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

In [2]:
```python
data=a.read_csv("fiat500.csv")
```

In [3]:
```python
data
```

Out[3]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	p
0	1	lounge	51	882	25000	1	44.907242	8.611560	8
1	2	pop	51	1186	32500	1	45.666359	12.241890	8
2	3	sport	74	4658	142228	1	45.503300	11.417840	4
3	4	lounge	51	2739	160000	1	40.633171	17.634609	6
4	5	pop	73	3074	106880	1	41.903221	12.495650	5
...	...	...	...	...	...	...	...	...	
1533	1534	sport	51	3712	115280	1	45.069679	7.704920	5
1534	1535	lounge	74	3835	112000	1	45.845692	8.666870	4
1535	1536	pop	51	2223	60457	1	45.481541	9.413480	7
1536	1537	lounge	51	2557	80750	1	45.000702	7.682270	5
1537	1538	pop	51	1766	54276	1	40.323410	17.568270	7

1538 rows × 9 columns

In [4]:
```python
data.head(5)
```

Out[4]:

	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 [5]:
```python
data.describe()
```

Out[5]:

	ID	engine_power	age_in_days	km	previous_owners	lat	
count	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	153
mean	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	1
std	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	
min	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	
25%	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	
50%	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	1
75%	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	1
max	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	1

In [6]:
```
data.tail(10)
```

Out[6]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	p
1528	1529	lounge	51	2861	126000	1	43.841980	10.51531	5
1529	1530	lounge	51	731	22551	1	38.122070	13.36112	9
1530	1531	lounge	51	670	29000	1	45.764648	8.99450	10
1531	1532	sport	73	4505	127000	1	45.528511	9.59323	4
1532	1533	pop	51	1917	52008	1	45.548000	11.54947	9
1533	1534	sport	51	3712	115280	1	45.069679	7.70492	5
1534	1535	lounge	74	3835	112000	1	45.845692	8.66687	4
1535	1536	pop	51	2223	60457	1	45.481541	9.41348	7
1536	1537	lounge	51	2557	80750	1	45.000702	7.68227	5
1537	1538	pop	51	1766	54276	1	40.323410	17.56827	7

In [7]:
```
data['model'].unique()
```

Out[7]:
```
array(['lounge', 'pop', 'sport'], dtype=object)
```

In [8]:
```
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 [9]:
```
data.groupby(['model']).count()
```

Out[9]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
model								
lounge	1094	1094	1094	1094	1094	1094	1094	1094
pop	358	358	358	358	358	358	358	358
sport	86	86	86	86	86	86	86	86

In [10]:
```
data.groupby(['previous_owners']).count()
```

Out[10]:

	ID	model	engine_power	age_in_days	km	lat	lon	price
previous_owners								
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 [11]:
```
data['model'].unique()
```

Out[11]:
```
array(['lounge', 'pop', 'sport'], dtype=object)
```

In [12]:
```
data.shape
#df=data
#data=df.loc[(df.model=='lounge')&(df.previous_owners==1)]
```

Out[12]:
```
(1538, 9)
```

In [13]:
```
data1=data.drop(['lat','ID'],axis=1) #unwanted columns removed
```

In [14]:
```
data2=data1.drop('lon',axis=1)
```

In [15]:
```
data2.shape
```

Out[15]:
```
(1538, 6)
```

In [16]:
```
data2.head(3)
```

Out[16]:

	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

In [17]:
```
data2=a.get_dummies(data2,dtype=int)
```

In [18]:
```
data2.shape
```

Out[18]:
```
(1538, 8)
```

In [19]:
```
data2.head(3)
```

Out[19]:

	engine_power	age_in_days	km	previous_owners	price	model_lounge	model_pop	model_
0	51	882	25000	1	8900	1	0	
1	51	1186	32500	1	8800	0	1	
2	74	4658	142228	1	4200	0	0	

In [20]:
```
y=data2['price']
X=data2.drop('price',axis=1)
```

In [21]:
```
y
```

Out[21]:
```
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 [22]:
```
X
```

Out[22]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_spo
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	

1538 rows × 7 columns

In [23]:
```
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_sta
```

In [24]:
```
X_test.head(5)
```

Out[24]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_spor
776	51	762	17000	1	1	0	
487	51	425	20636	1	1	0	
1462	62	3470	90000	1	0	1	
89	51	397	17912	1	1	0	
852	51	1035	33000	1	1	0	

In [25]:
```
X_train.shape
```

Out[25]:
```
(1030, 7)
```

In [26]:
```
y_train.shape
```

Out[26]:
```
(1030,)
```

In [27]:
```
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[27]:
```
▾ LinearRegression
LinearRegression()
```

In [28]:
```
ypred=reg.predict(X_test)
```

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

```
Out[29]: array([10077.0486545 , 10296.89113709,  6231.54053645, 10371.87050424,
          9543.8908106 , 10311.36861938,  8883.57598947,  7157.79300792,
          9944.27338867, 10426.6142839 ,  9912.02921839,  7492.70862718,
          9882.00387912,  6645.64608231, 10333.80213676,  8020.94888715,
         10228.57854658, 10530.25049259,  9188.10296552,  8658.91690616,
          7685.86894274,  9410.15189019,  8825.54005856,  9911.92508766,
         10103.04304765,  6699.79552359,  6254.1941181 ,  6353.64026044,
          4685.57678728, 10147.55738739,  9412.7006505 ,  4999.96883869,
          9396.11094252,  9701.00047514,  8245.07179202,  7014.74657124,
          9855.0499626 ,  8022.65546956,  6202.74656775, 10387.44465039,
          7477.65149429,  6555.14982661, 10368.68963757,  5057.57213783,
          9652.3799067 ,  9312.51608202,  8642.27980214,  9966.91032444,
          5663.75605319,  7526.68879785,  4958.61357652,  9346.67438618,
         10022.93146835, 10140.06080681,  6436.7985707 ,  5851.7441137 ,
          6827.22453472,  9850.66079486,  9880.32481877, 10437.9205592 ,
          9556.18602801,  7858.22024074, 10284.19351479, 10377.9052516 ,
          9964.8785774 ,  6982.85702955,  7664.86130973,  5224.84497605,
         10013.44733777,  6661.7812147 ,  5396.11965442,  6361.46202604,
          4401.88640849,  9772.32940609,  7414.07382234,  8897.76480227,
          5273.593021  ,  7056.67723507,  6741.4936419 ,  7918.05381693,
          5262.41488583,  6877.92504811,  6228.69809605,  8148.13997191,
          7263.5409503 ,  7473.63513447,  9826.20224457,  8897.76480227,
          9557.98317895,  9012.11120767, 10433.53601571,  6920.74758333,
          7936.69809147,  7795.72328257,  9622.06761789,  9945.83776355,
          9101.23004767,  8550.37389002,  9752.79218725,  6675.66063548,
          5707.55785515,  9036.4887014 ,  9427.66803929,  9664.73672275,
         10359.44492193, 10333.80213676, 10426.38938649, 10273.40271038,
          8580.6495184 , 10245.37375553,  9460.08517598,  6903.0439201 ,
          7759.02278758,  6308.83085321,  9973.98520286, 10182.60390159,
          9311.53993102, 10312.75644294,  5084.71810292,  4206.10307695,
         10375.87209307,  7835.20847593,  9829.10883449, 10071.04691043,
          5449.23018415,  9662.94272348,  9702.35081579,  7467.37810164,
          9501.86990917, 10369.86516893,  9101.74172436, 10023.162502  ,
          4495.56790401,  8825.05826205,  9936.74106722,  7019.45945617,
          9742.69668115,  9624.59771383,  4414.21892378, 10054.21095081,
          6714.78810684,  9920.31907615,  9370.70092693,  6575.63342387,
          6274.9133057 , 10281.46692255, 10340.50467635,  9116.64181116,
          6948.55023069,  6980.50146676, 10124.22428034,  6866.52382434,
         10509.16129574, 10151.68050671,  7005.03051233,  6153.8725305 ,
         10065.98671854,  8797.24151621, 10560.73777011, 10369.1019495 ,
          7726.96172248,  8635.11704424,  9644.45626512,  6394.57209121,
          9932.73947838,  9811.65908623,  9762.77076504, 10358.00701025,
          7139.13858818,  5029.24520368,  9100.61157977,  9880.51223329,
          7939.71080918, 10384.27734509,  6683.49610315,  9952.09748542,
          8842.70722808, 10076.08657336,  9912.93693368, 10357.28517541,
          6365.74719808,  7171.88810656,  8060.13837244,  8250.76388839,
          6787.4466476 ,  9739.00660461,  9969.09789418,  9789.77719657,
          9495.26479954, 10339.08722606,  6625.36952009,  7061.57841357,
          9623.64189981,  6921.78496788,  3232.37870456,  9398.0554513 ,
          9952.09748542,  8484.3127639 , 10307.57376441, 10265.72223874,
         10578.36841841,  9820.47370464,  9952.09748542, 10248.76122518,
          8821.39177649,  9734.29142913, 10031.45588321,  9931.61250109,
         10379.81687965,  7708.65477476,  9791.21369676, 10064.18293703,
          9436.0295791 , 10487.04638303,  4829.05570411,  6049.1016172 ,
         10090.60390397,  4256.46791286,  8048.81596848,  7254.09460377,
          8691.08558652,  7015.36320392,  4565.8697124 ,  6537.07872561,
          9763.71845344,  7358.30746556,  5390.16165428,  7299.4887424 ,
         10204.74382286,  8559.91112305,  9791.21369676, 10440.32854676,
          9252.31565776, 10291.62478924,  9111.77816936,  8083.26519264,
          9952.09748542,  9625.1922602 ,  5447.67746443,  9880.27485544,
          9782.22439163,  9503.91964339,  6434.64936292,  9644.45626512,
          7099.76674587, 10502.1894758 ,  8699.65825429, 10337.99114011,
         10348.65316137,  6964.73698525,  6148.81405855,  9012.11120767,
          6436.17574567,  8825.54005856, 10353.63154853,  9643.49204243,
```

```
 9952.09748542,  6948.39355333, 10581.4032754 ,  6752.3282492 ,
 5509.92811393, 10453.58841261,  5626.50283578,  9992.28093873,
10248.70500083,  5428.71684814,  6738.66197155,  6716.0887264 ,
 7754.28298788,  4756.91807609,  4603.09010115,  9343.94440138,
10167.73853803,  8656.5851236 , 10404.97756716, 10298.56820804,
10491.35691687,  9037.61832309, 10151.05237913,  5114.62804797,
 9872.52497631,  9841.51127284,  9820.82071044, 10102.57790392,
 9063.50180159,  6798.87443119,  9267.77582114, 10133.96318853,
 9999.5061881 ,  8565.19548027,  7722.12567984, 10055.36479518,
 9712.98534438,  7611.7711265 , 10395.12607818,  9620.19347274,
 9398.9675153 , 10369.95439659,  9775.03622363,  9593.02752701,
 7201.09750911,  9662.68748325,  4847.46205228,  6477.93510047,
10122.29051828,  9930.32823682,  9313.67355864,  5895.44822231,
 9048.63348639,  9863.08716305,  6727.69792206,  6512.6845661 ,
 5957.29501208,  5191.95679688,  8542.59963404,  9688.85875217,
 5850.20848648,  9977.62429844,  9682.0422519 , 10185.40111091,
 9704.82543556,  9799.8616315 , 10243.13878975,  6794.12141897,
 8314.39015989,  7762.96669509,  9742.69668115,  4332.51396391,
10512.71079189, 10465.4840542 , 10112.91071079,  9912.02921839,
10405.83001425, 10482.72618952,  7868.62760577,  9677.85573876,
10452.93694141,  6334.25148463, 10395.32124007,  7925.92062132,
 7686.86804535,  9989.14676763,  6910.58754398,  9904.74787555,
 9925.10262587,  5769.52460242,  4747.64113472,  6678.45970331,
 9997.75645567,  9977.82079473,  9894.03589094, 10244.83646675,
10359.41261911, 10077.18502473, 10155.56395642,  9844.41006158,
 9671.63971311,  8846.10235163,  6707.97759632,  8663.72211126,
10395.12607818,  9756.49947118, 10527.82548352,  8389.29199393,
 5680.6830438 ,  7682.91650384, 10226.09501531, 10395.39620588,
 6511.28708977,  4368.05713248, 10291.94339391,  9237.16308169,
10406.76536688, 10358.00701025,  6463.65823191, 10260.36691668,
10546.64419862, 10341.1209625 ,  8441.73935569,  5620.43361035,
 9742.06913148, 10432.28846409,  9898.57368152,  6687.31099545,
 5449.30701456,  9662.27275819,  7388.84124404,  8872.79965489,
 8785.05852344,  7720.78580083,  9269.41043103,  9046.45122031,
 9723.11282054,  9850.64744146, 10183.79170541,  5082.22786482,
 4829.05570411,  9716.32944836,  7935.96816077, 10468.57191408,
 7220.86489162,  6132.1047393 ,  5679.60112901,  9880.27485544,
 6932.31134191, 10164.44004258,  7309.61717471,  6448.30264497,
 9930.40396642,  9510.27299543, 10073.48329912,  9750.04677648,
10353.35395021,  6321.31286784,  7711.80067415,  5725.13591781,
10062.05549334,  9937.48010945,  7075.16435472,  9762.09760684,
 8920.1484395 ,  7347.44629752,  6773.49287052,  4289.71938476,
 6587.8866172 , 10322.78216331,  8602.2243167 ,  9641.29634707,
 4991.00005762, 10394.7680783 ,  5477.10493576,  9318.99339328,
 8969.98448728, 10054.81067725,  9027.45307896,  7810.04203451,
 9066.51469567,  7959.99454088,  9851.71570419, 10470.01040367,
10413.55885407,  7714.59974198,  9612.57926338,  9966.27606064,
 9414.81272201,  9741.77374786,  4781.84518595,  8523.48845655,
 6275.16403695,  9821.85149027,  8040.51167027,  7118.32222368,
10002.08284872,  8100.67364171,  5884.01603011,  8531.33485759,
 9816.35058532,  9037.0576136 ,  9488.6399785 ,  7114.18236839,
10350.77728959,  9878.79909092, 10353.7850036 ,  9383.86555836,
 7525.1903001 ,  6598.97578895,  8933.03394775, 10132.83038346,
 9871.62264908, 10090.63959936,  9108.26732791,  9171.65615057,
 5928.70196285,  5598.4103406 ,  9722.5683623 ,  4476.21769815,
 3280.24241805,  9905.33852022,  9536.10050303, 10047.31563073,
 6583.25056268, 10140.95700392,  8756.70270741,  6762.15097693,
 9715.7748751 ,  9577.97338515,  9712.28378506,  8334.56952302,
 9810.06606286, 10364.14902624,  9058.14123004,  8834.05672604,
10165.81156109,  9780.46422926,  9702.35081579,  6519.29507068,
 9836.19164617, 10022.35048335,  9724.61470396,  9913.73315641,
 7849.13469923, 10488.45199189,  7745.18732479,  8388.7875088 ,
10322.537953  ,  7570.91612614,  9239.03071235, 10439.86449007])
```

In [30]:
```
filename='pricemodeldummy1'
pickle.dump(reg,open(filename,'wb'))
```

In [31]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[31]: 0.8432871743994133

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

Out[32]: 577671.0281058006

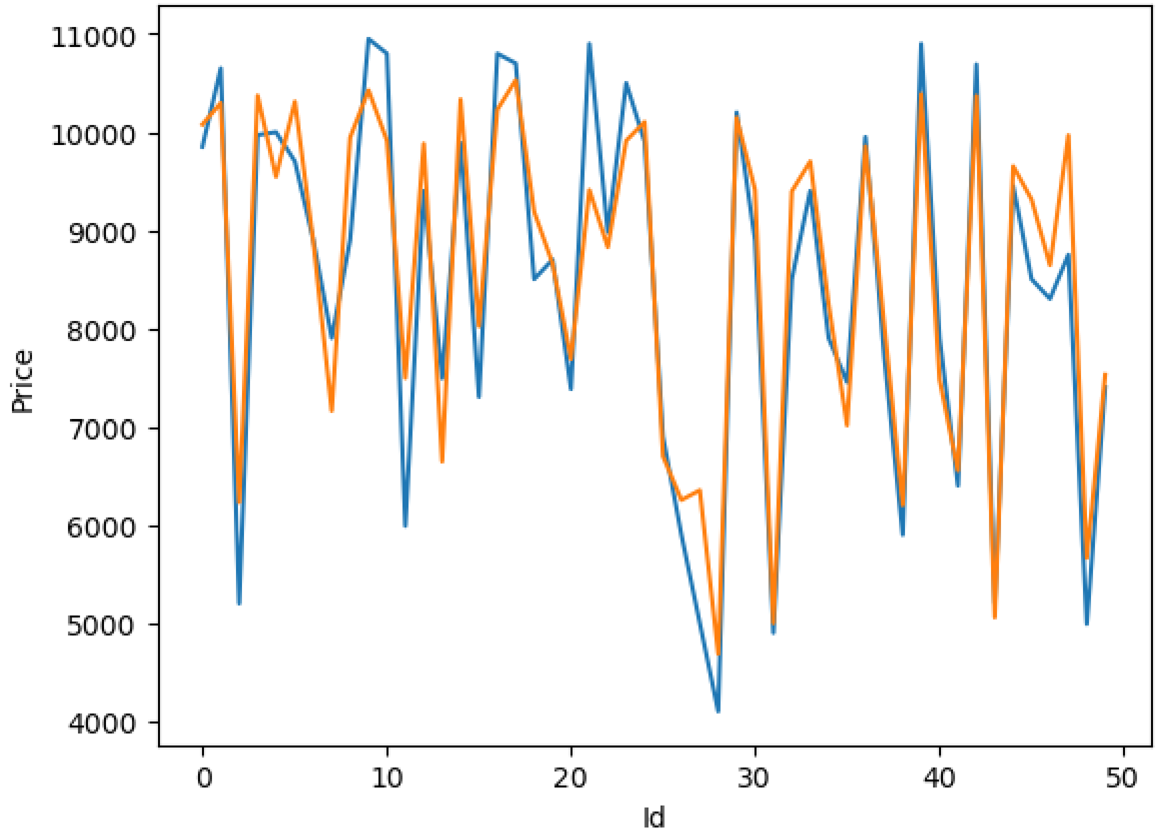
In [33]:
```
#Results= pd.DataFrame(columns=['Actual','Predicted'])
#Results['Actual']=y_test
Results= a.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=ypred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(15)
```

Out[33]:

	index	Price	Predicted	Id
0	776	9850	10077.048654	0
1	487	10650	10296.891137	1
2	1462	5199	6231.540536	2
3	89	9970	10371.870504	3
4	852	9999	9543.890811	4
5	12	9700	10311.368619	5
6	353	8900	8883.575989	6
7	76	7900	7157.793008	7
8	633	8900	9944.273389	8
9	181	10950	10426.614284	9
10	1111	10800	9912.029218	10
11	368	5990	7492.708627	11
12	1298	9400	9882.003879	12
13	1361	7490	6645.646082	13
14	713	9890	10333.802137	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()
```

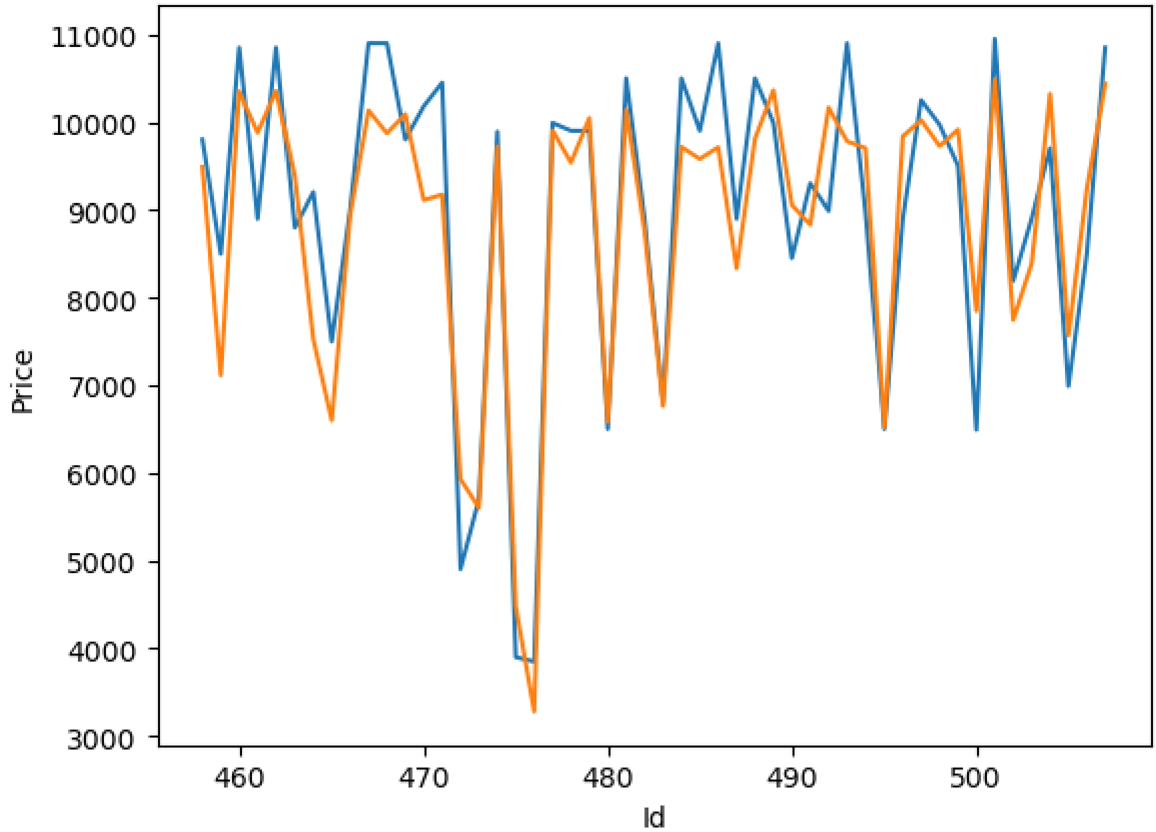
Out[34]: []



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

sns.lineplot(x='Id',y='Price',data=Results.tail(50))
sns.lineplot(x='Id',y='Predicted',data=Results.tail(50))
plt.plot()
```

Out[35]: []



In [36]:
```
new=[[51,2197,70000,1,1,0,0]]
```

In [37]:
```
real=reg.predict(new)
```

In [38]:
```
real
```

Out[38]:
```
array([7857.45949044])
```

In [39]:
```
from sklearn.model_selection import GridSearchCV
#from sklearn.grid_search 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[39]:
```
▸ GridSearchCV
▸ estimator: Ridge
    ▸ Ridge
```

In [40]:
```
ridge_regressor.best_params_
```

Out[40]:
```
{'alpha': 30}
```

In [41]:
```
ridge=Ridge(alpha=30)
ridge.fit(X_train,y_train)
y_pred_ridge=ridge.predict(X_test)
```

In [42]:
```
from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[42]:
```
578069.134875448
```

In [43]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_ridge)
```

Out[43]:
```
0.8431791744587434
```

In [44]:
```
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)
```

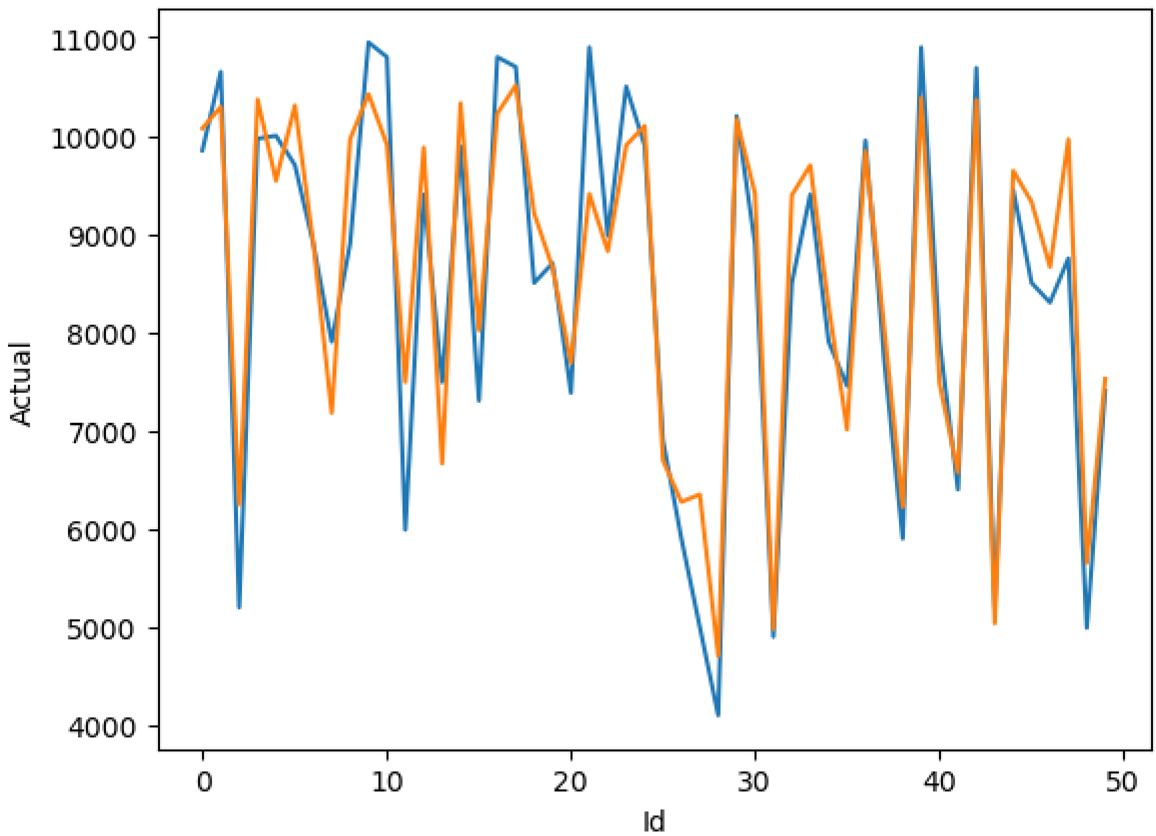
Out[44]:

	index	Actual	Predicted	Id
0	776	9850	10073.489785	0
1	487	10650	10293.318926	1
2	1462	5199	6250.181303	2
3	89	9970	10368.300682	3
4	852	9999	9540.320853	4
5	12	9700	10307.799776	5
6	353	8900	8902.872087	6
7	76	7900	7176.381624	7
8	633	8900	9963.615342	8
9	181	10950	10423.047838	9

In [45]:

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

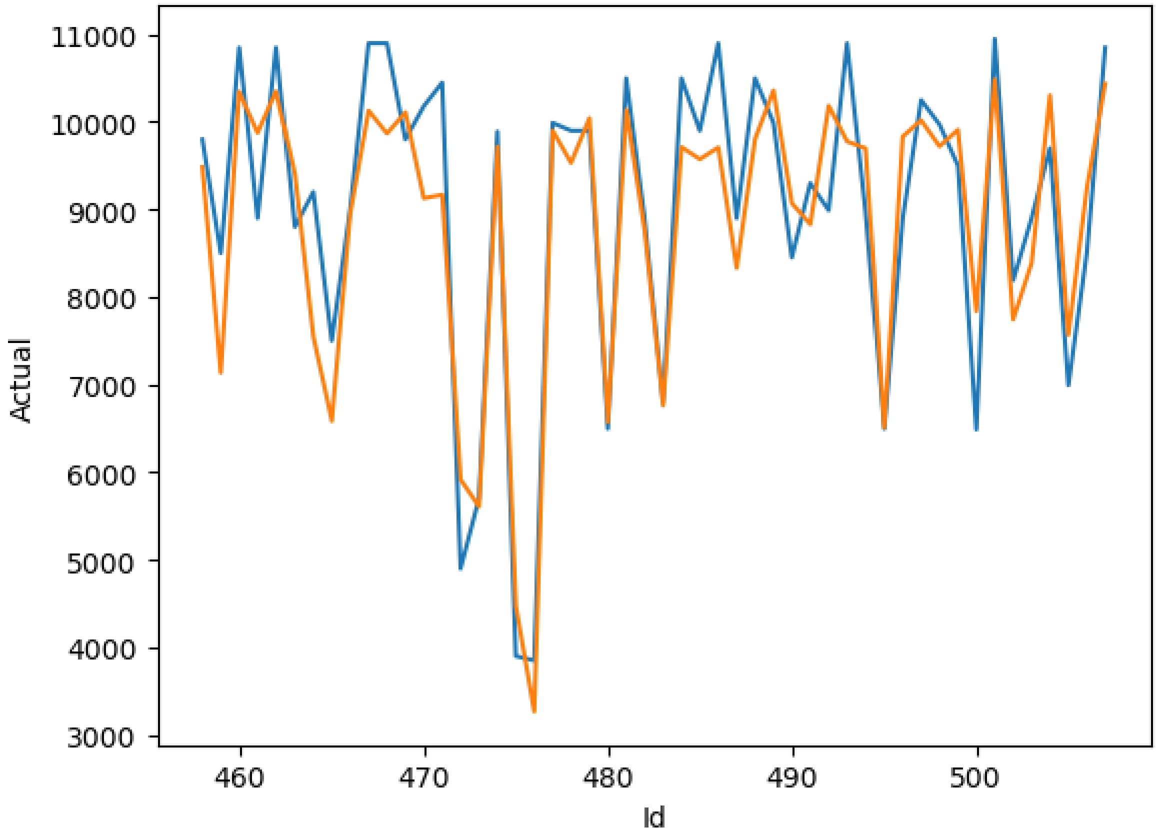
Out[45]: []



In [46]:

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

Out[46]: []



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
ElasticNet
```

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

Out[48]:
```
{'alpha': 1}
```

In [49]:
```
elastic=ElasticNet(alpha=1)
elastic.fit(X_train,y_train)
y_pred_elastic=elastic.predict(X_test)
```

In [50]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[50]:
```
0.8412324009157849
```

In [51]:
```
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[51]: 585245.2844327039

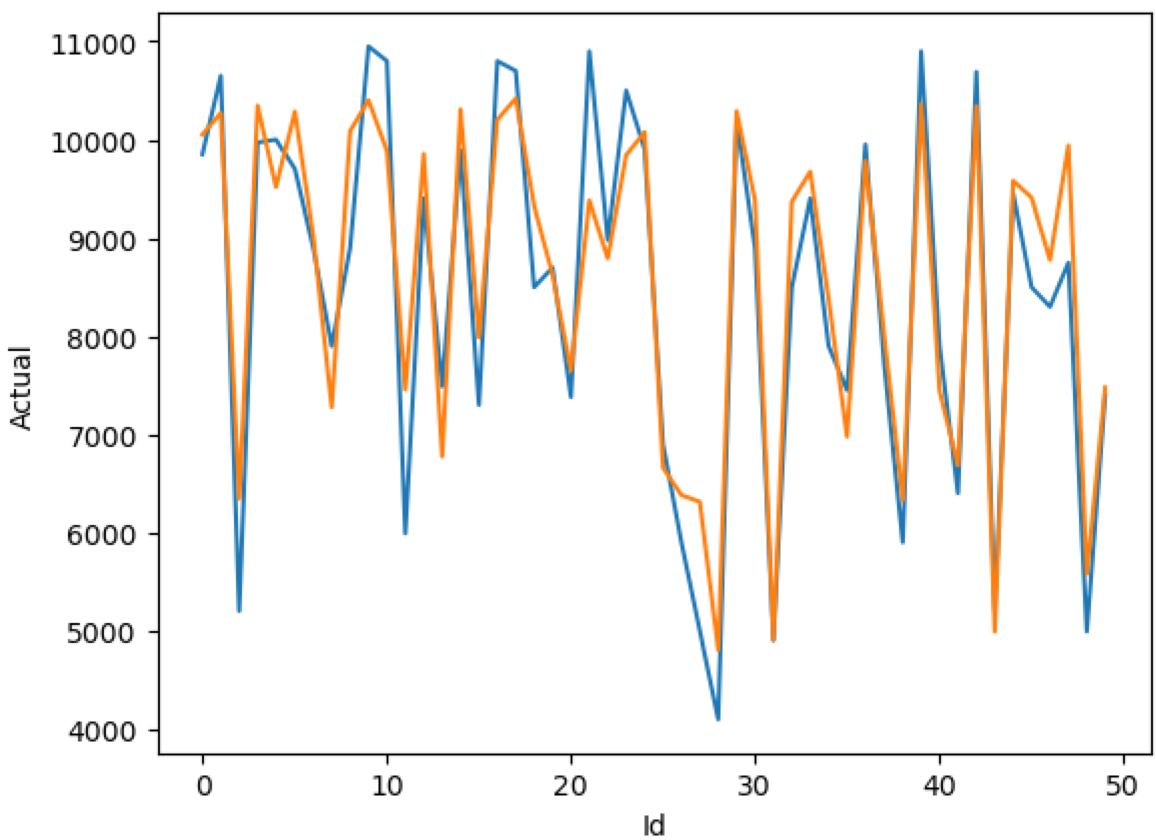
In [52]:
```
Results= a.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred_elastic
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[52]:

	index	Actual	Predicted	Id
0	776	9850	10050.384502	0
1	487	10650	10270.923705	1
2	1462	5199	6346.052383	2
3	89	9970	10346.210602	3
4	852	9999	9515.099320	4
5	12	9700	10285.489841	5
6	353	8900	9020.469803	6
7	76	7900	7275.281217	7
8	633	8900	10085.639183	8
9	181	10950	10401.195353	9

In [53]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[53]: []



In [54]:
```
from sklearn.model_selection import GridSearchCV #GridSearchCV is for parameter tur
from sklearn.ensemble import RandomForestRegressor
reg=RandomForestRegressor()
n_estimators=[25,50,75,100,125,150,175,200] #number of decision trees in the forest
criterion=['squared_error'] #criteria for choosing nodes default = 'gini'
max_depth=[3,5,10] #maximum number of nodes in a tree default = None (it will go ti
parameters={'n_estimators': n_estimators,'criterion':criterion,'max_depth':max_dept
RFC_reg = GridSearchCV(reg, parameters)
RFC_reg.fit(X_train,y_train)
```

Out[54]:
```
GridSearchCV
estimator: RandomForestRegressor
RandomForestRegressor
```

In [55]:
```
RFC_reg.best_params_
```

Out[55]:
```
{'criterion': 'squared_error', 'max_depth': 5, 'n_estimators': 100}
```

In [56]:
```
reg=RandomForestRegressor(n_estimators=200,criterion='squared_error',max_depth=5)
```

In [57]:
```
reg.fit(X_train,y_train)
```

Out[57]:
```
RandomForestRegressor
RandomForestRegressor(max_depth=5, n_estimators=200)
```

In [58]:
```
ypred=reg.predict(X_test)
ypred
```

```
Out[58]: array([ 9995.23426034, 10477.64628023,  5912.20394486, 10473.78392857,
         9766.67200633, 10504.81783578,  8526.12140348,  7154.43644499,
        10197.83609877, 10456.81457714, 10015.22570378,  7413.4366462 ,
         9938.23098777,  7384.77537857, 10494.98347804,  7531.95167065,
        10372.27620424, 10478.37548461,  8819.624903  ,  8791.6207983 ,
         7520.34389517,  9371.78871194,  8820.43382862,  9977.47703656,
         9998.45158645,  6775.34294943,  6445.92368978,  5929.3529609 ,
         4966.33683069, 10107.7241266 ,  9479.00998064,  4835.15405904,
         9483.86933153,  9487.91853591,  8374.95533194,  7235.26794517,
         9807.80282218,  7704.97065764,  6348.35109166, 10465.39485911,
         7588.51814754,  6964.37585026, 10491.27618607,  5195.35832357,
         9509.90118068,  9203.72266327,  8426.99266909,  9774.64038022,
         5518.82507673,  7326.15919988,  5040.8023411 ,  9234.14381218,
         9943.5014854 , 10090.95195136,  6250.24372843,  5714.27532462,
         7131.11872556, 10207.89967891,  9748.34983478, 10415.36508144,
         9692.98842809,  7844.16508357, 10477.34193983, 10473.78392857,
        10004.65939515,  7261.3490691 ,  7396.49945349,  5094.07178972,
        10130.36745025,  7068.38248001,  5084.16909068,  6318.11902803,
         4654.1026874 ,  9732.27739282,  7321.26267947,  8511.82751952,
         5032.84181751,  7098.59273257,  6644.56257081,  7805.02744264,
         5082.64352479,  7133.49106248,  5862.64916648,  7883.49649763,
         7328.95669938,  7359.75628282, 10140.97314901,  8511.82751952,
         9297.81641656,  8784.64524025, 10437.80815122,  7132.04082305,
         7710.22174802,  7738.6833708 ,  9520.23261911,  9893.43637371,
         9641.11650249,  8653.2593225 ,  9503.8667286 ,  6823.14375594,
         6318.21596211,  9228.97809103,  9175.64002889,  9544.84452327,
        10475.22921686, 10494.98347804, 10456.81457714, 10330.8090257 ,
         8031.61317923, 10255.01879471,  9744.43288583,  6688.2438553 ,
         7723.31062116,  6018.87865792,  9932.44654983, 10104.49208071,
         9305.36996931, 10390.41849533,  5100.80185275,  4571.20792236,
        10456.02994833,  7793.45513761, 10178.46082833, 10212.30334664,
         5541.27149694,  9542.81360944,  9511.25458011,  7591.05124165,
         9694.46977273, 10473.78392857,  8960.83210938,  9884.10761189,
         4774.94199559,  9111.99606033,  9952.14241426,  7371.20606919,
         9817.1043073 ,  9573.74710301,  4927.94624491, 10014.55958396,
         6754.72397653,  9739.55417908,  9301.48231811,  6024.86559263,
         6140.53940562, 10480.85220857, 10367.30803654,  8796.34507525,
         7220.70030279,  6734.66736393, 10078.65967395,  7327.92228573,
        10383.58764441, 10113.44198375,  7456.59255207,  6087.57449996,
        10188.22011238,  9640.0699366 , 10370.13838968, 10491.27618607,
         7859.31993059,  8430.13881437,  9659.54820929,  6164.89845808,
         9952.63779131,  9770.14268661,  9538.50289349, 10498.80151523,
         7335.33932885,  4950.91319179,  9641.11650249,  9748.34983478,
         7909.08946214, 10465.39485911,  6468.69094912,  9940.05795471,
         8813.60317556, 10030.98862129,  9925.08956802, 10488.14974914,
         6306.71657958,  7144.73108823,  7803.96340906,  7834.7653002 ,
         6733.4730245 ,  9503.41044933,  9946.64210768,  9514.34799664,
         9671.98770385, 10493.19121644,  7117.53512876,  7152.1335018 ,
         9570.71367347,  6797.23273744,  4401.89433475,  9242.30973807,
         9940.05795471,  8649.14627056, 10475.03178831, 10194.59691193,
        10259.56748327,  9782.37393125,  9940.05795471, 10123.59509806,
         8942.11426469,  9716.42287911,  9830.74284563, 10009.62322077,
        10472.09192253,  7442.69090771,  9826.37130047,  9965.21974798,
        10180.77262401, 10383.69823623,  4939.86504862,  5832.70806849,
        10140.0399975 ,  4758.37783354,  7822.17019008,  7126.79669508,
         7954.93995002,  7376.19957154,  4520.25596566,  5966.39258927,
         9510.91315213,  7315.29334508,  5257.88281591,  7194.57446814,
        10237.0932625 ,  8452.16047517,  9826.37130047, 10415.65715827,
         8888.62688395, 10477.64628023,  9223.65724995,  7903.29735514,
         9940.05795471,  9496.5200667 ,  5314.64861026, 10079.97164334,
         9528.38826658,  9523.25923888,  6283.04235566,  9659.54820929,
         7446.39623041, 10383.58764441,  8943.23363064, 10504.81783578,
        10493.90445307,  7130.91249042,  6149.58261361,  8784.64524025,
         6614.04956341,  8820.43382862, 10388.67906332,  9489.37866877,
```

```
 9940.05795471,  6059.98160351, 10480.09149017,  7112.21665063,
  5282.42103571, 10413.03521184,  5496.29564013, 10105.90061903,
 10123.59509806,  5374.89518976,  6360.94747499,  6729.03319071,
  7720.30516615,  5027.59341548,  4837.8847493 ,  9226.6616469 ,
 10226.21696175,  8919.89321448, 10473.78392857, 10487.44578543,
 10385.91751401,  8790.60951733, 10113.44198375,  5037.18843207,
 10014.53000818,  9756.16679874,  9798.29606979, 10029.16140311,
  8920.86495091,  7119.78166158,  8846.82005965, 10003.78446245,
  9916.73833987,  9427.44415594,  7587.42624403,  9996.94985575,
  9540.38959434,  7830.49022711, 10302.22310183,  9516.67852921,
  9307.69956193, 10486.40319606, 10172.45078321,  9451.8775657 ,
  7434.52308539, 10193.0363768 ,  5040.8023411 ,  6607.64061956,
 10222.45677301,  9921.03205572,  9685.18014606,  5714.56669271,
  8844.77080689,  9975.15743046,  7205.07727335,  7106.79791138,
  5737.17715038,  5091.97530757,  8558.73919055,  9934.26411986,
  6088.72104318,  9999.44011915,  9565.58836734, 10220.55460539,
 10164.42800307,  9503.63145128, 10121.21090361,  7255.52434755,
  8365.98419549,  8133.02587967,  9817.1043073 ,  4582.73994677,
 10482.09598708, 10385.91751401, 10010.59391193, 10015.22570378,
 10460.27713578, 10383.58764441,  7702.47027119,  9539.63011085,
 10456.81457714,  6146.86162247, 10486.40319606,  7909.5621729 ,
  7557.77564372,  9825.93839727,  7366.58009258,  9973.8232826 ,
  9952.14241426,  5703.81684094,  4909.31547399,  6965.19219735,
  9832.41602763,  9957.89057329,  9915.58466649, 10460.83167046,
 10498.80151523,  9860.32840205, 10012.04066715,  9814.65046252,
  9761.55015367,  8337.77946488,  6892.17123461,  7960.34485612,
 10302.22310183,  9797.05175581, 10403.9430184 ,  7843.48068494,
  5675.95581416,  7773.4803391 , 10374.32793832, 10486.40319606,
  6530.37431421,  4653.42306231, 10477.64628023,  9702.27475563,
 10471.45980598, 10498.80151523,  5866.62770776, 10300.58357736,
 10385.97183886, 10503.37254749,  8666.67370181,  5081.01649499,
  9727.60772279, 10415.65715827, 10006.21775582,  6115.65953208,
  5376.13485017,  9984.73969139,  7268.02189125,  8526.12140348,
  8820.43382862,  7396.49945349,  9705.37997486,  9747.96731007,
  9754.42159509,  9778.4432165 , 10246.39606104,  4986.94136001,
  4939.86504862,  9499.83122441,  7769.50348237, 10418.04135271,
  7396.61023364,  5766.29982884,  5139.98843285, 10079.97164334,
  6923.25780984, 10226.21696175,  7365.99382379,  6249.34966973,
  9946.78084849,  9654.13915496, 10212.30334664,  9786.8023957 ,
 10449.13110462,  6343.80683554,  7612.76569007,  5904.43753409,
  9996.94985575, 10009.49752484,  7304.25499385,  9516.54811019,
  8728.70704192,  7647.93702102,  7085.63938132,  4793.45394077,
  7111.25356382, 10503.37254749,  8633.50723872,  9559.32964597,
  4899.93451173, 10445.08801088,  5470.80870028,  9500.80457214,
  7974.1062478 , 10013.01256015,  9651.53304415,  7744.58062592,
  8927.49059256,  7848.90711405,  9778.4432165 , 10413.68250884,
 10466.58326041,  7352.38879342,  9492.94084178,  9957.89057329,
 10180.37706763,  9514.87329554,  4926.68755755,  7950.25616907,
  6797.9178531 ,  9792.59627567,  8274.18189184,  7094.97897206,
  9956.82666223,  7713.51187929,  6165.73437276,  8575.33479179,
  9786.26018045,  8877.65892279,  9694.46977273,  7348.99668254,
 10448.54059464,  9944.69225813, 10449.13110462,  9121.04898715,
  7733.40576486,  5959.639328  , 10108.66803009, 10275.06746616,
  9923.07027599, 10039.47394279,  8866.08393886,  9227.38520936,
  5714.63690546,  5665.71753315,  9822.31258469,  4616.75685062,
  4263.64359356, 10010.57366174,  9726.94557477,  9960.19263248,
  6453.87222145, 10210.09502584,  8814.41306563,  7138.77790676,
  9661.00517835,  9733.72826996,  9520.26829554,  8466.38644983,
  9700.81322669, 10473.78392857,  8735.26200569,  8955.76777479,
 10113.44198375,  9500.01273585,  9511.25458011,  6416.27340918,
  9926.38552929,  9952.5006104 ,  9493.5817973 ,  9947.16842563,
  7666.59116338, 10385.91751401,  7574.99712246,  7867.38291753,
 10278.18221159,  7375.327634  ,  9237.73282769, 10413.68250884])
```

In [59]:
```
#from sklearn.ensemble import RandomForestRegressor
#from sklearn.datasets import make_regression
#x, y = make_regression(n_features=4, n_informative=2,random_state=0, shuffle=False
#regr = RandomForestRegressor(max_depth=2, random_state=0)
#regr.fit(X_train, y_train)
```

In [60]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[60]:
```
0.834912603724135
```

In [ ]: