

```
In [58]: import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

```
In [59]: data= pd.read_csv('/home/placement/Desktop/fiat500.csv')
data
```

Out[59]:

	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 [60]: `data.head(10)`

Out[60]:

	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 [61]: `data.columns`

Out[61]: Index(['ID', 'model', 'engine\_power', 'age\_in\_days', 'km', 'previous\_owners',  
                  'lat', 'lon', 'price'],  
              dtype='object')

In [62]: `data.shape`

Out[62]: (1538, 9)

In [63]: `data.describe()`

Out[63]:

	ID	engine_power	age_in_days	km	previous_owners	lat	lon	price
<b>count</b>	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000	1538.000000
<b>mean</b>	769.500000	51.904421	1650.980494	53396.011704	1.123537	43.541361	11.563428	8576.003901
<b>std</b>	444.126671	3.988023	1289.522278	40046.830723	0.416423	2.133518	2.328190	1939.958641
<b>min</b>	1.000000	51.000000	366.000000	1232.000000	1.000000	36.855839	7.245400	2500.000000
<b>25%</b>	385.250000	51.000000	670.000000	20006.250000	1.000000	41.802990	9.505090	7122.500000
<b>50%</b>	769.500000	51.000000	1035.000000	39031.000000	1.000000	44.394096	11.869260	9000.000000
<b>75%</b>	1153.750000	51.000000	2616.000000	79667.750000	1.000000	45.467960	12.769040	10000.000000
<b>max</b>	1538.000000	77.000000	4658.000000	235000.000000	4.000000	46.795612	18.365520	11100.000000

## Removing unwanted columns

In [64]: `data1=data.drop(columns=["ID","lat","lon"])`

In [65]: data1

Out[65]:

	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 [66]: data1=pd.get\_dummies(data1)

In [67]: data1

Out[67]:

	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 [68]: data1.shape

Out[68]: (1538, 8)

## remove the actual value from the dataframe

```
In [69]: y=data1['price']
          x=data1.drop(columns='price')
```

In [70]:

x

Out[70]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
0	51	882	25000	1	1	0	0
1	51	1186	32500	1	0	1	0
2	74	4658	142228	1	0	0	1
3	51	2739	160000	1	1	0	0
4	73	3074	106880	1	0	1	0
...	...	...	...	...	...	...	...
1533	51	3712	115280	1	0	0	1
1534	74	3835	112000	1	1	0	0
1535	51	2223	60457	1	0	1	0
1536	51	2557	80750	1	1	0	0
1537	51	1766	54276	1	0	1	0

1538 rows × 7 columns

In [71]:

y

Out[71]:

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

## split the data into training set and testing set

```
In [72]: 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)# 66% and 33%
```

```
In [73]: x_test
```

Out[73]:

	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
...	...	...	...	...	...	...	...
291	51	701	22000	1	1	0	0
596	51	3347	85500	1	0	1	0
1489	51	366	22148	1	0	1	0
1436	51	1797	61000	1	1	0	0
575	51	366	19112	1	1	0	0

508 rows × 7 columns

In [74]: x\_train

Out[74]:

	engine_power	age_in_days	km	previous_owners	model_lounge	model_pop	model_sport
527	51	425	13111	1	1	0	0
129	51	1127	21400	1	1	0	0
602	51	2039	57039	1	0	1	0
331	51	1155	40700	1	1	0	0
323	51	425	16783	1	1	0	0
...	...	...	...	...	...	...	...
1130	51	1127	24000	1	1	0	0
1294	51	852	30000	1	1	0	0
860	51	3409	118000	1	0	1	0
1459	51	762	16700	1	1	0	0
1126	51	701	39207	1	1	0	0

1030 rows × 7 columns

In [75]: y\_test.head(5)

Out[75]: 481 7900  
76 7900  
1502 9400  
669 8500  
1409 9700  
Name: price, dtype: int64



```
In [76]: y_train.head(5)
```

```
Out[76]: 527    9990
         129    9500
         602    7590
         331    8750
         323    9100
         Name: price, dtype: int64
```

## Linear Regression

```
In [77]: from sklearn.linear_model import LinearRegression
         reg=LinearRegression()#creating object of LinearRegression
         reg.fit(x_train,y_train)#training and fitting
```

```
Out[77]: ▼ LinearRegression
         LinearRegression()
```

```
In [78]: y_pred_regression=reg.predict(x_test) #predict the price using x_test data
```

In [79]: `y_pred_regression`

Out[79]: `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,`  
`8820.78555100, 10025.82571520, 7270.77100000, 8411.45004000`

In [80]: `from sklearn.metrics import r2_score #to know the efficiency bw the predicted price`  
`r2_score(y_test,y_pred_regression)`

Out[80]: `0.8415526986865394`

In [81]: `from sklearn.metrics import mean_squared_error#calaculating mse`  
`mean_squared_error(y_test,y_pred_regression)`

Out[81]: `581887.727391353`

In [82]: `import math`  
`a=581887.727391353`  
`print(math.sqrt(a))`

`762.8156575420782`

```
In [83]: y_test.head(10)
```

```
Out[83]: 481      7900
         76      7900
         1502    9400
         669    8500
         1409    9700
         1414    9900
         1089    9900
         1507    9950
         970   10700
         1198    8999
         Name: price, dtype: int64
```

```
In [84]: results=pd.DataFrame(columns=['Price','Predicted']) #create datafame for price and predicted
         results['Price']=y_test
         results['Predicted']=y_pred_regression
         results=results.reset_index() #remove the index as ID values
         results['id']=results.index
```

In [85]: results

Out[85]:

	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
...	...	...	...	...
503	291	10900	10032.665135	503
504	596	5699	6281.536277	504
505	1489	9500	9986.327508	505
506	1436	6990	8381.517020	506
507	575	10900	10371.142553	507

508 rows × 4 columns

In [86]: results["Difference"]=results.apply(lambda x:x.Price-x.Predicted,axis=1)#add the column for difference b/w

In [87]: `results.head(10)`

Out[87]:

	index	Price	Predicted	id	Difference
0	481	7900	5867.650338	0	2032.349662
1	76	7900	7133.701423	1	766.298577
2	1502	9400	9866.357762	2	-466.357762
3	669	8500	9723.288745	3	-1223.288745
4	1409	9700	10039.591012	4	-339.591012
5	1414	9900	9654.075826	5	245.924174
6	1089	9900	9673.145630	6	226.854370
7	1507	9950	10118.707281	7	-168.707281
8	970	10700	9903.859527	8	796.140473
9	1198	8999	9351.558284	9	-352.558284

In [88]: results

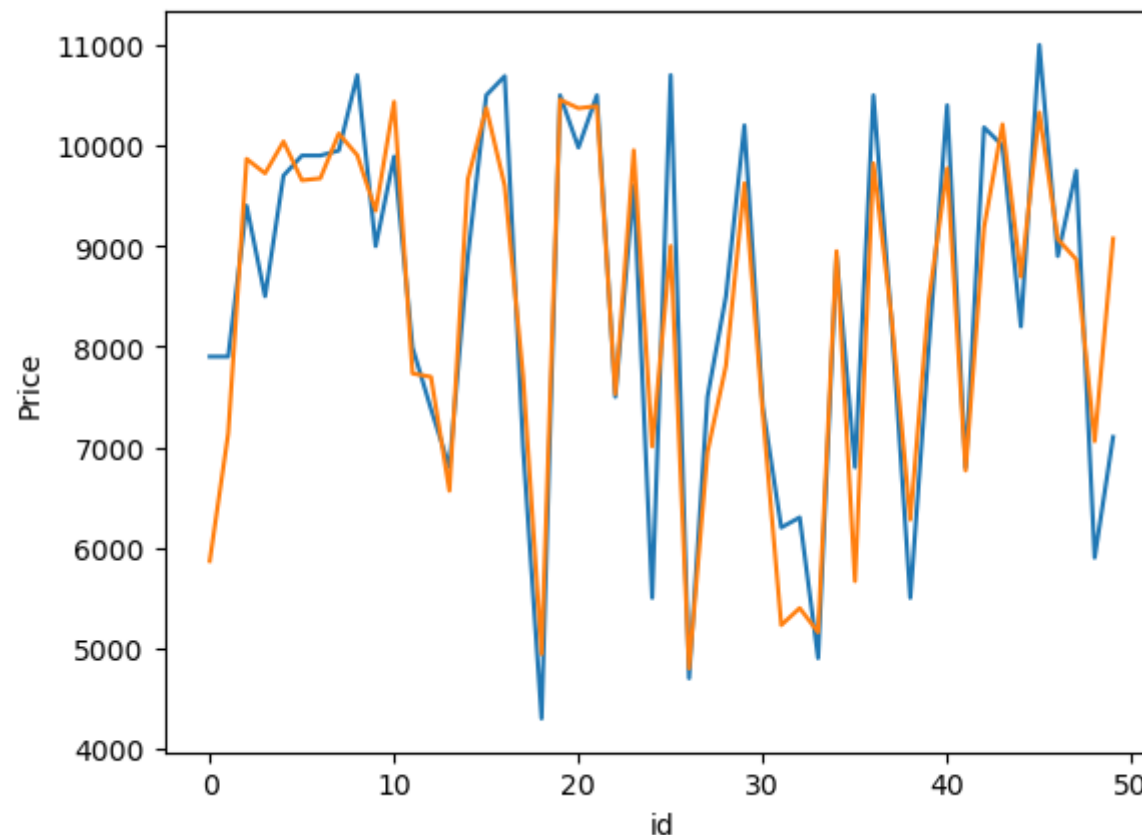
Out[88]:

	index	Price	Predicted	id	Difference
0	481	7900	5867.650338	0	2032.349662
1	76	7900	7133.701423	1	766.298577
2	1502	9400	9866.357762	2	-466.357762
3	669	8500	9723.288745	3	-1223.288745
4	1409	9700	10039.591012	4	-339.591012
...	...	...	...	...	...
503	291	10900	10032.665135	503	867.334865
504	596	5699	6281.536277	504	-582.536277
505	1489	9500	9986.327508	505	-486.327508
506	1436	6990	8381.517020	506	-1391.517020
507	575	10900	10371.142553	507	528.857447

508 rows × 5 columns

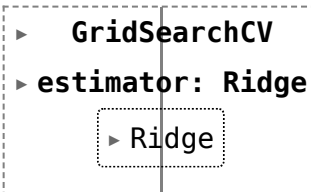
## plot the data using seaborn and matplotlib libraries

```
In [89]: import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='id',y='Price',data=results.head(50)) #actual color=blue
sns.lineplot(x='id',y='Predicted',data=results.head(50)) #predicted color=orange
plt.show()
```



## Ridge\_Regression

```
In [90]: from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Ridge
alpha=[1e-15,1e-10,1e-4,1e-3,1e-2,15,10,20,30]
ridge=Ridge() #creating an object for Ridge
parameters={'alpha':alpha}
ridge_regressor=GridSearchCV(ridge,parameters)
ridge_regressor.fit(x_train,y_train) #training and fitting
```

```
Out[90]: 
```

```
In [91]: ridge_regressor.best_params_
```

```
Out[91]: {'alpha': 30}
```

```
In [92]: ridge=Ridge(alpha=30)
ridge.fit(x_train,y_train)
y_pred_ridge=ridge.predict(x_test)
```

```
In [93]: from sklearn.metrics import r2_score #to know the efficiency of the predicted price
r2_score(y_test,y_pred_ridge)
```

```
Out[93]: 0.8421969385523054
```

```
In [94]: from sklearn.metrics import mean_squared_error
Ridge_Error=mean_squared_error(y_pred_ridge,y_test)
Ridge_Error
```

```
Out[94]: 579521.7970897449
```



```
In [95]: results=pd.DataFrame(columns=['Actual','Predicted']) #create the dataframe for actual and predicted values
results['Actual']=y_test
results['Predicted']=y_pred_ridge
results=results.reset_index() #remove the index as ID values
results['id']=results.index
```

```
In [96]: results
```

```
Out[96]:
```

	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
...	...	...	...	...
503	291	10900	10029.070743	503
504	596	5699	6297.833772	504
505	1489	9500	10008.285472	505
506	1436	6990	8375.789449	506
507	575	10900	10368.170257	507

508 rows × 4 columns

```
In [97]: results["Difference"]=results.apply(lambda x:x.Actual-x.Predicted,axis=1)#add the column for difference b/
```

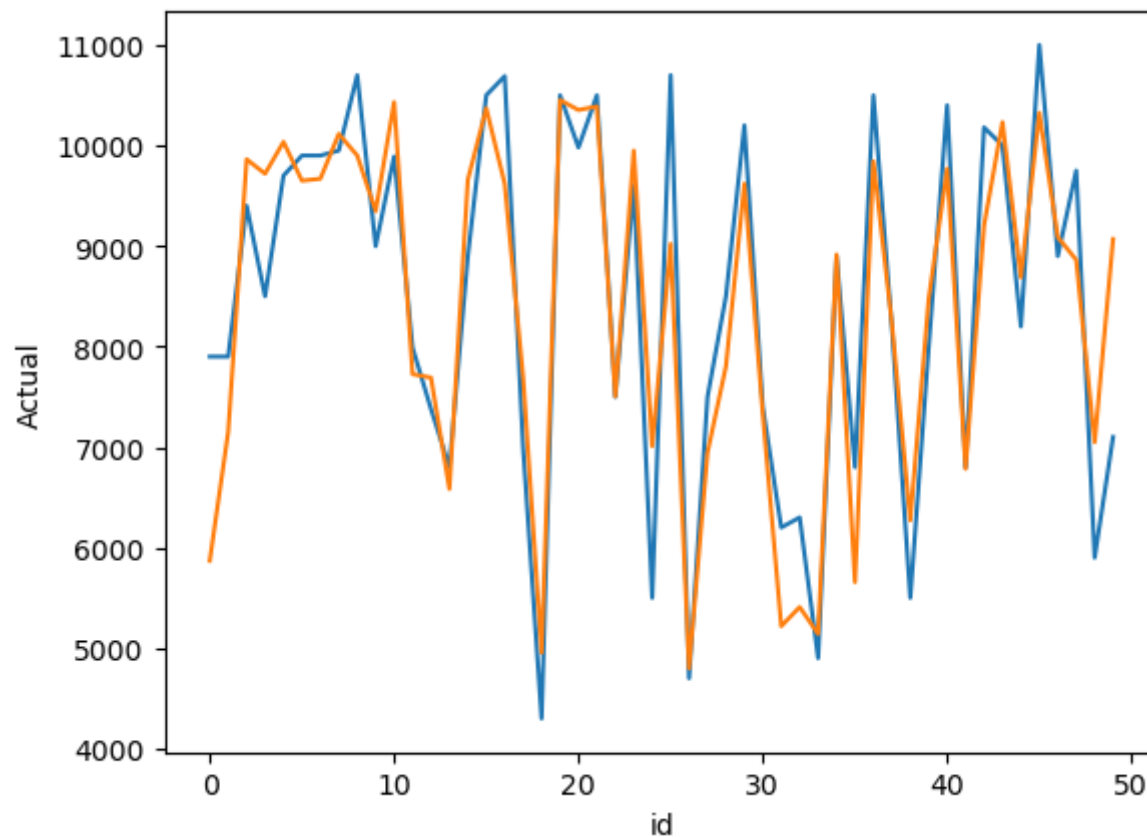
In [98]: results

Out[98]:

	index	Actual	Predicted	id	Difference
0	481	7900	5869.741155	0	2030.258845
1	76	7900	7149.563327	1	750.436673
2	1502	9400	9862.785355	2	-462.785355
3	669	8500	9719.283532	3	-1219.283532
4	1409	9700	10035.895686	4	-335.895686
...	...	...	...	...	...
503	291	10900	10029.070743	503	870.929257
504	596	5699	6297.833772	504	-598.833772
505	1489	9500	10008.285472	505	-508.285472
506	1436	6990	8375.789449	506	-1385.789449
507	575	10900	10368.170257	507	531.829743

508 rows × 5 columns

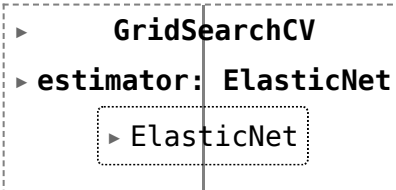
```
In [99]: import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='id',y='Actual',data=results.head(50))
sns.lineplot(x='id',y='Predicted',data=results.head(50))
plt.show()
```



## ElastiNet model

```
In [100]: from sklearn.linear_model import ElasticNet
from sklearn.model_selection import GridSearchCV
elastic=ElasticNet() #creating an object for ElasticNet
parameters={'alpha':[1e-15,1e-10,1e-8,1e-4,1e-3,1e-2,15,10,20]}
elastic_regressor=GridSearchCV(elastic,parameters)
elastic_regressor.fit(x_train,y_train) #training and fitting
```

```
Out[100]:
```



```
  ▶ GridSearchCV
  ▶ estimator: ElasticNet
    ▶ ElasticNet
```

```
In [101]: elastic_regressor.best_params_
```

```
Out[101]: {'alpha': 0.01}
```

```
In [102]: elastic=ElasticNet(alpha=0.01)
elastic.fit(x_train,y_train)
y_pred_enet=elastic.predict(x_test)
```

```
In [103]: from sklearn.metrics import mean_squared_error
elastic_Error=mean_squared_error(y_pred_enet,y_test)
elastic_Error
```

```
Out[103]: 581390.7642825295
```

```
In [104]: from sklearn.metrics import r2_score #to know the efficiency of the predicted price
r2_score(y_test,y_pred_enet)
```

```
Out[104]: 0.841688021120299
```

```
In [105]: results=pd.DataFrame(columns=['Actual','Predicted']) #create the dataframe for actual and predicted values
results['Actual']=y_test
results['Predicted']=y_pred_enet
results=results.reset_index() #remove the index as ID values
results['id']=results.index
```

In [106]: results

Out[106]:

	index	Actual	Predicted	id
<b>0</b>	481	7900	5867.742075	0
<b>1</b>	76	7900	7136.527402	1
<b>2</b>	1502	9400	9865.726723	2
<b>3</b>	669	8500	9722.573593	3
<b>4</b>	1409	9700	10038.936496	4
...	...	...	...	...
<b>503</b>	291	10900	10032.030157	503
<b>504</b>	596	5699	6284.484674	504
<b>505</b>	1489	9500	9990.379510	505
<b>506</b>	1436	6990	8380.465651	506
<b>507</b>	575	10900	10370.628731	507

508 rows × 4 columns

In [107]: results["Difference"]=results['Actual']-results['Predicted']*#add the column for difference b/w the actual*

In [108]: results

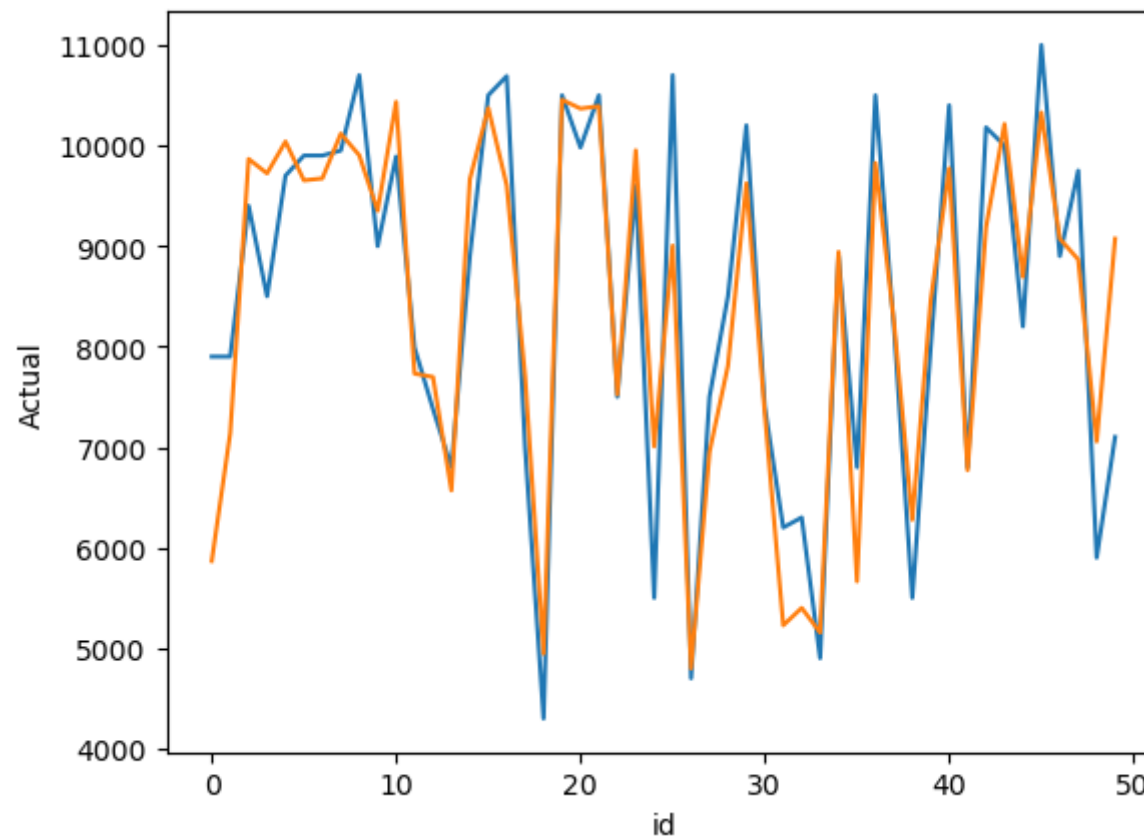
Out[108]:

	index	Actual	Predicted	id	Difference
0	481	7900	5867.742075	0	2032.257925
1	76	7900	7136.527402	1	763.472598
2	1502	9400	9865.726723	2	-465.726723
3	669	8500	9722.573593	3	-1222.573593
4	1409	9700	10038.936496	4	-338.936496
...	...	...	...	...	...
503	291	10900	10032.030157	503	867.969843
504	596	5699	6284.484674	504	-585.484674
505	1489	9500	9990.379510	505	-490.379510
506	1436	6990	8380.465651	506	-1390.465651
507	575	10900	10370.628731	507	529.371269

508 rows × 5 columns

## Plot the data using seaborn and matplotlib libraries

```
In [109]: import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='id',y='Actual',data=results.head(50))
sns.lineplot(x='id',y='Predicted',data=results.head(50))
plt.show()
```



```
In [ ]:
```