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

In [2]:
```
data=pd.read_csv("/home/palacement/Downloads/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]:
```
data1=data.drop(['ID','lat','lon'],axis=1)
```

In [5]:
```
data
```

Out[5]:

	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

1538 rows × 9 columns

In [6]:
```
data2=pd.get_dummies(data)
```

In [7]:
```
data1.shape
```

Out[7]: (1538, 6)

In [8]:
```
data2=pd.get_dummies(data2)
```

In [9]:
```
data2.shape
```

Out[9]: (1538, 11)

In [10]:
```
y=data2['price']
```

In [11]:
```
x=data2.drop('price',axis=1)
```

In [12]:
```
y
```

Out[12]:
```
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 [13]:
```
x
```

Out[13]:

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

1538 rows × 10 columns

In [14]:
```
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 [15]:
```
x_test.head(5)
```

Out[15]:

	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 [16]:
```
x_train.shape
```
Out[16]: (1030, 10)

In [17]:
```
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 [18]:
```
x_test.head(5)
```
Out[18]:

	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 [19]:
```
dat1=data.drop('lat',axis=1)
```

In [20]:
```
dat1.shape
```
Out[20]: (1538, 8)

In [21]:
```
y_train
```

Out[21]:
```
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 [22]:
```
from sklearn.linear_model import LinearRegression
reg=LinearRegression ()
reg.fit (x_train,y_train)
```

Out[22]:
```
▾ LinearRegression
LinearRegression()
```

In [ ]:

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

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

```
        10371.21600688,  5343.40955708,  9793.46202119, 10248.73590685,
        10350.88894338,  9418.6649538 ,  9246.75432375,  9726.77158038,
         5646.60360194,  4954.59993355,  4854.00609399,  9667.14070802,
         6106.03185061,  9895.4585107 , 10067.23023087,  4939.52480184,
         8024.89878537,  9702.49506011,  5897.90997934, 10144.6495611 ,
         5395.93461448,  9622.44225965, 10171.86736173, 10103.58498957,
         9481.19877071,  4918.69676305,  5809.10532945,  7076.07274648,
        10066.02424638, 10430.97776811, 10050.79995384,  7801.53792597,
         8738.32379912,  9963.07184541, 10250.69391036,  9856.67153089,
         8383.84152492,  9307.84587539,  8530.90168144,  9859.23075392,
         9733.54483496,  9744.86150125,  6741.410463  ,  7342.18893371,
         8772.20704958,  9959.77345301,  9692.26944677, 10524.54487623,
         8221.41396472,  6722.97284178,  9894.93188478,  8849.71168914,
         9786.53980838, 10262.59139607, 10382.67498044,  9988.41681508,
         9336.80741819,  9902.52039123,  9109.63147621, 10147.01866123,
         7831.00036415,  6059.56493387,  8827.96184211, 10302.33416028,
         5660.1705204 , 10068.83508852,  9595.70115109,  7698.86996869,
         9319.54039166,  7421.93077111, 10397.65812756, 10008.49656229,
        10572.26845119,  9890.79746015,  9995.86970892,  6328.88724858,
```

In [25]:
```
from sklearn.metrics import r2_score
r2_score(y_test,ypred)
```

Out[25]:
```
0.8428319728488683
```

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

Out[26]:
```
577189.6736608233
```

In [27]:
```
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)
```

```
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=9.5143e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.38942e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.45639e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.93626e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09552e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.00948e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57945e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.22998e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.92606e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

```
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09075e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.01957e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57225e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.23226e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=6.92596e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.09069e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.01967e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.57214e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarning:
Ill-conditioned matrix (rcond=7.23225e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[27]:

```
▸ GridSearchCV
▸ estimator: Ridge
    ▸ Ridge
```

In [28]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
```

In [29]:

```
alpha = [1e-15, 1e-10, 1e-8, 1e-4, 1e-3,1e-2, 1, 5, 10, 20,30]
```

In [30]:
```
ridge = Ridge()
```

In [31]:
```
parameters = {'alpha': alpha}
```

In [32]:
```
ridge_regressor = GridSearchCV(ridge, parameters)
```

In [33]:
```
ridge_regressor.fit(x_train, y_train)
```

```
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=9.5143e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.38942e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=6.45639e-26): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=6.93626e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.09552e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.00948e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.57945e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.22998e-23): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=6.92606e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.09075e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.01957e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.57225e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.23226e-21): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

```
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=6.92596e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.09069e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.01967e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.57214e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
/home/palacement/anaconda3/lib/python3.10/site-packages/sklearn/linear_model/_ridge.py:216: LinAlgWarnin
g: Ill-conditioned matrix (rcond=7.23225e-17): result may not be accurate.
  return linalg.solve(A, Xy, assume_a="pos", overwrite_a=True).T
```

Out[33]:

```
▸ GridSearchCV
▸ estimator: Ridge
    ▸ Ridge
```

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]:

```
574728.5696156605
```

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

Out[37]: 0.8435021284061197

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)
```

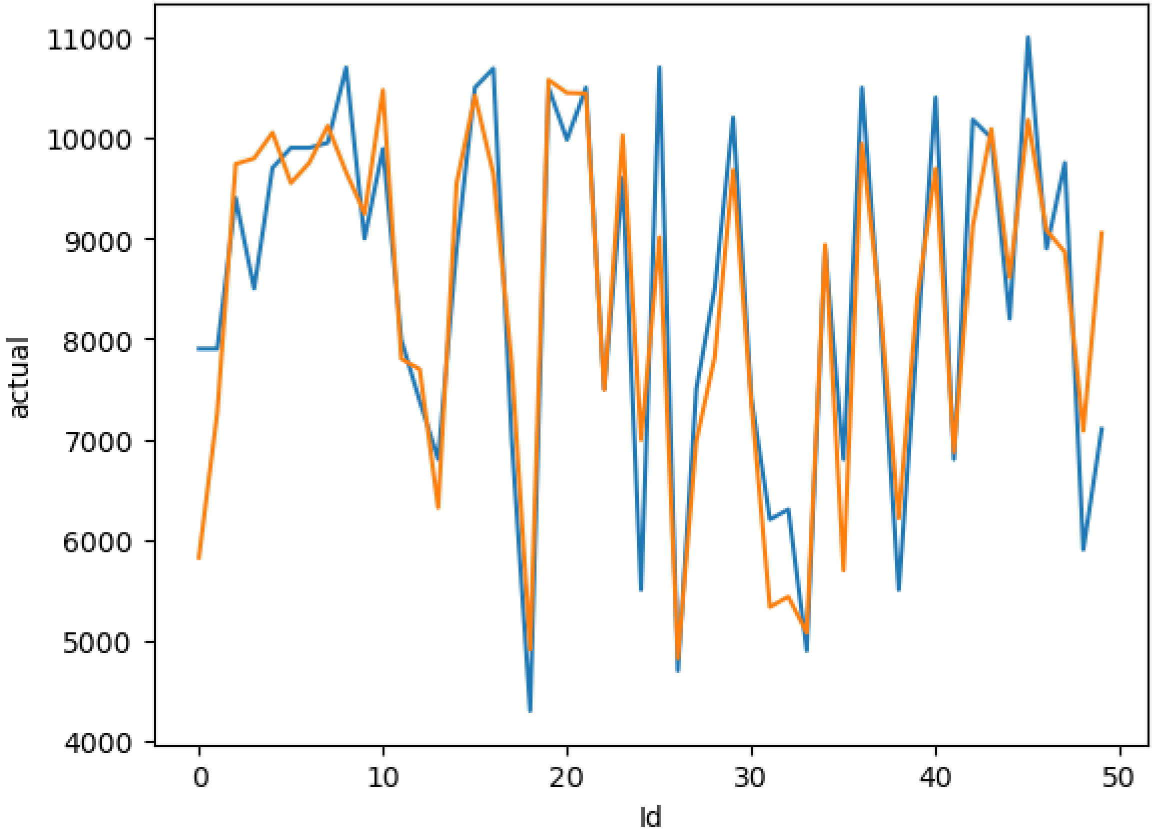
Out[38]:

	index	actual	predicted	Id
0	481	7900	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 [39]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [40]:
```
sns.lineplot(x='Id',y='actual',data=Results.head(50))
sns.lineplot(x='Id',y='predicted',data=Results.head(50))
plt.plot()
```

Out[40]: []



In [41]:
```
import pandas as a
import pickle
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import GridSearchCV
```

In [42]:
```
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[42]:
```
▸ GridSearchCV
▸ estimator: ElasticNet
    ▸ ElasticNet
```

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

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

In [44]:
```
elastic=ElasticNet(alpha=0.1)
elastic.fit(x_train,y_train)
y_pred_elastic=elastic.predict(x_test)
```

In [45]:
```
from sklearn.metrics import mean_squared_error
Elastic_Error=mean_squared_error(y_pred_elastic,y_test)
Elastic_Error
```

Out[45]:
```
573512.0849841419
```

In [46]:
```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred_elastic)
```

Out[46]: 0.843833375651731

In [48]:
```
Results=pd.DataFrame(columns=['actual','predicted'])
Results['actual']=y_test
Results['predicted']=y_pred_ridge
import seaborn as sns
import matplotlib.pyplot as pltd_elastic
Results=Results.reset_index()
Results['Id']=Results.index
Results.head(10)
```

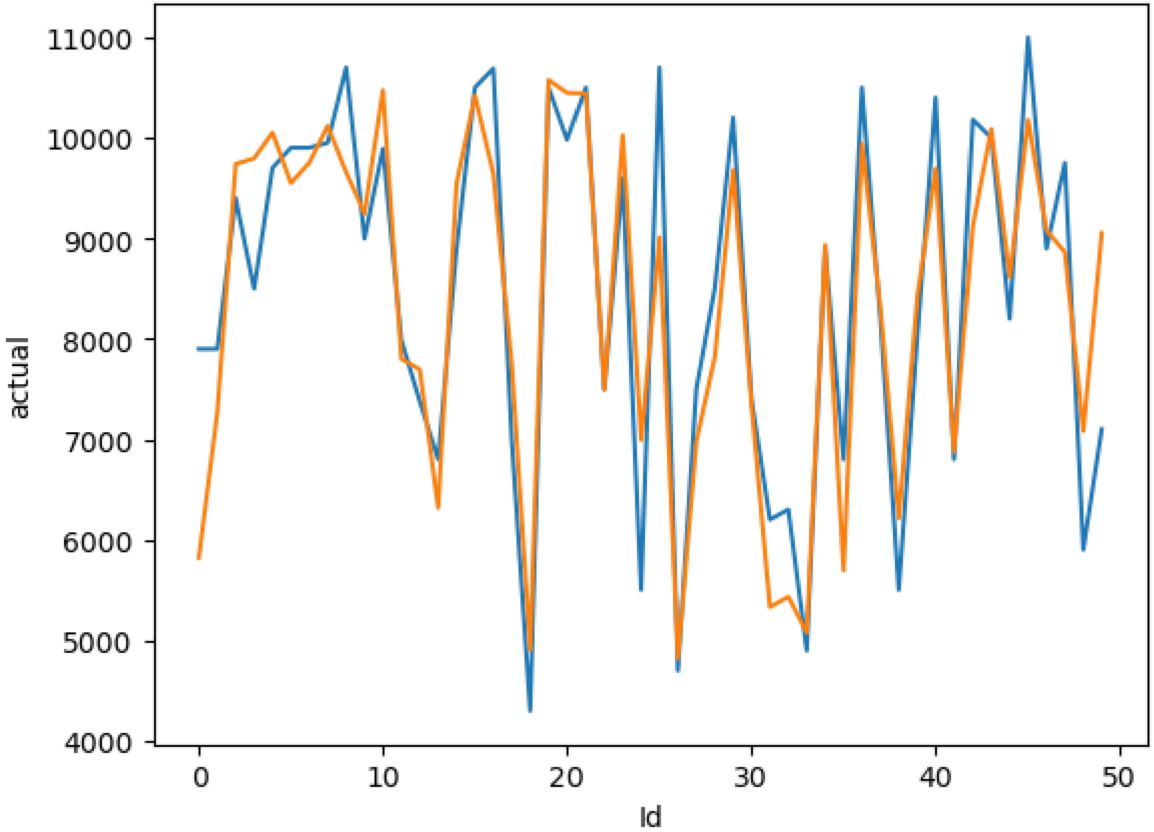
Out[48]:

	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 [49]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

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

Out[50]: []



In [ ]: