3835  112000                1  45.845692   8.666870   4600             1          0            0
1535  1536            51         2223   60457                1  45.481541   9.413480   7500             0          1            0
1536  1537            51         2557   80750                1  45.000702   7.682270   5990             1          0            0
1537  1538            51         1766   54276                1  40.323410  17.568270   7900             0          1            0

1538 rows × 11 columns
```

```
In [46]: y=data1['price']
         x=data1.drop('price',axis=1)
```

In [47]:

```
y
```

Out[47]:

```
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 [48]:

```
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 [49]:

```
x_test.head(5)
```

Out[49]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	model_lounge	model_pop	model_sport
481	482	51	3197	120000	2	40.174702	18.167629	0	1	0
76	77	62	2101	103000	1	45.797859	8.644440	0	1	0
1502	1503	51	670	32473	1	41.107880	14.208810	1	0	0
669	670	51	913	29000	1	45.778591	8.946250	1	0	0
1409	1410	51	762	18800	1	45.538689	9.928310	1	0	0

In [50]:
```
x_train.head(5)
```

Out[50]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	model_lounge	model_pop	model_sport
527	528	51	425	13111	1	45.022388	7.58602	1	0	0
129	130	51	1127	21400	1	44.332531	7.54592	1	0	0
602	603	51	2039	57039	1	40.748241	14.52835	0	1	0
331	332	51	1155	40700	1	42.143860	12.54016	1	0	0
323	324	51	425	16783	1	41.903221	12.49565	1	0	0

In [51]:
```
y_test.head(5)
```

Out[51]:
```
481     7900
76      7900
1502    9400
669     8500
1409    9700
Name: price, dtype: int64
```

In [52]:
```
y_train.head(5)
```

Out[52]:
```
527    9990
129    9500
602    7590
331    8750
323    9100
Name: price, dtype: int64
```

In [53]:
```
x_train.shape
```

Out[53]:
```
(1030, 10)
```

In [54]:
```
y_train
```

Out[54]:
```
527      9990
129      9500
602      7590
331      8750
323      9100
        ...  
1130    10990
1294     9800
860      5500
1459     9990
1126     8900
Name: price, Length: 1030, dtype: int64
```

In [55]:
```
#linear_regression
from sklearn.linear_model import LinearRegression
reg=LinearRegression()
reg.fit(x_train,y_train)
```

Out[55]:
```
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 [56]:
```
ypred=reg.predict(x_test)
```

```
In [57]: ypred
```

```
Out[57]: array([ 5819.19308764,  7248.82914161,  9741.8936974 ,  9798.98033074,
                10055.00624601,  9551.4955679 ,  9758.01743879, 10122.9778365 ,
                 9654.9661814 ,  9251.1403257 , 10478.09512253,  7807.3005255 ,
                 7705.15873781,  6295.63244894,  9545.40486313, 10422.92177704,
                 9616.90811615,  7756.9171161 ,  4893.88454414, 10581.46142719,
                10465.24078346, 10443.29318231,  7518.43696046, 10028.21911459,
                 6990.73118896,  8989.86900819,  4823.51364349,  6989.03118684,
                 7822.83203734,  9683.17944083,  7344.21343132,  5341.43860798,
                 5420.78405336,  5092.38401339,  8971.44357515,  5702.81242412,
                 9920.16285466,  8334.58448277,  6220.93323723,  8389.23958511,
                 9695.84208061,  6859.59630725,  9101.22635456, 10063.22592995,
                 8621.83915759, 10175.06753933,  9063.21918346,  8867.24865352,
                 7094.44228184,  9058.37693565,  9474.82390731, 10406.09102832,
                10112.65006224,  6820.90463865,  9700.36507783,  9382.18149429,
                 9632.57617775, 10553.81356008,  9847.21129432,  7247.16814789,
                 9990.23331336,  7084.23300123,  9977.34233656,  7245.01115798,
                 6490.89305576,  9737.86785115,  9853.54349825,  8568.7125607 ,
                 8506.81438703,  6484.69051659,  7883.1895563 ,  6870.28308427,
                 8263.36833348, 10551.03496347,  7434.71134313,  8637.85174602,
                 9763.07817027, 10019.4780977 ,  7324.68808828,  9537.73426033
```

In [58]:

```
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[58]:

	index	price	predicted	ID
0	481	7900	5819.193088	0
1	76	7900	7248.829142	1
2	1502	9400	9741.893697	2
3	669	8500	9798.980331	3
4	1409	9700	10055.006246	4
5	1414	9900	9551.495568	5
6	1089	9900	9758.017439	6
7	1507	9950	10122.977837	7
8	970	10700	9654.966181	8
9	1198	8999	9251.140326	9
10	1088	9890	10478.095123	10
11	576	7990	7807.300526	11
12	965	7380	7705.158738	12
13	1488	6800	6295.632449	13
14	1432	8900	9545.404863	14

In [59]:

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

In [60]:

```
Results
```

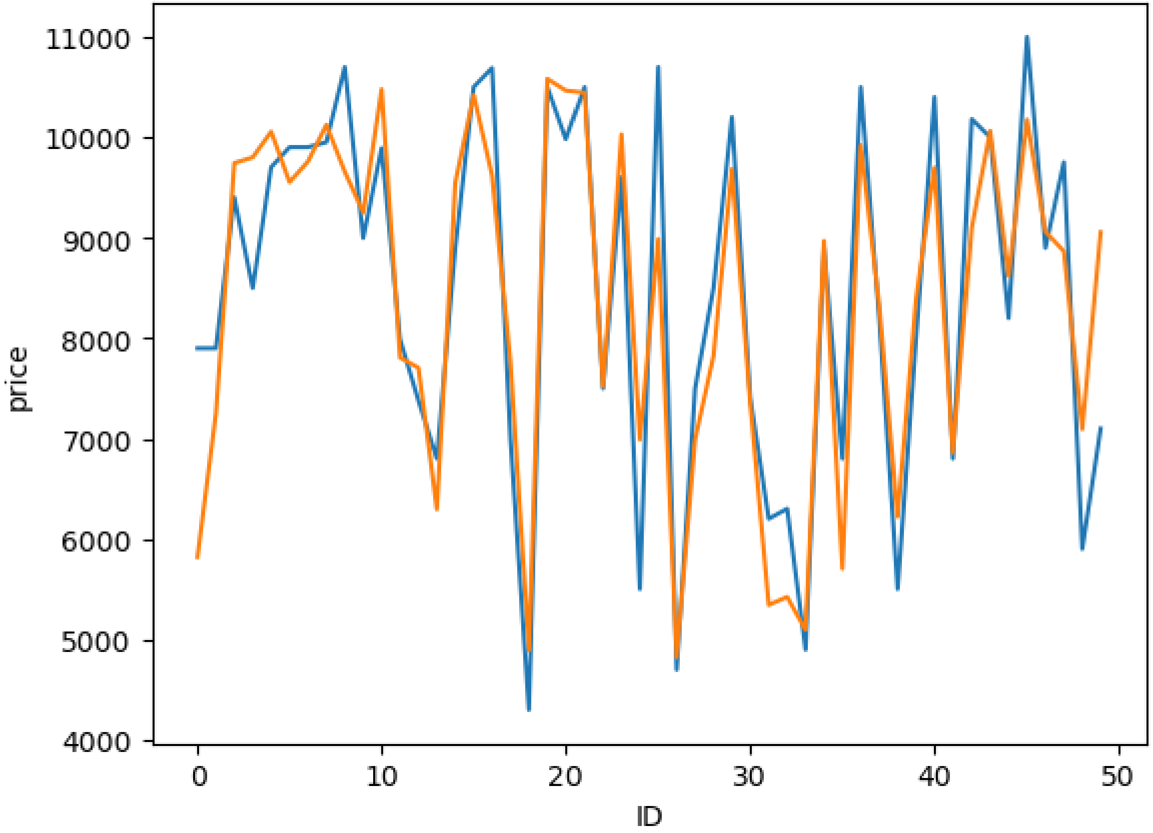
Out[60]:

```
     index  price     predicted   ID         diff
  0    481   7900   5819.193088    0  2080.806912
  1     76   7900   7248.829142    1   651.170858
  2   1502   9400   9741.893697    2  -341.893697
  3    669   8500   9798.980331    3 -1298.980331
  4   1409   9700  10055.006246    4  -355.006246
...    ...    ...           ...  ...          ...
503    291  10900  10121.593384  503   778.406616
504    596   5699   6288.648282  504  -589.648282
505   1489   9500  10016.505537  505  -516.505537
506   1436   6990   8248.746492  506 -1258.746492
507    575  10900  10337.345820  507   562.654180

508 rows × 5 columns
```

In [61]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='ID',y='price',data=Results.head(50))
sns.lineplot(x='ID',y='predicted',data=Results.head(50))
plt.plot()
```

Out[61]:
```
[]
```



In [62]:

```
import warnings
warnings.filterwarnings('ignore')
```

In [63]:

```
#ridge regression
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[63]:

```
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 [64]:

```
ridge_regressor.best_params_
```

Out[64]:

```
{'alpha': 30}
```

In [65]:

```
ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

In [66]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[66]:
```
574728.5696156605
```

In [67]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[67]:
```
0.8435021284061197
```

In [68]:
```
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)
```

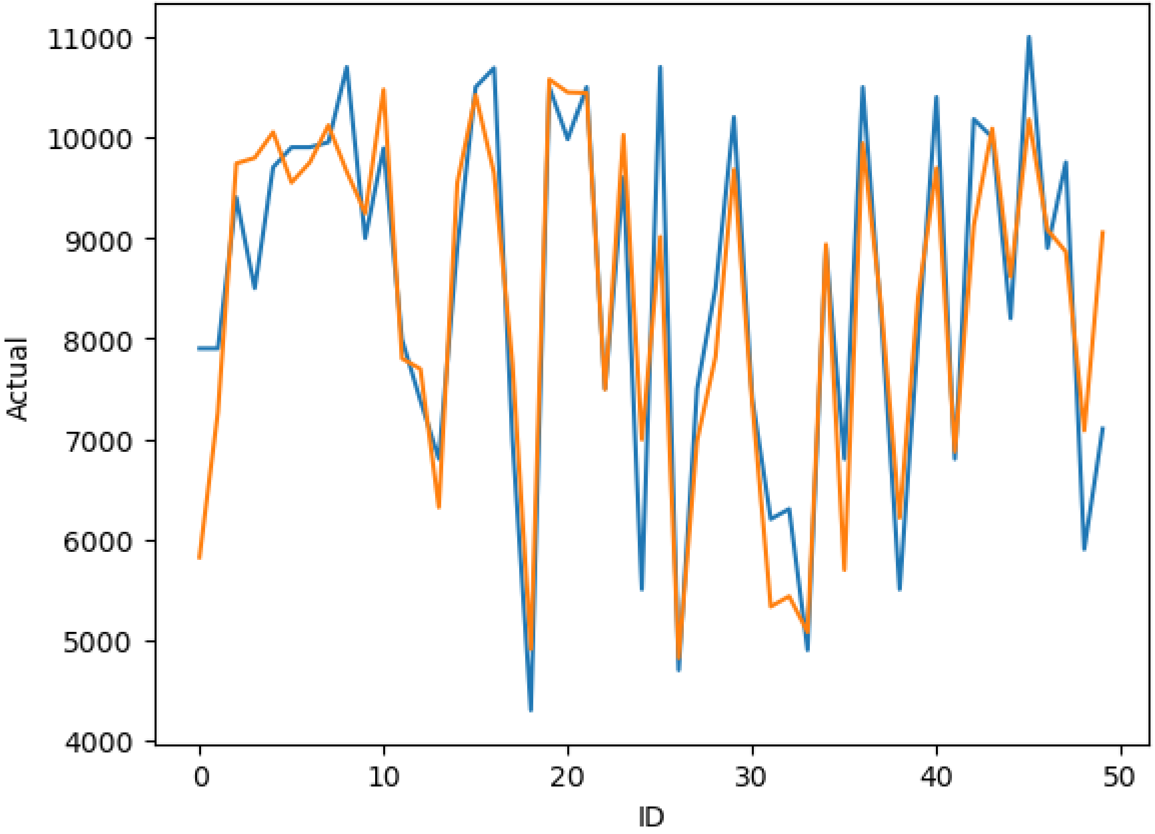
Out[68]:

	index	Actual	Predicted	ID
0	481	7900	5819.298540	0
1	76	7900	7264.574918	1
2	1502	9400	9738.882706	2
3	669	8500	9794.478395	3
4	1409	9700	10050.350724	4
5	1414	9900	9548.821263	5
6	1089	9900	9750.202837	6
7	1507	9950	10118.769447	7
8	970	10700	9656.236315	8
9	1198	8999	9247.205270	9

In [69]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [71]:
```
sns.lineplot(x='ID',y='Actual',data=Results.head(50))
sns.lineplot(x='ID',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[71]:
```
[]
```



In [72]:

```
#elastic_model
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[72]:

```
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 [73]:

```
elastic_regressor.best_params_
```

Out[73]:

```
{'alpha': 0.01}
```

In [74]:

```
elastic=ElasticNet(alpha=.33)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [75]:

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

Out[75]:

```
0.8445968963244241
```

In [76]:

```
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[76]:

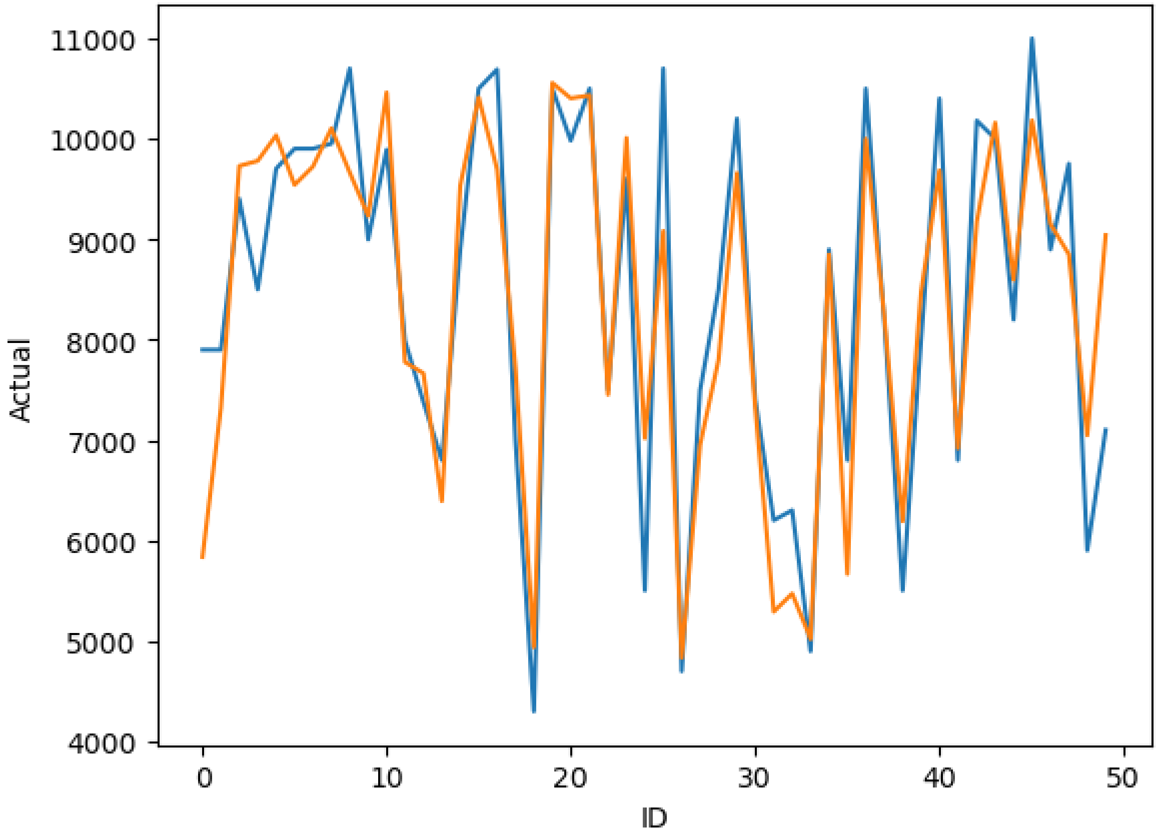
	index	Actual	Predicted	ID
0	481	7900	5834.887172	0
1	76	7900	7318.839756	1
2	1502	9400	9727.583531	2
3	669	8500	9778.566002	3
4	1409	9700	10033.013512	4
5	1414	9900	9538.968427	5
6	1089	9900	9721.786450	6
7	1507	9950	10102.881546	7
8	970	10700	9661.277720	8
9	1198	8999	9233.614930	9

In [77]:

```
import seaborn as sns
import matplotlib.pyplot as plt
```

```
In [79]: sns.lineplot(x='ID',y='Actual',data=Results.head(50))
         sns.lineplot(x='ID',y='Predicted',data=Results.head(50))
         plt.plot()

Out[79]: []
```



In [ ]: