Import libraries

In [5]:
```
import pandas as pd
import warnings
warnings.filterwarnings('ignore')
import seaborn as sns
import matplotlib.pyplot as plt
```

Exploratory Data Analysis

In [6]:
```
data=pd.read_csv('/home/placement/Downloads/Advertising.csv')#read the titanic csv file
print(data)
```

```
     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
..          ...    ...    ...        ...    ...
195         196   38.2    3.7       13.8    7.6
196         197   94.2    4.9        8.1    9.7
197         198  177.0    9.3        6.4   12.8
198         199  283.6   42.0       66.2   25.5
199         200  232.1    8.6        8.7   13.4

[200 rows x 5 columns]
```

In [2]:
```
data.head()
```

Out[2]:

	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 [3]:
```
data.describe()
```

Out[3]:

	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 [8]:
```
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 [9]:
```
data.columns
```

Out[9]:
```
Index(['Unnamed: 0', 'TV', 'radio', 'newspaper', 'sales'], dtype='object')
```

In [10]:
```
data1=data.drop(columns='Unnamed: 0')
```

In [11]:
```
data1
```

Out[11]:

	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 [15]:
```
y=data1['sales'] #copy the sales
x=data1.drop(columns='sales')
```

In [16]:
```
x
```

Out[16]:

	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

splitting data into training and testing sets

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

Lasso

In [28]:
```
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-2,5,10,20,30]}
lasso_regressor=GridSearchCV(lasso,parameters)
lasso_regressor.fit(x_train,y_train)
```

Out[28]:
```
GridSearchCV(estimator=Lasso(),
             param_grid={'alpha': [1e-15, 1e-10, 1e-08, 0.0001, 0.001, 0.01, 5,
                                   10, 20, 30]})
```

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 [29]:
```
lasso_regressor.best_params_
```

Out[29]:
```
{'alpha': 0.01}
```

In [32]:
```
lasso=Lasso(alpha=0.01)
lasso.fit(x_train,y_train)
y_pred_lasso=lasso.predict(x_test)
```

In [33]:
```
from sklearn.metrics import r2_score #to know the efficiency of the predicted price
r2_score(y_test,y_pred_lasso)
```

Out[33]:
```
0.8555927456329158
```

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

Out[26]:
```
3.727001722653106
```

In [34]:
```
results=pd.DataFrame(columns=['Actual','Predicted'])   #create the dataframe for actual and predicted values
results['Actual']=y_test
results['Predicted']=y_pred_lasso
results=results.reset_index()  #remove the index as ID values
results['id']=results.index
```

In [35]:
```
results
```

Out[35]:

	index	Actual	Predicted	id
0	95	16.9	16.586103	0
1	15	22.4	21.184946	1
2	30	21.4	21.667103	2
3	158	7.3	10.810215	3
4	128	24.7	22.251471	4
...	...	...	...	...
61	97	15.5	15.279738	61
62	31	11.9	11.456759	62
63	12	9.2	11.122240	63
64	35	12.8	16.601060	64
65	119	6.6	6.906611	65

66 rows × 4 columns

In [36]:
```
results["Difference"]=results.apply(lambda x:x.Actual-x.Predicted,axis=1)#add the column for difference b/w
```

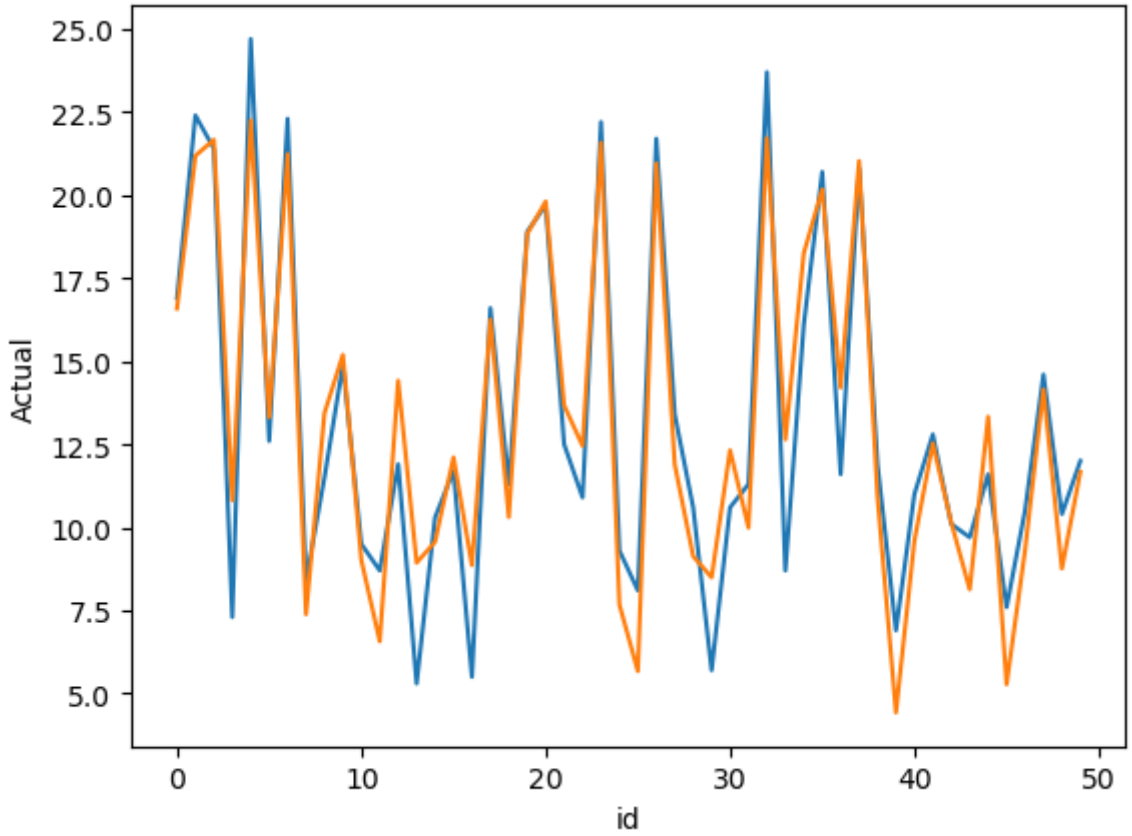
In [37]:
```
results
```

Out[37]:

	index	Actual	Predicted	id	Difference
0	95	16.9	16.586103	0	0.313897
1	15	22.4	21.184946	1	1.215054
2	30	21.4	21.667103	2	-0.267103
3	158	7.3	10.810215	3	-3.510215
4	128	24.7	22.251471	4	2.448529
...	...	...	...	...	...
61	97	15.5	15.279738	61	0.220262
62	31	11.9	11.456759	62	0.443241
63	12	9.2	11.122240	63	-1.922240
64	35	12.8	16.601060	64	-3.801060
65	119	6.6	6.906611	65	-0.306611

66 rows × 5 columns

In [38]:
```
sns.lineplot(x='id',y='Actual',data=results.head(50))    #plot the data
sns.lineplot(x='id',y='Predicted',data=results.head(50))
plt.show()
```



In [ ]: