In [1]:
```
import pandas as a
import pickle
import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
data=a.read_csv("/home/manu/Desktop/online/fiat500.csv")
```

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

Out[3]:

	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
5	6	pop	74	3623	70225	1	45.000702	7.682270	7900
6	7	lounge	51	731	11600	1	44.907242	8.611560	10750
7	8	lounge	51	1521	49076	1	41.903221	12.495650	9190
8	9	sport	73	4049	76000	1	45.548000	11.549470	5600
9	10	sport	51	3653	89000	1	45.438301	10.991700	6000

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 [ ]:

In [ ]:

In [5]:
```
data.tail(100)
```

Out[5]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
1438	1439	pop	51	790	44687	1	41.107880	14.20881	8800
1439	1440	lounge	51	1461	103519	1	43.879860	10.78281	7500
1440	1441	lounge	51	4474	143900	1	44.643131	10.93406	5400
1441	1442	lounge	51	1066	69916	1	45.571220	9.15914	8980
1442	1443	lounge	51	670	32835	1	41.107880	14.20881	9400
...	...	...	...	...	...	...	...	...	...
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

100 rows × 9 columns

In [6]:
```
data['engine_power'].unique()
```

Out[6]:
```
array([51, 74, 73, 62, 63, 66, 77, 58])
```

In [7]:
```
data.groupby(['engine_power']).count()
```

Out[7]:

engine_power	ID	model	age_in_days	km	previous_owners	lat	lon	price
51	1453	1453	1453	1453	1453	1453	1453	1453
58	1	1	1	1	1	1	1	1
62	43	43	43	43	43	43	43	43
63	1	1	1	1	1	1	1	1
66	1	1	1	1	1	1	1	1
73	22	22	22	22	22	22	22	22
74	14	14	14	14	14	14	14	14
77	3	3	3	3	3	3	3	3

In [8]:
```
data.groupby(['previous_owners']).count()
```

Out[8]:

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 [9]:
```
data['model'].unique()
```

Out[9]:
```
array(['lounge', 'pop', 'sport'], dtype=object)
```

In [10]:
```
#df=data
#data=df.loc[(df.model=='lounge')&(df.previous_owners==1)]
```

In [11]:
```
data1=data.drop(['lat','ID'],axis=1)
```

In [12]:
```
#2-3
data2=data1.drop('lon',axis=1)
```

In [13]:
```
data2.shape
#data2['model'] = data['model'].map({'lounge':1,'pop':2,'sport':3})
```

Out[13]: (1538, 6)

In [14]:
```
data2=a.get_dummies(data2)
```

In [15]:
```
#data2.groupby(['previous_owners'])
data2.shape
```

Out[15]: (1538, 8)

In [16]:
```
data2
```

Out[16]:

	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 [17]:
```
y=data2['price']
X=data2.drop('price',axis=1)
```

In [18]:
```
y
```

Out[18]:
```
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) #0.67 data will be
```

In [20]:
```
X_test.head(5)
```

Out[20]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
481	51	3197	120000	2	0	1	0
76	62	2101	103000	1	0	1	0
1502	51	670	32473	1	1	0	0
669	51	913	29000	1	1	0	0
1409	51	762	18800	1	1	0	0

In [21]:
```
X_train.shape
```

Out[21]:
```
(1030, 7)
```

In [22]:
```
y_train
```

Out[22]:
```
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 [23]:
```
from sklearn.linear_model import LinearRegression
reg = LinearRegression() #creating object of LinearRegression
reg.fit(X_train,y_train) #training and fitting LR object using training data
```

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

In [24]:
```
#X_test=[[1,51,1000,28800,3],[1,51,780,18800,1]]
```

In [25]:
```
#above line to actual
```

In [26]:
```
ypred=reg.predict(X_test)
```

In [27]:
```
ypred
```

Out[27]:
```
array([ 5867.6503378 ,  7133.70142341,  9866.35776216,  9723.28874535,
       10039.59101162,  9654.07582608,  9673.14563045, 10118.70728123,
        9903.85952664,  9351.55828437, 10434.34963575,  7732.26255693,
        7698.67240131,  6565.95240435,  9662.90103518, 10373.20344286,
        9599.94844451,  7699.34400418,  4941.33017994, 10455.2719478 ,
       10370.51555682, 10391.60424404,  7529.06622456,  9952.37340054,
        7006.13845729,  9000.1780961 ,  4798.36770637,  6953.10376491,
        7810.39767825,  9623.80497535,  7333.52158317,  5229.18705519,
        5398.21541073,  5157.65652129,  8948.63632836,  5666.62365159,
        9822.1231461 ,  8258.46551788,  6279.2040404 ,  8457.38443276,
        9773.86444066,  6767.04074749,  9182.99904787, 10210.05195479,
        8694.90545226, 10328.43369248,  9069.05761443,  8866.7826029 ,
        7058.39787506,  9073.33877162,  9412.68162121, 10293.69451263,
       10072.49011135,  6748.5794244 ,  9785.95841801,  9354.09969973,
        9507.9444386 , 10443.01608254,  9795.31884316,  7197.84932877,
       10108.31707235,  7009.6597206 ,  9853.90699412,  7146.87414965,
        6417.69133992,  9996.97382441,  9781.18795953,  8515.83255277,
        8456.30006203,  6499.76668237,  7768.57829985,  6832.86406122,
        8347.96113362, 10439.02404036,  7356.43463051,  8562.56562053,
```

In [28]:
```
filename='pricemodeldummy1'
pickle.dump(reg,open(filename,'wb'))
```

In [ ]:
```

```

In [29]:
```
#savedmodel=pickle.load(open(filename,'rb'))

#X_test=[[1,75,1062,8000,1]]
#savedmodel.predict(X_test)
```

In [30]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[30]:
```
0.8415526986865394
```

In [ ]:

In [31]:
```
from sklearn.metrics import mean_squared_error #calculating MSE
mean_squared_error(ypred,y_test)
```

Out[31]: 581887.727391353

In [32]:
```
#from sklearn.metrics import accuracy_score
#accuracy_score(y_test,ypred)
```

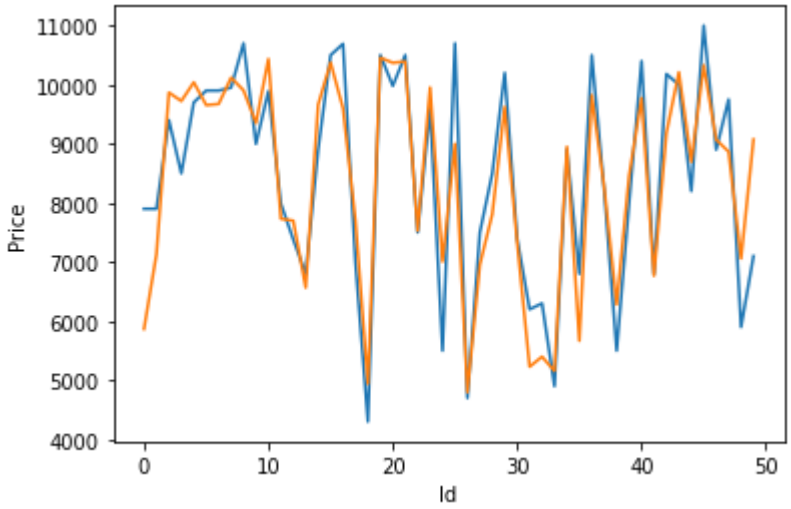
In [33]:
```
#Results= pd.DataFrame(columns=['Actual','Predicted'])
#Results['Actual']=y_test
Results= a.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=ypred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(15)
```

Out[33]:

	index	Price	Predicted	Id
0	481	7900	5867.650338	0
1	76	7900	7133.701423	1
2	1502	9400	9866.357762	2
3	669	8500	9723.288745	3
4	1409	9700	10039.591012	4
5	1414	9900	9654.075826	5
6	1089	9900	9673.145630	6
7	1507	9950	10118.707281	7
8	970	10700	9903.859527	8
9	1198	8999	9351.558284	9
10	1088	9890	10434.349636	10
11	576	7990	7732.262557	11
12	965	7380	7698.672401	12
13	1488	6800	6565.952404	13
14	1432	8900	9662.901035	14

In [34]:
```
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()
```

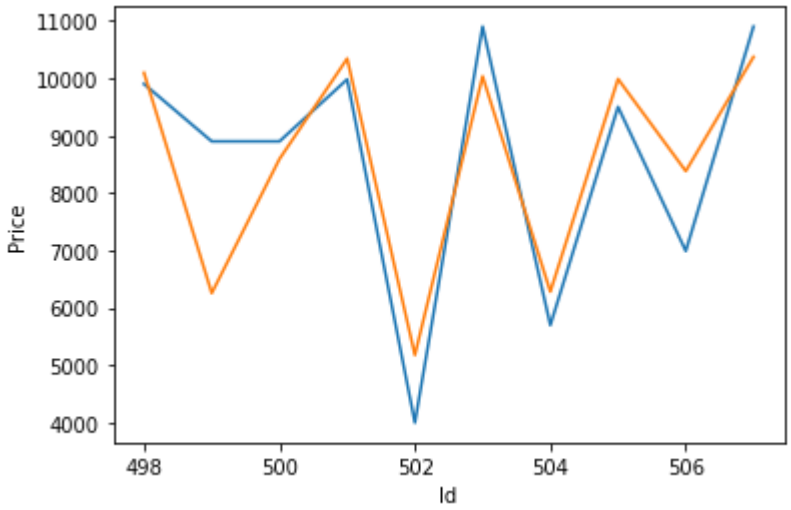
Out[34]: []



In [35]:
```
import seaborn as sns
import matplotlib.pyplot as plt

sns.lineplot(x='Id',y='Price',data=Results.tail(10))
sns.lineplot(x='Id',y='Predicted',data=Results.tail(10))
plt.plot()
```

Out[35]: []



In [ ]:

In [ ]:

In [36]:
```
# ridge regression
```

In [37]:
```
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[37]:
```
GridSearchCV(estimator=Ridge(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20, 30]})
```

In [38]:
```
ridge_regressor.best_params_
```

Out[38]:
```
{'alpha': 30}
```

In [39]:
```
#X_train=[2]
```

In [62]:
```
ridge=Ridge(alpha=30)
ridge.fit(X_train,y_train)
y_pred_ridge=ridge.predict(X_test)
```

In [63]:
```
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[63]:
```
579521.7970897449
```

In [64]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[64]:
```
0.8421969385523054
```

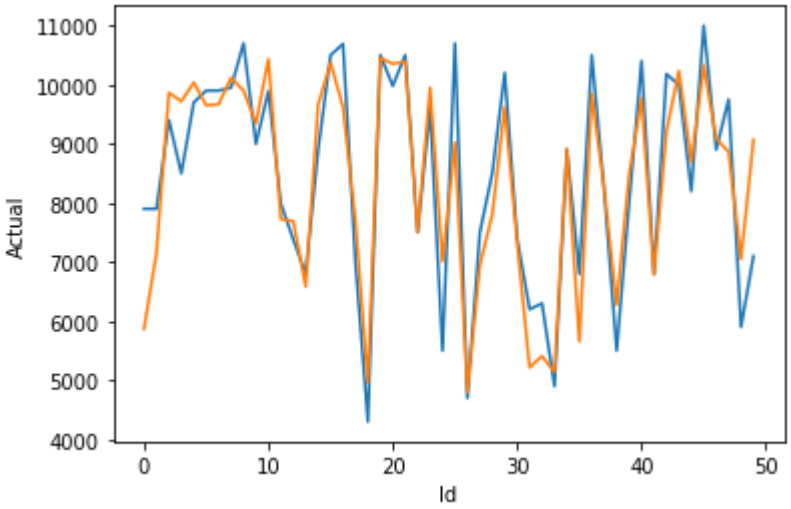
In [65]:
```
Results= a.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred_ridge
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[65]:

	index	Actual	Predicted	Id
0	481	7900	5869.741155	0
1	76	7900	7149.563327	1
2	1502	9400	9862.785355	2
3	669	8500	9719.283532	3
4	1409	9700	10035.895686	4
5	1414	9900	9650.311090	5
6	1089	9900	9669.183317	6
7	1507	9950	10115.128380	7
8	970	10700	9900.241944	8
9	1198	8999	9347.080772	9

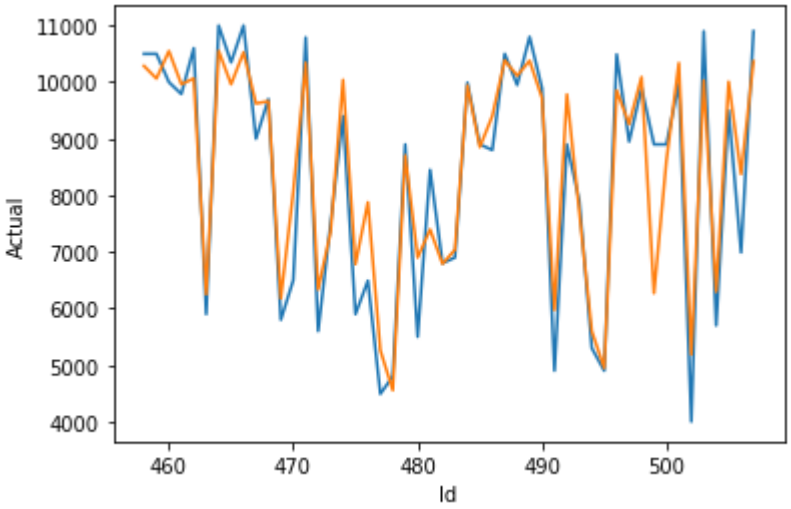
In [66]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[66]: []



In [67]:
```
sns.lineplot(x='Id',y='Actual',data=Results.tail(50))
sns.lineplot(x='Id',y='Predicted',data=Results.tail(50))
plt.plot()
```

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



In [46]:
```
#elastic
```

In [47]:
```
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)
```

Out[47]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

In [48]:
```
elastic_regressor.best_params_
```

Out[48]:
```
{'alpha': 0.01}
```

In [68]:
```
elastic=ElasticNet(alpha=.01)
elastic.fit(X_train,y_train)
y_pred_elastic=elastic.predict(X_test)
```

In [70]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[70]:
```
0.841688021120299
```

In [69]:
```
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[69]:
```
581390.7642825295
```

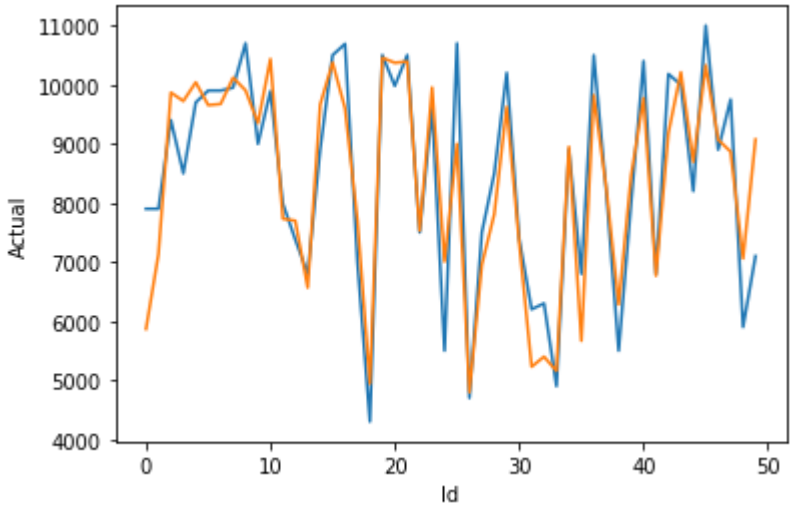
In [51]:
```
Results= a.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred_elastic
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[51]:

	index	Actual	Predicted	Id
0	481	7900	5867.650338	0
1	76	7900	7133.701423	1
2	1502	9400	9866.357762	2
3	669	8500	9723.288745	3
4	1409	9700	10039.591012	4
5	1414	9900	9654.075826	5
6	1089	9900	9673.145630	6
7	1507	9950	10118.707281	7
8	970	10700	9903.859527	8
9	1198	8999	9351.558284	9

In [52]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[52]:
```
[]
```



In [53]:
```
from sklearn.model_selection import GridSearchCV #GridSearchCV is for parameter tuning
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
n_estimators=[25,50,75,100,125,150,175,200] #number of decision trees in the forest, default = 100
criterion=['mse'] #criteria for choosing nodes default = 'gini'
max_depth=[3,5,10] #maximum number of nodes in a tree default = None (it will go till all possible nodes)
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_depth}
RFC_reg = GridSearchCV(reg, parameters)
RFC_reg.fit(X_train,y_train)
```

Out[53]:
```
GridSearchCV(estimator=RandomForestRegressor(),
             param_grid={'criterion': ['mse'], 'max_depth': [3, 5, 10],
                         'n_estimators': [25, 50, 75, 100, 125, 150, 175, 200]})
```

In [54]:
```
RFC_reg.best_params_
```

Out[54]:
```
{'criterion': 'mse', 'max_depth': 5, 'n_estimators': 75}
```

In [71]:
```
reg=RandomForestRegressor(n_estimators=75,criterion='mse',max_depth=5)
```

In [72]:
```
reg.fit(X_train,y_train)
```

Out[72]:
```
RandomForestRegressor(criterion='mse', max_depth=5, n_estimators=75)
```

In [73]:
```
y_pred=reg.predict(X_test)
```

In [74]:
```
y_pred
```

Out[74]:
```
array([ 6052.55942204,  7146.51384386,  9785.3299818 ,  9767.10570666,
       10037.39779793,  9626.40128413,  9753.73850635, 10095.04055737,
        9798.48718081,  9500.79946761, 10386.06232737,  7778.03702259,
        7346.63132259,  7041.1236589 ,  9555.13454358, 10424.01463438,
        9686.03020579,  7824.88263341,  4979.21815036, 10379.69133173,
       10169.73188107, 10434.55516771,  7415.77338692,  9775.00870899,
        7339.48505105,  9136.32624345,  4710.74160089,  7049.55626767,
        7611.54599939,  9569.58447188,  7122.01491277,  5131.58482815,
        5430.87534808,  5041.60810916,  8642.85038042,  5542.16175337,
       10145.40459048,  7370.94782267,  6615.86179881,  8918.10772623,
        9772.97024392,  6743.23314464,  9418.91660991, 10406.43568418,
        8229.11825752, 10451.80869086,  9233.83740643,  8801.46025389,
        6438.99522746,  8892.0336176 ,  9496.8176384 , 10387.37539473,
       10081.59247715,  7114.70511154,  9640.65896762,  9406.73230548,
        9675.96780942, 10431.55297841,  9764.99564074,  7071.51422247,
       10083.56797466,  7149.52858218,  9760.40039044,  7096.08457515,
        5824.62234285,  9673.38989837,  9773.59461844,  8909.68837971,
        8590.20818291,  6307.21084001,  7642.11825699,  6973.48724948,
        8767.2895037 , 10385.25163621,  7135.18306682,  8468.67666715,
```

In [75]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[75]:
```
0.8466322362876058
```

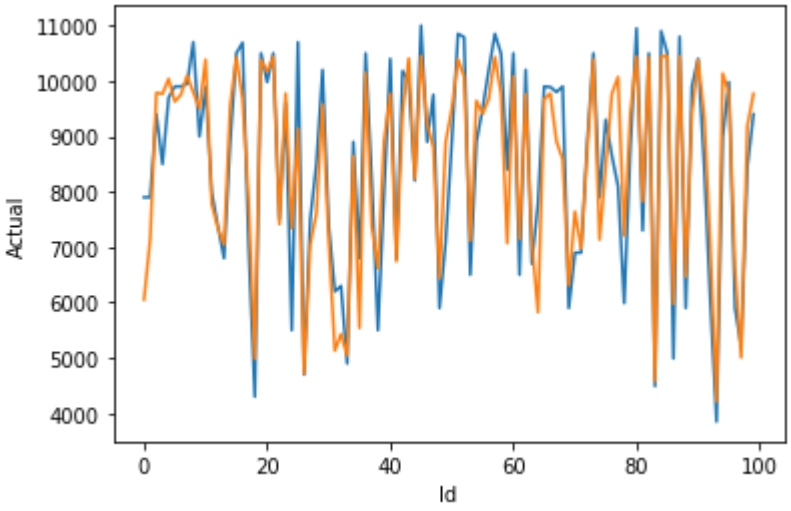
In [76]:
```
Results= a.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[76]:

	index	Actual	Predicted	Id
0	481	7900	6052.559422	0
1	76	7900	7146.513844	1
2	1502	9400	9785.329982	2
3	669	8500	9767.105707	3
4	1409	9700	10037.397798	4
5	1414	9900	9626.401284	5
6	1089	9900	9753.738506	6
7	1507	9950	10095.040557	7
8	970	10700	9798.487181	8
9	1198	8999	9500.799468	9

In [77]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(100))
sns.lineplot(x='Id',y='Predicted',data=Results.head(100))
plt.plot()
```

Out[77]: []



In [ ]:

In [ ]:

In [ ]:

In [ ]: