In [35]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings("ignore")
```

In [36]:

```
data=pd.read_csv("/home/placement/Desktop/prasanna/Advertising.csv")
```

In [37]:

```
data.describe()
```

Out[37]:

	Unnamed: 0	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

In [38]:

```
data.head()
```

Out[38]:

	Unnamed: 0	TV	radio	newspaper	sales
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

In [39]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   Unnamed: 0  200 non-null    int64  
 1   TV          200 non-null    float64
 2   radio       200 non-null    float64
 3   newspaper   200 non-null    float64
 4   sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

In [40]:
```
list(data)
```

Out[40]:
```
['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales']
```

In [41]:
```
data1=data.drop(['Unnamed: 0'],axis=1)
```

In [42]:
```
data1
```

Out[42]:

	TV	radio	newspaper	sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	9.3
3	151.5	41.3	58.5	18.5
4	180.8	10.8	58.4	12.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	9.7
197	177.0	9.3	6.4	12.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	13.4

200 rows × 4 columns

In [43]:
```
list(data1)
```

Out[43]:
```
['TV', 'radio', 'newspaper', 'sales']
```

In [44]:
```
y=data1['sales']#predicted value removed from data frame
x=data1.drop(['sales'],axis=1)
```

```
In [45]: y
```

```
Out[45]: 0      22.1
         1      10.4
         2       9.3
         3      18.5
         4      12.9
                ...
         195     7.6
         196     9.7
         197    12.8
         198    25.5
         199    13.4
         Name: sales, Length: 200, dtype: float64
```

```
In [46]: x
```

Out[46]:

	TV	radio	newspaper
0	230.1	37.8	69.2
1	44.5	39.3	45.1
2	17.2	45.9	69.3
3	151.5	41.3	58.5
4	180.8	10.8	58.4
...	...	...	...
195	38.2	3.7	13.8
196	94.2	4.9	8.1
197	177.0	9.3	6.4
198	283.6	42.0	66.2
199	232.1	8.6	8.7

200 rows × 3 columns

In [47]:

```
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 [48]:

```
x_test.head(10)
```

Out[48]:

	TV	radio	newspaper
95	163.3	31.6	52.9
15	195.4	47.7	52.9
30	292.9	28.3	43.2
158	11.7	36.9	45.2
128	220.3	49.0	3.2
115	75.1	35.0	52.7
69	216.8	43.9	27.2
170	50.0	11.6	18.4
174	222.4	3.4	13.1
45	175.1	22.5	31.5

In [49]:

```
y_test.head(10)
```

Out[49]:

```
95     16.9
15     22.4
30     21.4
158     7.3
128    24.7
115    12.6
69     22.3
170     8.4
174    11.5
45     14.9
Name: sales, dtype: float64
```

In [50]:

```
x_train.head(5)
```

Out[50]:

	TV	radio	newspaper
42	293.6	27.7	1.8
189	18.7	12.1	23.4
90	134.3	4.9	9.3
136	25.6	39.0	9.3
51	100.4	9.6	3.6

In [51]:

```
y_train.head(5)
```

Out[51]:

```
42     20.7
189     6.7
90     11.2
136     9.5
51     10.7
Name: sales, dtype: float64
```

In [53]:

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import Lasso
lasso=Lasso()
parameters={'alpha':[1e-15,1e-10,1e-8,1e-4,1e-3,1e-2,1,5,10,20]}
lasso_regressor=GridSearchCV(lasso,parameters)
lasso_regressor.fit(x_train,y_train)
```

Out[53]:

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

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

```
In [54]: lasso_regressor.best_params_

Out[54]: {'alpha': 1}
```

```
In [55]: lasso=Lasso(alpha=1)
```

```
In [56]: lasso.fit(x_train,y_train)
         y_pred_lasso=lasso.predict(x_test)
```

```
In [57]: from sklearn.metrics import mean_squared_error
         Lasso_Error=mean_squared_error(y_pred_lasso,y_test)
         Lasso_Error

Out[57]: 3.641439660278575
```

```
In [58]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred_lasso)

Out[58]: 0.8589079527148957
```

In [60]:
```
results=pd.DataFrame(columns=['actual','Predicted'])
results['actual']=y_test
results['Predicted']=y_pred_lasso
results=results.reset_index()
results['Id']=results.index
results.head(10)
```

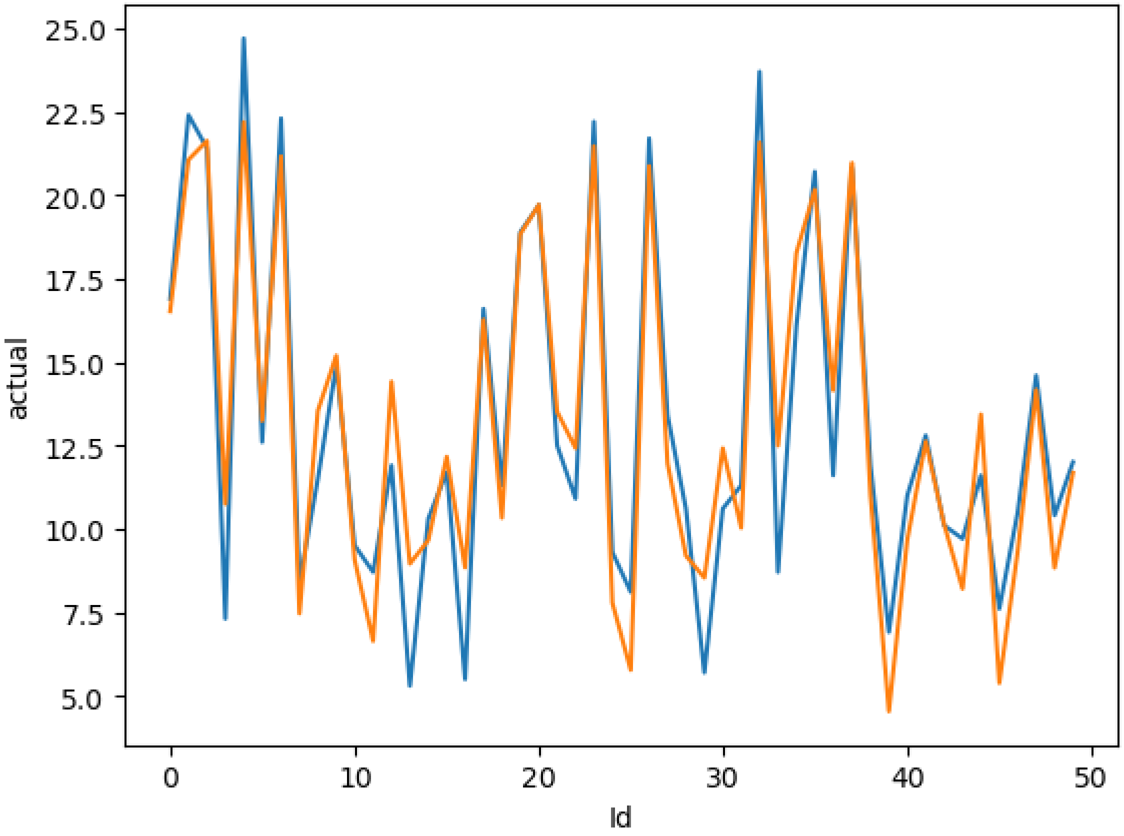
Out[60]:
```
   index  actual  Predicted  Id
0     95    16.9  16.523920   0
1     15    22.4  21.058219   1
2     30    21.4  21.624966   2
3    158     7.3  10.745724   3
4    128    24.7  22.188269   4
5    115    12.6  13.243102   5
6     69    22.3  21.161155   6
7    170     8.4   7.454875   7
8    174    11.5  13.541765   8
9     45    14.9  15.197360   9
```

In [61]:

```
import seaborn as sns
import matplotlib.pyplot as plt
sns.lineplot(x='Id',y='actual',data=results.head(50)) #red is actual
sns.lineplot(x='Id',y='Predicted',data=results.head(50)) #blue is predicted
```

Out[61]:

```
<Axes: xlabel='Id', ylabel='actual'>
```



In [ ]: