In [29]:
```
import pandas as pd
import pickle
```

In [35]:
```
data=pd.read_csv("/home/manu/Desktop/online/fiat500.csv")
```

In [36]:
```
data.head()
```

Out[36]:

	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 [32]:
```
#data.describe()
list(data)
```

Out[32]:
```
['ID',
 'model',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

In [33]:
```
data1['model'] = data['model'].map({'lounge':1,'pop':2,'sport':3})
```

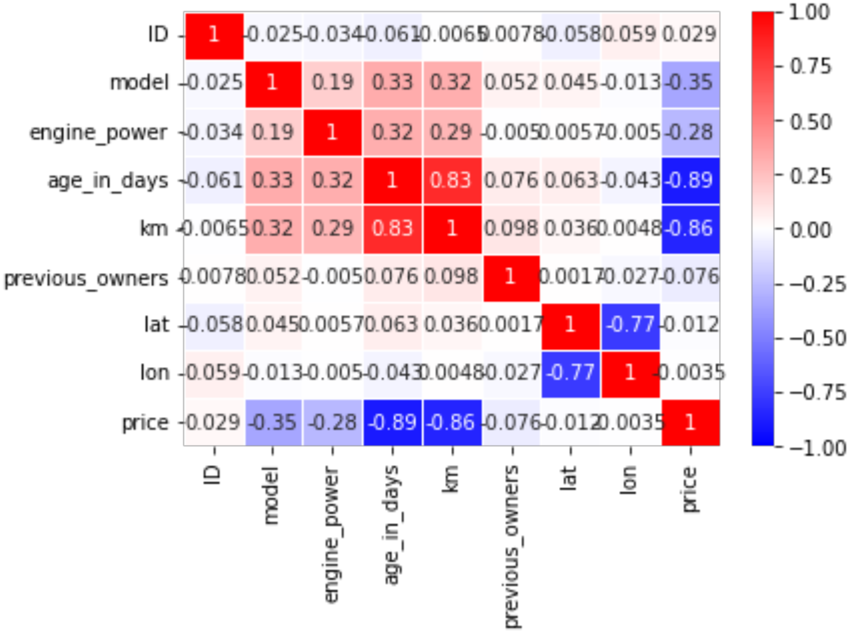
In [38]:
```
cor=data1.corr()
cor
```

Out[38]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1.000000	-0.024740	-0.034059	-0.060753	-0.006537	0.007803	-0.058207	0.058941	0.028516
model	-0.024740	1.000000	0.189906	0.326508	0.319580	0.052480	0.044901	-0.013200	-0.349885
engine_power	-0.034059	0.189906	1.000000	0.319190	0.285495	-0.005030	0.005721	-0.005032	-0.277235
age_in_days	-0.060753	0.326508	0.319190	1.000000	0.833890	0.075775	0.062982	-0.042667	-0.893328
km	-0.006537	0.319580	0.285495	0.833890	1.000000	0.097539	0.035519	0.004839	-0.859373
previous_owners	0.007803	0.052480	-0.005030	0.075775	0.097539	1.000000	0.001697	-0.026836	-0.076274
lat	-0.058207	0.044901	0.005721	0.062982	0.035519	0.001697	1.000000	-0.766646	-0.011733
lon	0.058941	-0.013200	-0.005032	-0.042667	0.004839	-0.026836	-0.766646	1.000000	-0.003541
price	0.028516	-0.349885	-0.277235	-0.893328	-0.859373	-0.076274	-0.011733	-0.003541	1.000000

In [40]:
```
import seaborn as sns
sns.heatmap(cor,vmax=1,vmin=-1,annot=True,linewidths=.5,cmap='bwr')
```

Out[40]: <AxesSubplot:>



In [5]:
```
data1=data.drop(['model'],axis=1)
```

In [28]:
```
data1=data.sort_values('price')
data1
```

Out[28]:

	ID	model	engine_power	age_in_days	km	previous_owners	lat	lon	price
960	961	pop	51	4414	201000	1	45.141140	9.108080	2500
1330	1331	lounge	51	4627	171150	1	45.830688	12.033340	2900
268	269	lounge	51	4383	135762	1	43.879860	10.782810	3390
928	929	pop	51	4627	148000	1	45.356602	9.203450	3500
935	936	pop	73	4658	165000	2	45.584091	9.270960	3600
...	...	...	...	...	...	...	...	...	...
745	746	lounge	51	425	11416	1	41.903221	12.495650	11000
272	273	lounge	51	366	14816	2	38.122070	13.361120	11000
154	155	lounge	51	701	21580	1	41.125870	16.866659	11090
491	492	pop	51	670	9000	1	44.988739	9.010500	11090
675	676	lounge	51	487	17800	1	45.351528	10.844090	11100

1538 rows × 9 columns

In [6]:
```
cor=data1.corr()
cor
```

Out[6]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
ID	1.000000	-0.034059	-0.060753	-0.006537	0.007803	-0.058207	0.058941	0.028516
engine_power	-0.034059	1.000000	0.319190	0.285495	-0.005030	0.005721	-0.005032	-0.277235
age_in_days	-0.060753	0.319190	1.000000	0.833890	0.075775	0.062982	-0.042667	-0.893328
km	-0.006537	0.285495	0.833890	1.000000	0.097539	0.035519	0.004839	-0.859373
previous_owners	0.007803	-0.005030	0.075775	0.097539	1.000000	0.001697	-0.026836	-0.076274
lat	-0.058207	0.005721	0.062982	0.035519	0.001697	1.000000	-0.766646	-0.011733
lon	0.058941	-0.005032	-0.042667	0.004839	-0.026836	-0.766646	1.000000	-0.003541
price	0.028516	-0.277235	-0.893328	-0.859373	-0.076274	-0.011733	-0.003541	1.000000

In [7]:
```
data2=data.loc[(data.model=='lounge')&(data.previous_owners==1)]
```

In [12]:
```
#data['previous_owners'].unique()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-12-5097fa9a2cba> in <module>
      1 #data['previous_owners'].unique()
----> 2 data['price'].sort()

~/.local/lib/python3.8/site-packages/pandas/core/generic.py in __getattr__(self, name)
   5485         ):
   5486             return self[name]
-> 5487         return object.__getattribute__(self, name)
   5488 
   5489     def __setattr__(self, name: str, value) -> None:

AttributeError: 'Series' object has no attribute 'sort'
```

In [6]:
```
list(data.columns)
```

Out[6]:
```
['ID',
 'model',
 'engine_power',
 'age_in_days',
 'km',
 'previous_owners',
 'lat',
 'lon',
 'price']
```

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

Out[8]:

model	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
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 [9]:
```
data.groupby(['previous_owners']).count()
```

Out[9]:

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 [10]:
```
data['model'].unique()
```

Out[10]:
```
array(['lounge', 'pop', 'sport'], dtype=object)
```

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

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

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

In [14]:
```
data2
```

Out[14]:

	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 [16]:
```
#data2.groupby(['previous_owners'])
```

In [17]:
```
data2
```

Out[17]:

	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 [14]:
```
y=data2['price']
X=data2.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 [16]:
```
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 [17]:
```
X_test.head(5)
```

Out[17]:

	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 [18]:
```
X_train.shape
```

Out[18]:
```
(1030, 5)
```

In [19]:
```
y_train
```

Out[19]:
```
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 [20]:
```




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[20]:
```
LinearRegression()
```

In [21]:
```
#X_test=[[1,51,1000,28800,3],[1,51,780,18800,1]]
```

In [22]:
```
#above line to actual
```

In [23]:
```
ypred=reg.predict(X_test)
```

In [24]:
```
ypred
```

Out[24]:
```
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 [25]:
```
filename='pricemodel'
pickle.dump(reg,open(filename,'wb'))
```

In [ ]:
```
```

In [26]:
```
#savedmodel=pickle.load(open(filename,'rb'))

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

In [27]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[27]:
```
0.8383895235218546
```

In [ ]:

In [28]:
```
from sklearn.metrics import mean_squared_error #calculating MSE
mean_squared_error(ypred,y_test)
```

Out[28]: 593504.2888137395

In [29]:
```
#from sklearn.metrics import accuracy_score
#accuracy_score(y_test,ypred)
```

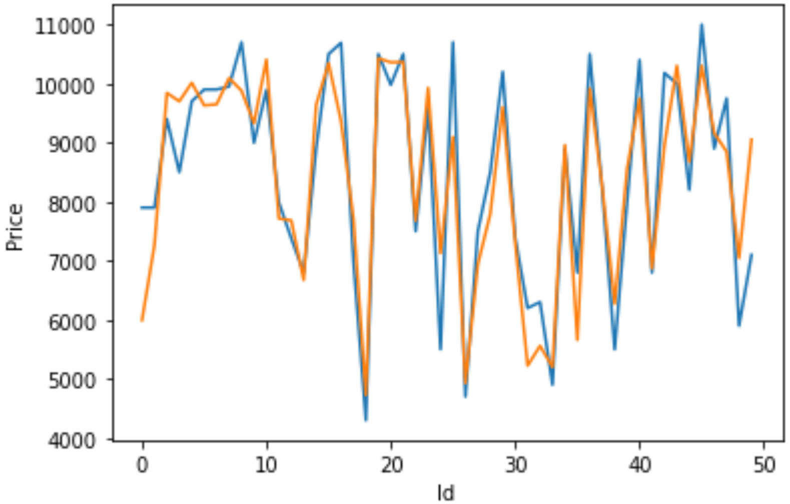
In [35]:
```
#Results= pd.DataFrame(columns=['Actual','Predicted'])
#Results['Actual']=y_test
Results= pd.DataFrame(columns=['Price','Predicted'])
Results['Price']=y_test
Results['Predicted']=ypred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[35]:

	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

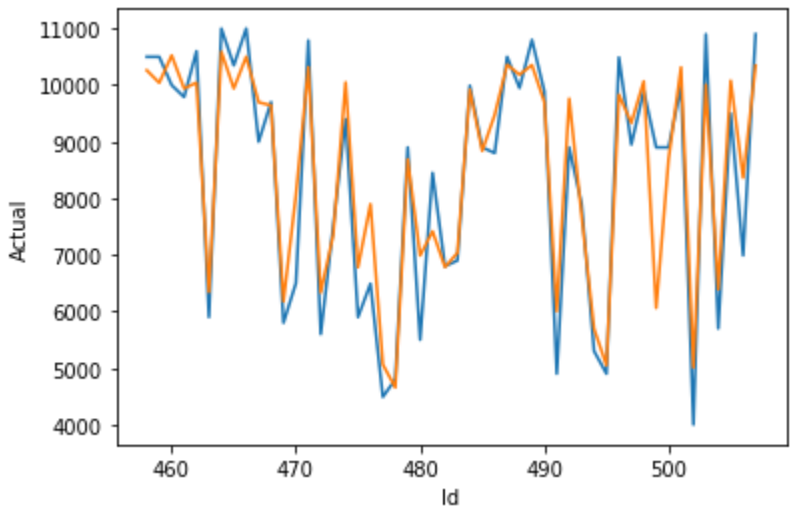
In [37]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='Price',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[37]: []



In [32]:
```
sns.lineplot(x='Id',y='Actual',data=Results.tail(50))
sns.lineplot(x='Id',y='Predicted',data=Results.tail(50))
plt.plot()
```

Out[32]: []



In [ ]:

In [ ]:

In [30]:
```
# ridge regression
```

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

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

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

In [33]:
```
#X_train=[2]
```

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

In [35]:
```
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

Out[35]:
```
590569.9121697355
```

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

Out[36]:
```
0.8391885506165899
```

In [37]:

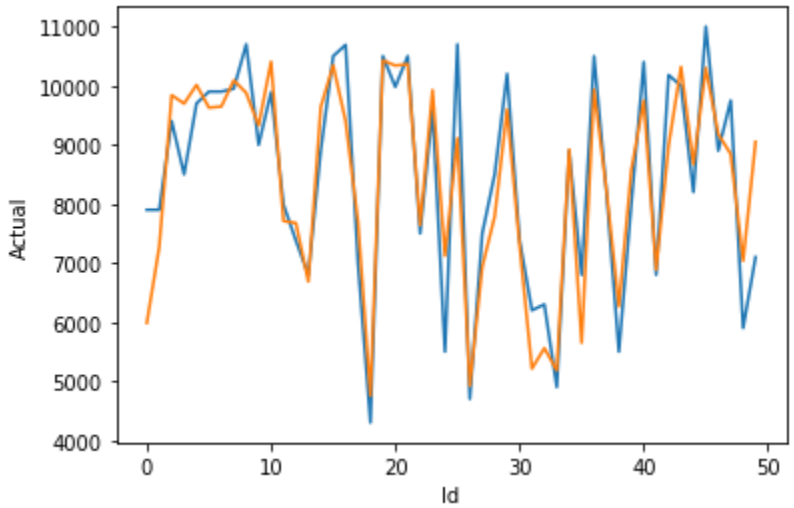
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred_ridge
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[37]:

	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

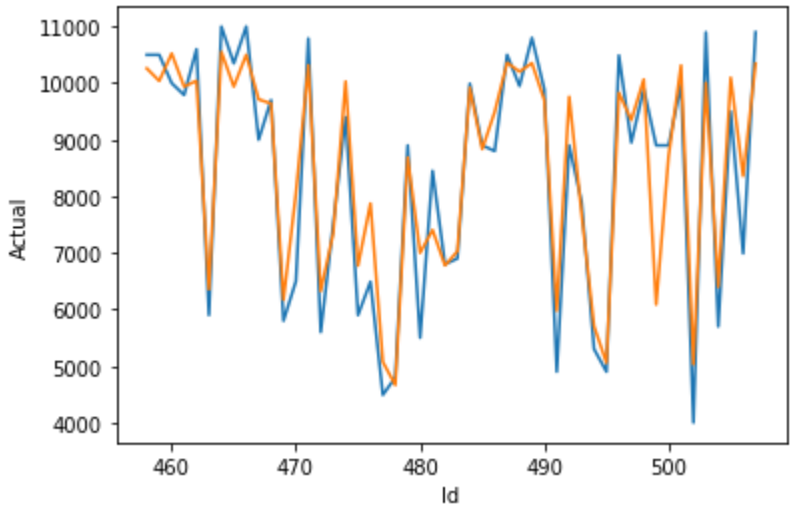
In [38]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[38]: []



In [39]:
```
sns.lineplot(x='Id',y='Actual',data=Results.tail(50))
sns.lineplot(x='Id',y='Predicted',data=Results.tail(50))
plt.plot()
```

Out[39]: []



In [40]:
```
#elastic
```

In [41]:
```
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)
```

```
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.379e+08, tolerance: 3.149e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.378e+08, tolerance: 3.129e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.443e+08, tolerance: 3.204e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.433e+08, tolerance: 3.065e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.493e+08, tolerance: 3.114e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.318e+08, tolerance: 3.149e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
of the features or consider increasing regularisation. Duality gap: 2.316e+08, tolerance: 3.129e+05
  model = cd_fast.enet_coordinate_descent(
/home/manu/.local/lib/python3.8/site-packages/sklearn/linear_model/_coordinate_descent.py:647: Convergence
Warning: Objective did not converge. You might want to increase the number of iterations, check the scale
```

```
of the features or consider increasing regularisation. Duality gap: 2.417e+08, tolerance: 3.114e+05
  model = cd_fast.enet_coordinate_descent(
```

Out[41]:
```
GridSearchCV(estimator=ElasticNet(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 1,
                                   5, 10, 20]})
```

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

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

In [43]:
```
elastic=ElasticNet(alpha=0.01)
elastic.fit(X_train,y_train)
y_pred_elastic=elastic.predict(X_test)
```

In [44]:
```
elastic_Error=mean_squared_error(y_pred_elastic,y_test)
elastic_Error
```

Out[44]:
```
592914.7556700263
```

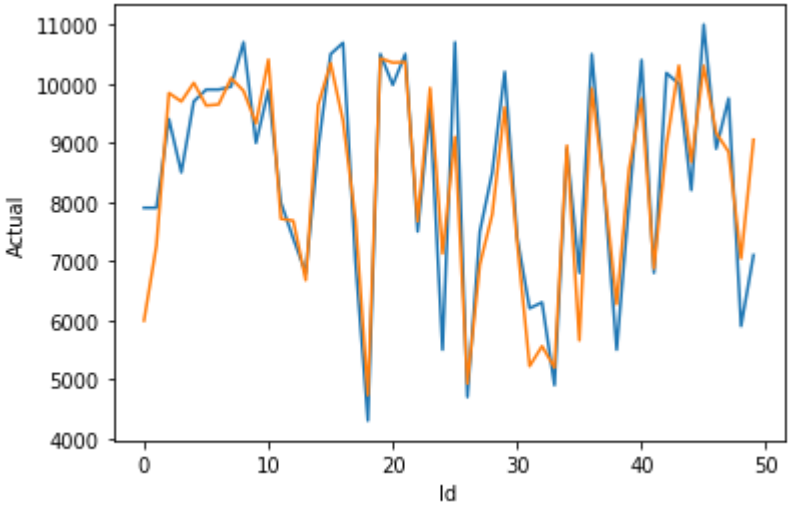
In [45]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred_elastic
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[45]:

	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 [46]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(50))
sns.lineplot(x='Id',y='Predicted',data=Results.head(50))
plt.plot()
```

Out[46]: []



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

```
/home/manu/.local/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:396: FutureWarning: Criterion 'm
se' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is eq
uivalent.
  warn(
/home/manu/.local/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:396: FutureWarning: Criterion 'm
se' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is eq
uivalent.
  warn(
/home/manu/.local/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:396: FutureWarning: Criterion 'm
se' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is eq
uivalent.
  warn(
/home/manu/.local/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:396: FutureWarning: Criterion 'm
se' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is eq
uivalent.
  warn(
/home/manu/.local/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:396: FutureWarning: Criterion 'm
se' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is eq
uivalent.
```

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

Out[48]:
```
{'criterion': 'mse', 'max_depth': 5, 'n_estimators': 50}
```

In [49]:
```
reg=RandomForestRegressor(n_estimators=200,criterion='mse',max_depth=5)
```

In [50]:
```
reg.fit(X_train,y_train)
```

```
/home/manu/.local/lib/python3.8/site-packages/sklearn/ensemble/_forest.py:396: FutureWarning: Criterion 'm
se' was deprecated in v1.0 and will be removed in version 1.2. Use `criterion='squared_error'` which is eq
uivalent.
  warn(
```

Out[50]:
```
RandomForestRegressor(criterion='mse', max_depth=5, n_estimators=200)
```

In [51]:
```
y_pred=reg.predict(X_test)
```

In [52]:
```
y_pred
```

Out[52]:
```
array([ 6031.71790936,  7116.79209995,  9786.74546584,  9744.44315157,
       10033.01250991,  9629.47656637,  9725.30622265, 10083.32622577,
        9827.68047728,  9330.60138407, 10367.59658403,  7776.14165353,
        7384.25358129,  7057.94718176,  9573.13965395, 10437.15931229,
        9678.2146621 ,  7890.07808152,  4921.57201206, 10371.87480021,
       10149.95253005, 10438.95859904,  7411.75186534,  9776.33154619,
        7363.96619937,  9150.34665672,  4884.67714928,  7090.87523082,
        7704.96174077,  9569.82401695,  7110.47574762,  5086.30321792,
        5488.87441171,  5012.2235297 ,  8641.03163094,  5482.95639145,
       10169.37188844,  7404.16352977,  6476.53303213,  8907.65724998,
        9769.02088768,  6801.17841555,  9327.81059815, 10327.15347559,
        8145.43902011, 10441.5870099 ,  9314.92162897,  8780.43960011,
        6440.29706047,  8844.26830125,  9511.33728003, 10381.25986813,
       10077.86068832,  7106.74880617,  9604.98221304,  9520.27757383,
        9633.45262024, 10433.98572582,  9759.20637029,  7044.17220871,
       10076.01099585,  7123.88648829,  9761.37201648,  7118.30123321,
        5816.32448799,  9705.80426611,  9776.2797593 ,  9031.53378045,
        8591.03060791,  6255.9528851 ,  7696.31655256,  7126.46326613,
        8672.93278744, 10393.24148915,  7102.24576057,  8474.71360635,
```

In [53]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

Out[53]:
```
0.8458159519560708
```

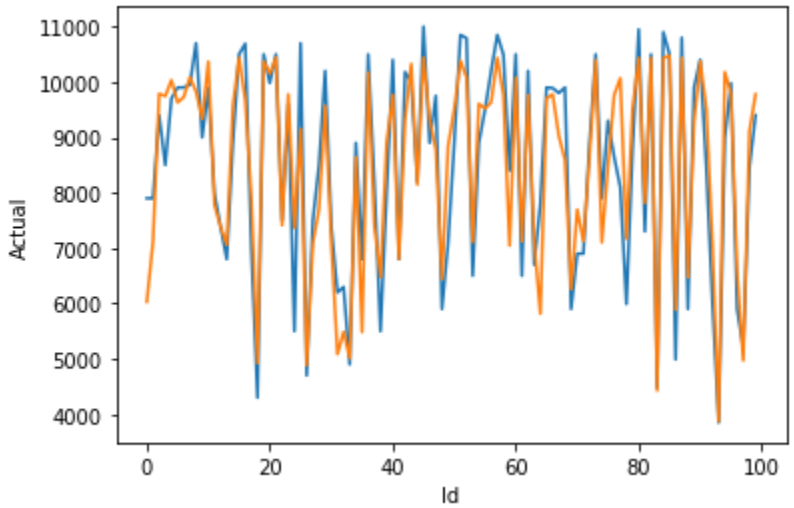
In [54]:
```
Results= pd.DataFrame(columns=['Actual','Predicted'])
Results['Actual']=y_test
Results['Predicted']=y_pred
#Results['km']=X_test['km']
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

Out[54]:

	index	Actual	Predicted	Id
0	481	7900	6031.717909	0
1	76	7900	7116.792100	1
2	1502	9400	9786.745466	2
3	669	8500	9744.443152	3
4	1409	9700	10033.012510	4
5	1414	9900	9629.476566	5
6	1089	9900	9725.306223	6
7	1507	9950	10083.326226	7
8	970	10700	9827.680477	8
9	1198	8999	9330.601384	9

In [55]:
```
sns.lineplot(x='Id',y='Actual',data=Results.head(100))
sns.lineplot(x='Id',y='Predicted',data=Results.head(100))
plt.plot()
```

Out[55]: []



In [ ]:

In [ ]: