Problem Statement

Title: Predicting Sales Based on Advertising Spend

Problem Description:

In today's competitive market, companies are constantly looking for ways to optimize their advertising budgets to maximize sales. Understanding the relationship between advertising expenditure across various media channels and sales performance is crucial for making informed marketing decisions. This dataset contains data on advertising expenditures across three channels: TV, Radio, and Newspaper, along with the resulting sales figures

Objective:

The objective of this analysis is to build a predictive model that can accurately forecast sales based on the amount of money spent on TV, Radio, and Newspaper advertisements. This model will help in understanding the impact of each advertising channel on sales and in making strategic decisions regarding budget allocation to maximize sales

In [4]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

In [5]:
```
code=pd.read_csv('advertising.csv')
code
```

Out[5]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...
195	38.2	3.7	13.8	7.6
196	94.2	4.9	8.1	14.0
197	177.0	9.3	6.4	14.8
198	283.6	42.0	66.2	25.5
199	232.1	8.6	8.7	18.4

200 rows × 4 columns

Dataset Description:

In [ ]:
```
The dataset consists of the following columns:
1.TV: Amount of money spent on TV advertising (in thousands of dollars).
2.Radio: Amount of money spent on Radio advertising (in thousands of dollars).
3.Newspaper: Amount of money spent on Newspaper advertising (in thousands of dollars).
4.Sales: Sales of the product (in thousands of units).
```

Data Understanding

In [21]:
```
code.head()
```

Out[21]:

	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	12.0
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9

In [22]:
```
code.describe()
```

Out[22]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

Initial check up

In [23]:
```
code.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   TV         200 non-null    float64
 1   Radio      200 non-null    float64
 2   Newspaper  200 non-null    float64
 3   Sales      200 non-null    float64
dtypes: float64(4)
memory usage: 6.4 KB
```

In [ ]:
```
#Asking Questions
#1.tv wise sales
#2.radio wise sales
#3.newspaper wise sales
#4.maximun sales on tv
#5.minimum sales on radio
#6.highest sales on newspaper
```

In [25]:
```
#1.tv wise sales
code['TV'].value_counts(normalize=True)*100
```

Out[25]:
```
TV
199.8    1.0
109.8    1.0
17.2     1.0
177.0    1.0
222.4    1.0
        ...
139.3    0.5
216.8    0.5
199.1    0.5
26.8     0.5
232.1    0.5
Name: proportion, Length: 190, dtype: float64
```

In [28]:
```
#2.radio wise sales
code['Radio'].value_counts(normalize=True)*100
```

Out[28]:
```
Radio
4.1     1.5
5.7     1.5
13.9    1.0
14.3    1.0
36.9    1.0
       ...
42.8    0.5
14.5    0.5
30.6    0.5
33.0    0.5
8.6     0.5
Name: proportion, Length: 167, dtype: float64
```

In [29]:
```
#3.newspaper wise sales
code['Newspaper'].value_counts(normalize=True)*100
```

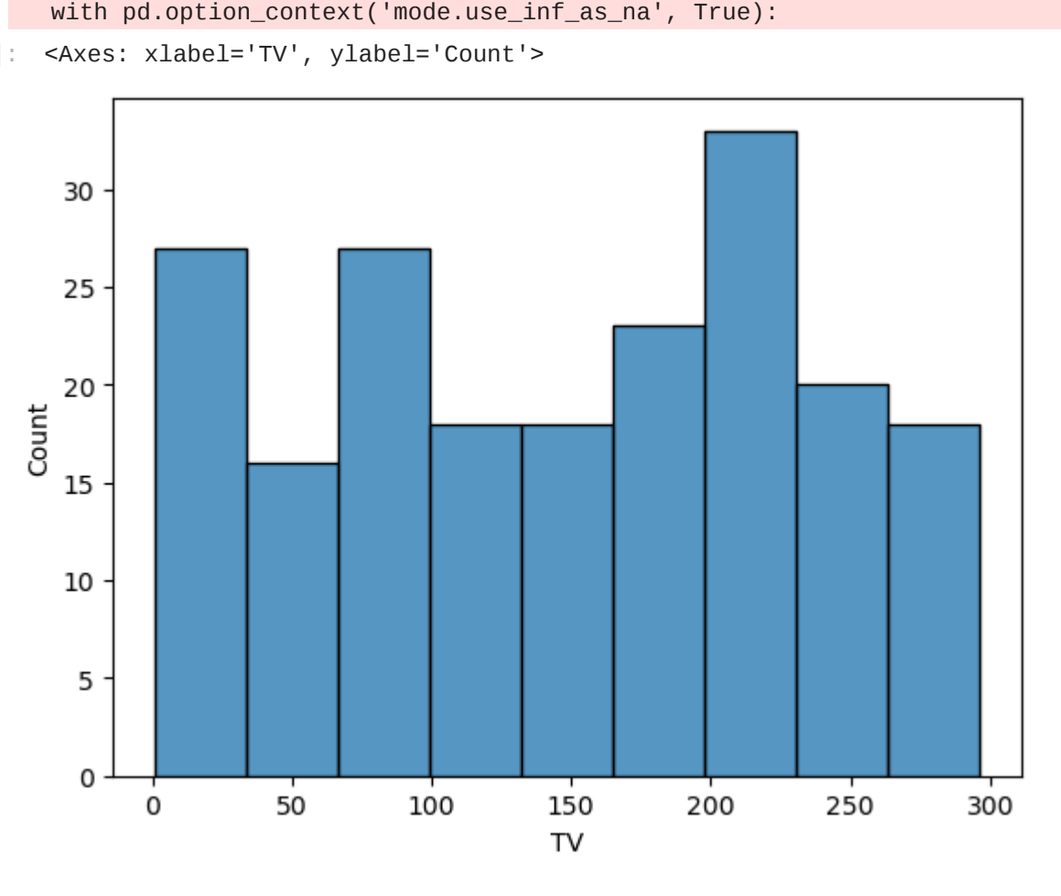
Out[29]:
```
Newspaper
9.3     1.5
25.6    1.5
8.7     1.5
34.6    1.0
8.5     1.0
       ...
27.2    0.5
31.7    0.5
19.3    0.5
31.3    0.5
66.2    0.5
Name: proportion, Length: 172, dtype: float64
```

Data Visualization

In [7]:
```
sns.histplot(code,x='TV')
```

```
C:\Users\jyoth\anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

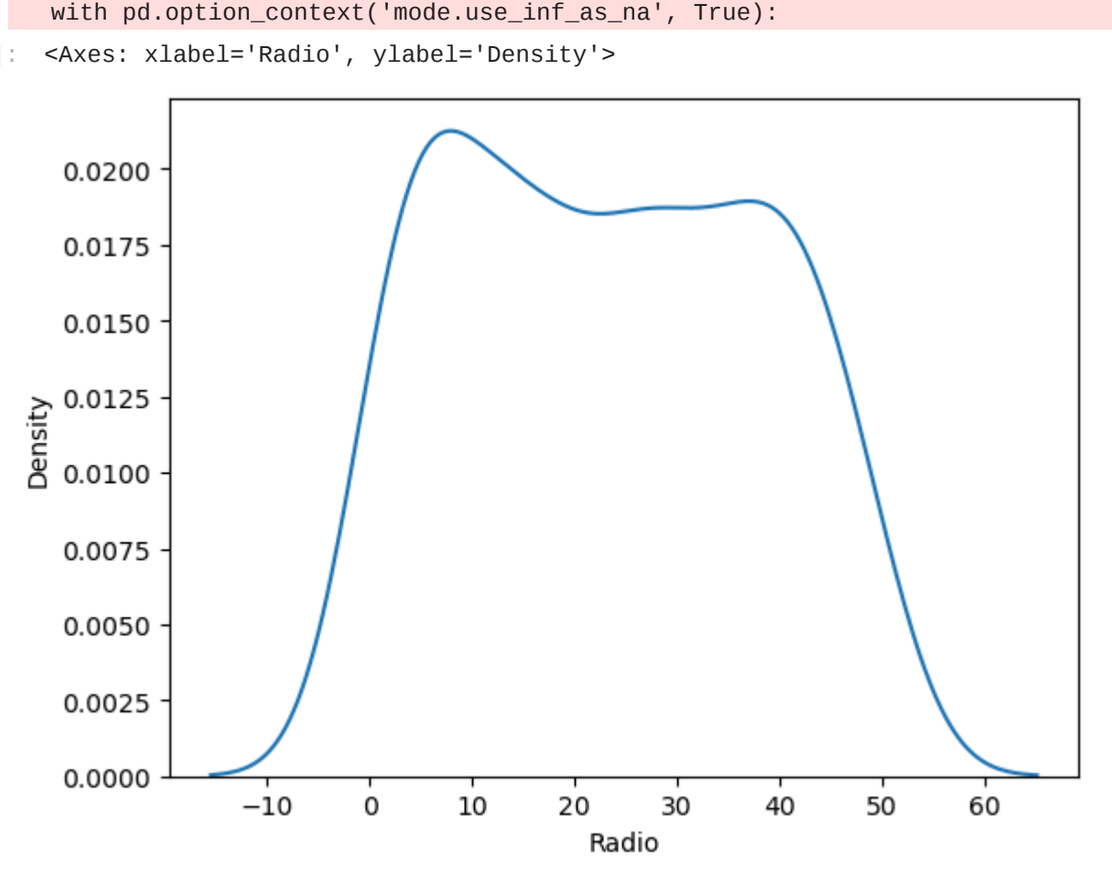
Out[7]: <Axes: xlabel='TV', ylabel='Count'>



In [8]:
```
sns.kdeplot(code,x='Radio')
```

```
C:\Users\jyoth\anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

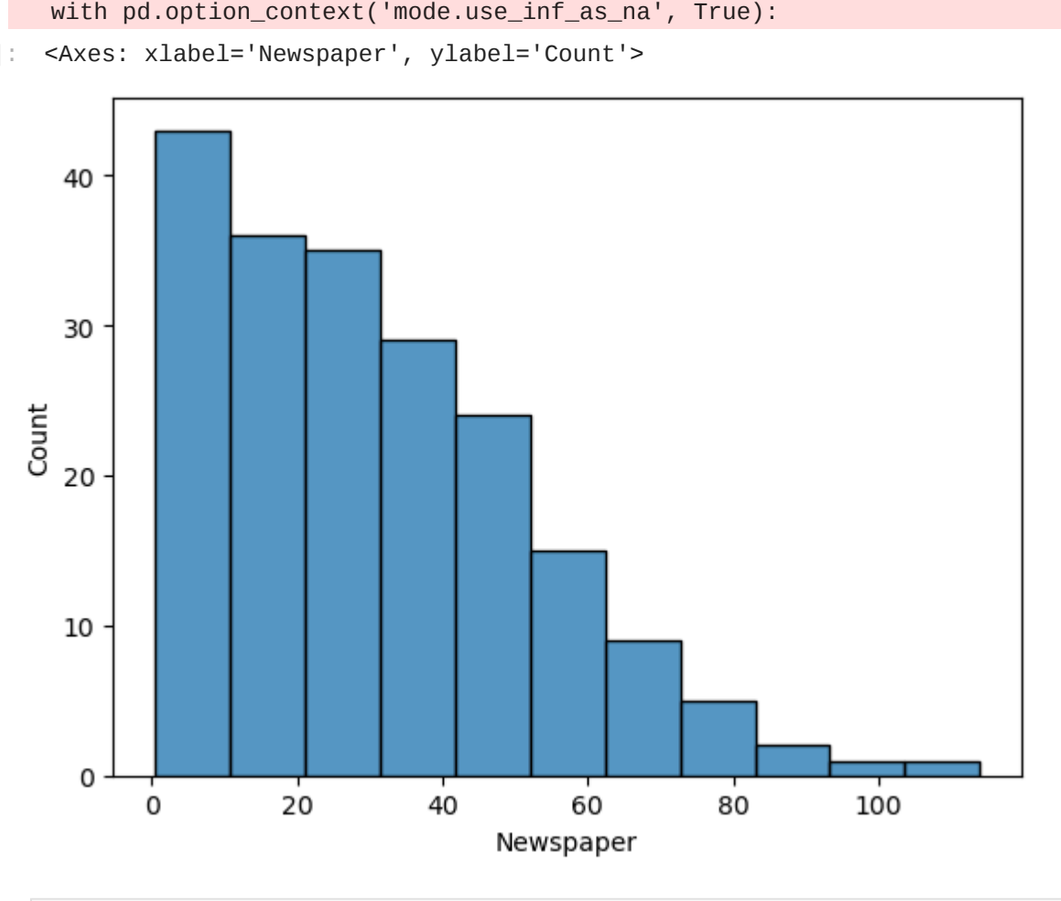
Out[8]: <Axes: xlabel='Radio', ylabel='Density'>



In [9]:
```
sns.histplot(code,x='Newspaper')
```

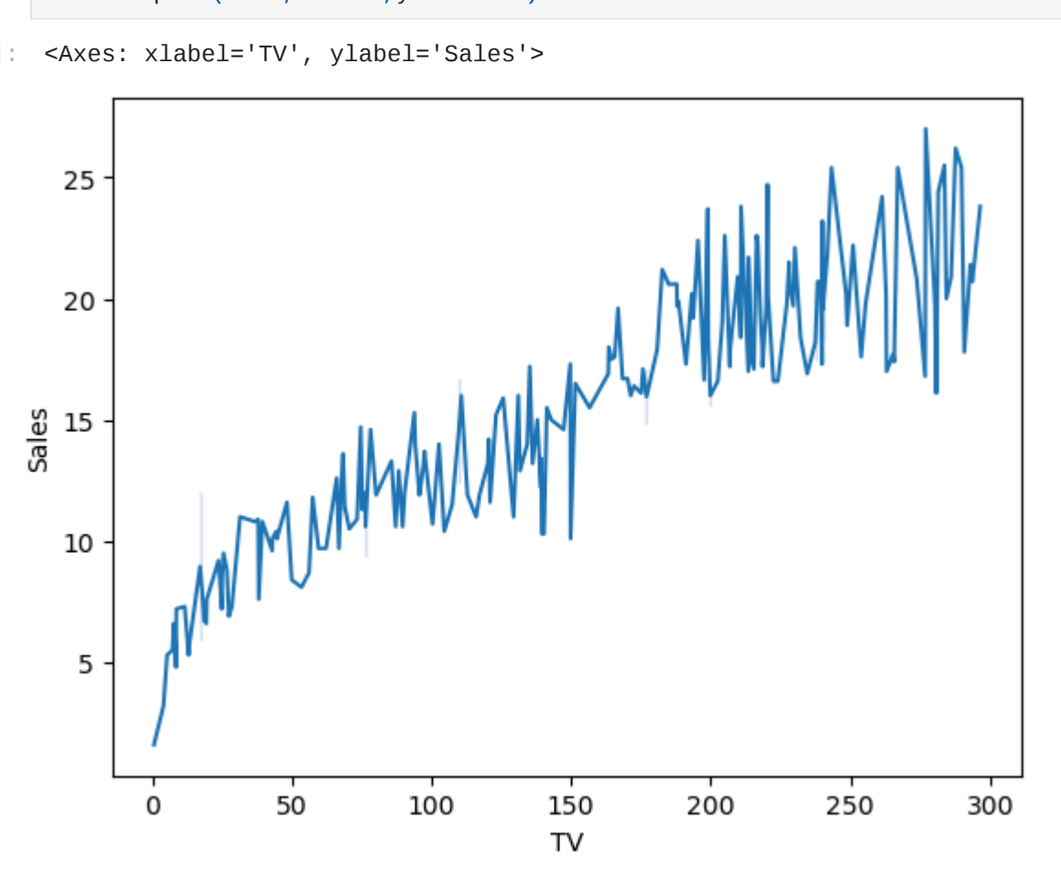
```
C:\Users\jyoth\anaconda\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

Out[9]: <Axes: xlabel='Newspaper', ylabel='Count'>



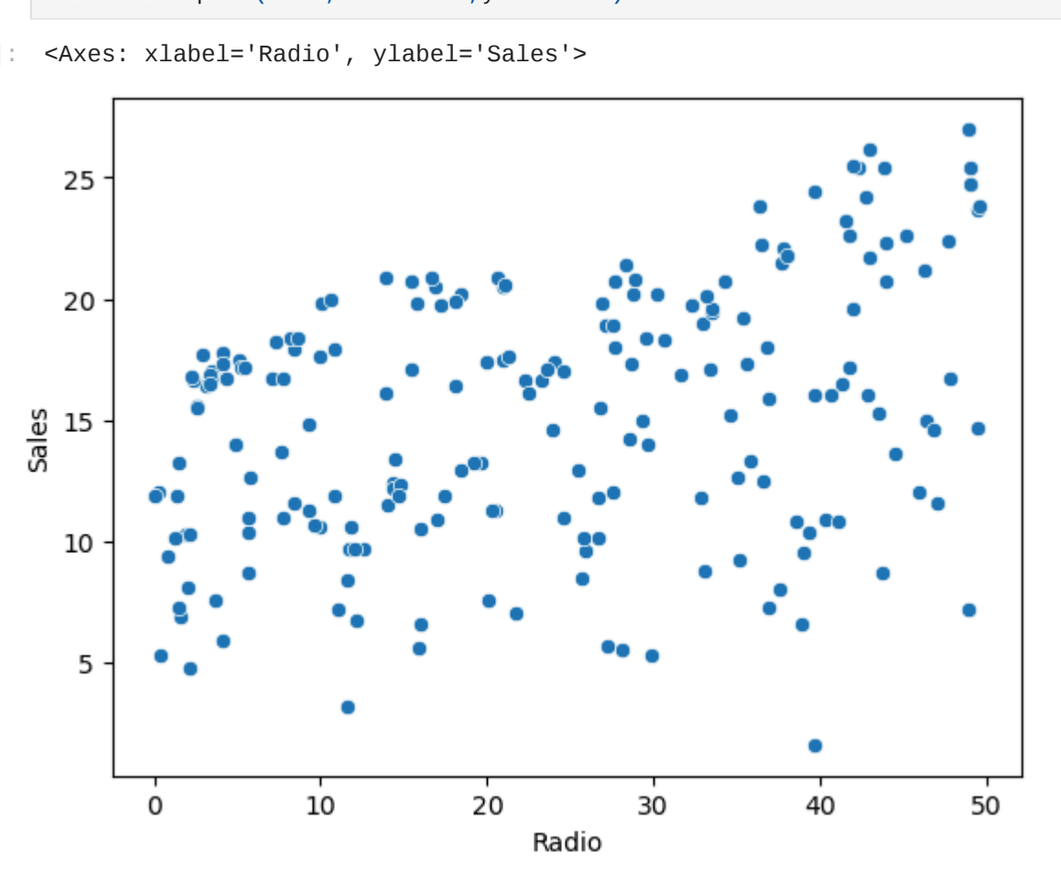
In [58]:
```
#4.maximun sales on tv
sns.lineplot(code,x='TV',y='Sales')
```

Out[58]: <Axes: xlabel='TV', ylabel='Sales'>



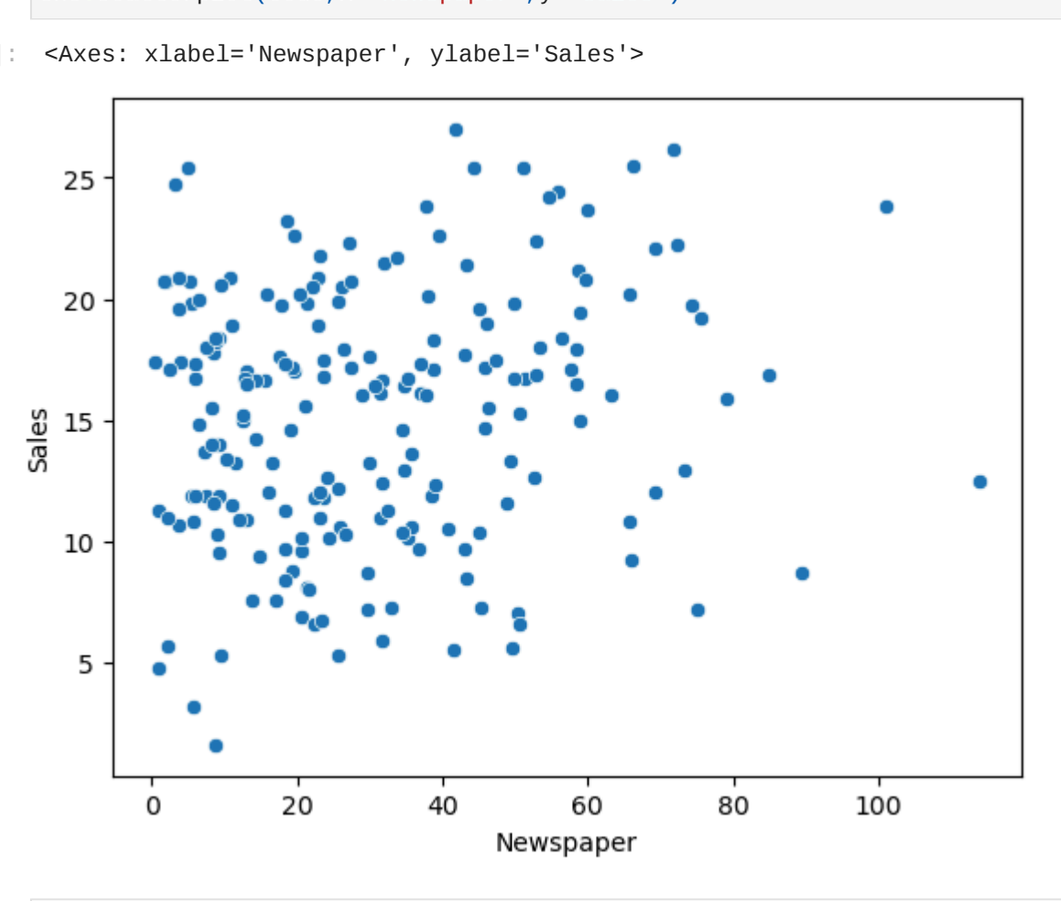
In [66]:
```
#5.minimum sales on radio
sns.scatterplot(code,x='Radio',y='Sales')
```

Out[66]: <Axes: xlabel='Radio', ylabel='Sales'>



In [14]:
```
#6.highest sales on newspaper
sns.scatterplot(code,x='Newspaper',y='Sales')
```

Out[14]: <Axes: xlabel='Newspaper', ylabel='Sales'>



In [70]:
```
code.shape
```

Out[70]: (200, 4)

In [12]:
```
X=code.iloc[:,0:1]
y=code.iloc[:,-1]
```

train_test_split

In [13]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=2)
X_train.head()
```

Out[13]:

	TV
137	273.7
163	163.5
111	241.7
123	123.1
109	255.4

In [14]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
```

Fit

In [17]:
```
lr.fit(X_train,y_train)
```

Out[17]:
```
▾ LinearRegression
LinearRegression()
```

In [16]:
```
X_train.head()
```

Out[16]:

	TV
137	273.7
163	163.5
111	241.7
123	123.1
109	255.4

In [34]:
```
lr.coef_
```

Out[34]: array([0.05838585])

In [35]:
```
lr.intercept_
```

Out[35]: 6.57452093430827

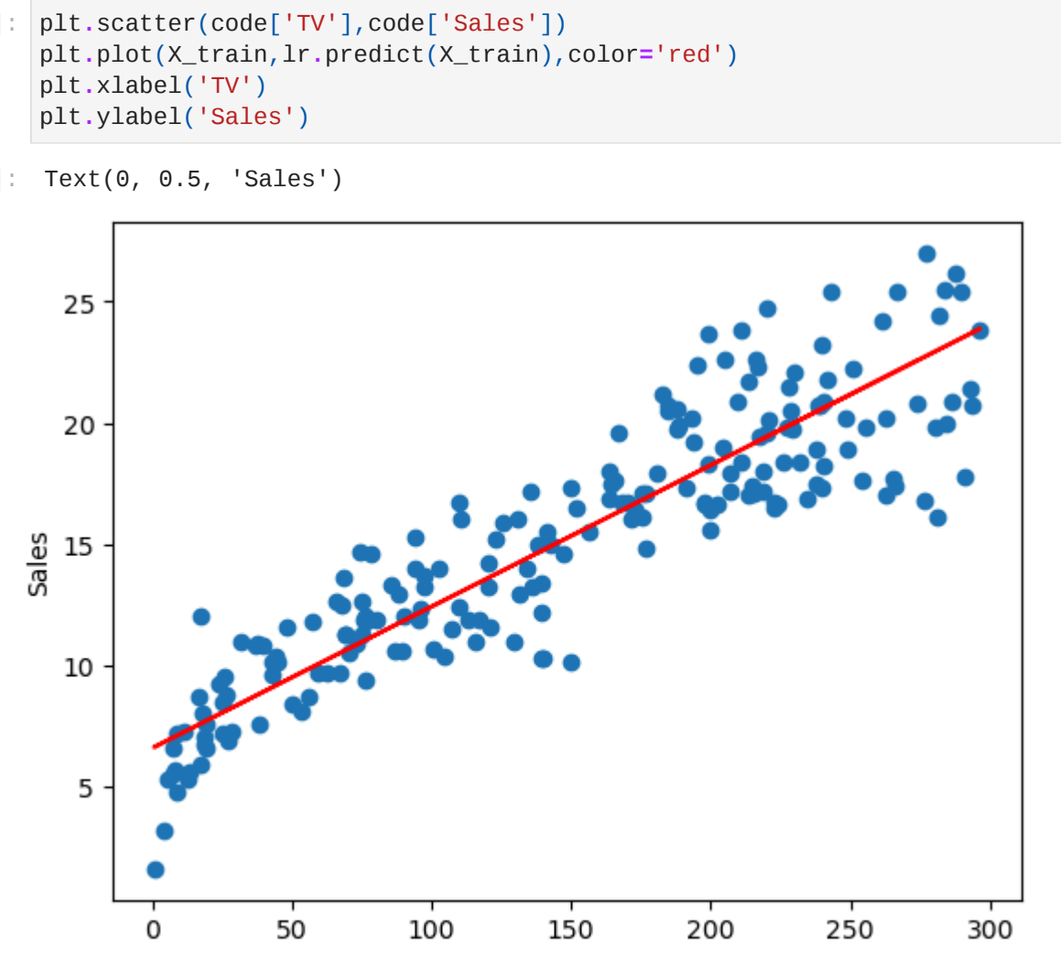
In [36]:
```
lr.predict([[8.54]])
```

```
C:\Users\jyoth\anaconda\Lib\site-packages\sklearn\base.py:439: UserWarning: X does not have valid feature names, but LinearRegression was fitted with feature names
  warnings.warn(
```

Out[36]: array([7.07313608])

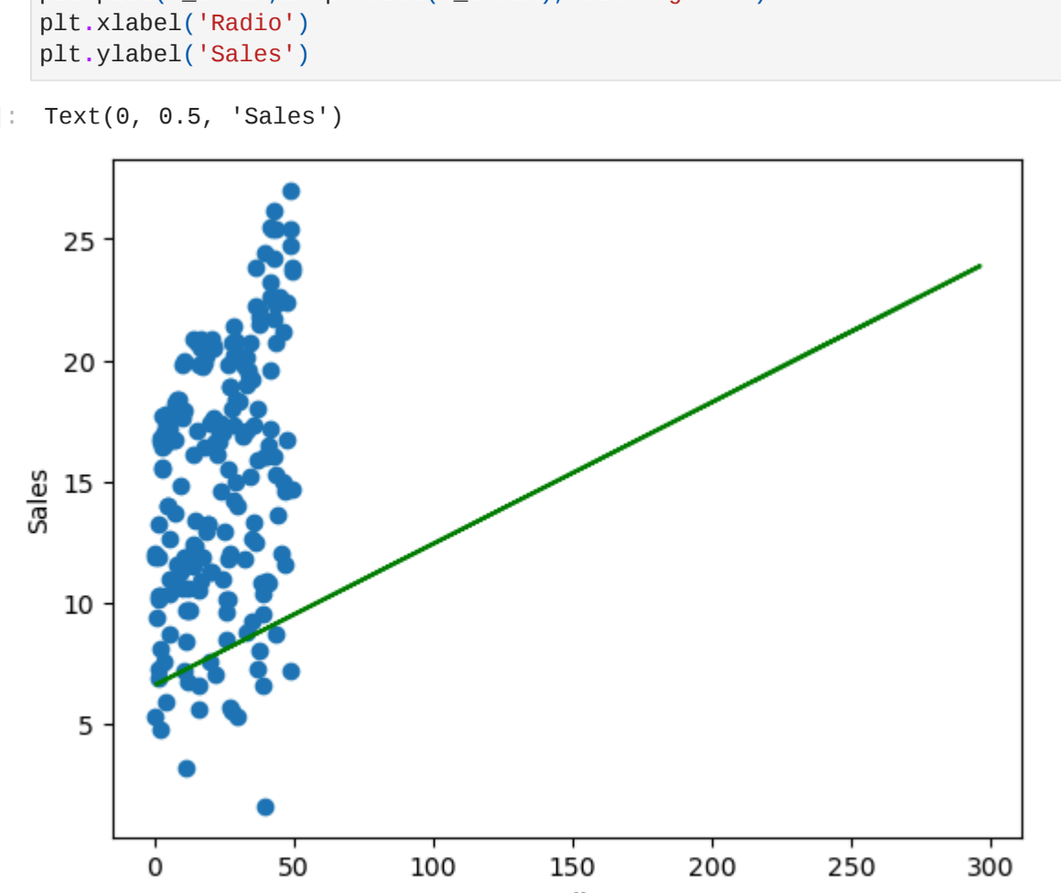
In [18]:
```
plt.scatter(code['TV'],code['Sales'])
plt.plot(X_train,lr.predict(X_train),color='red')
plt.xlabel('TV')
plt.ylabel('Sales')
```

Out[18]: Text(0, 0.5, 'Sales')



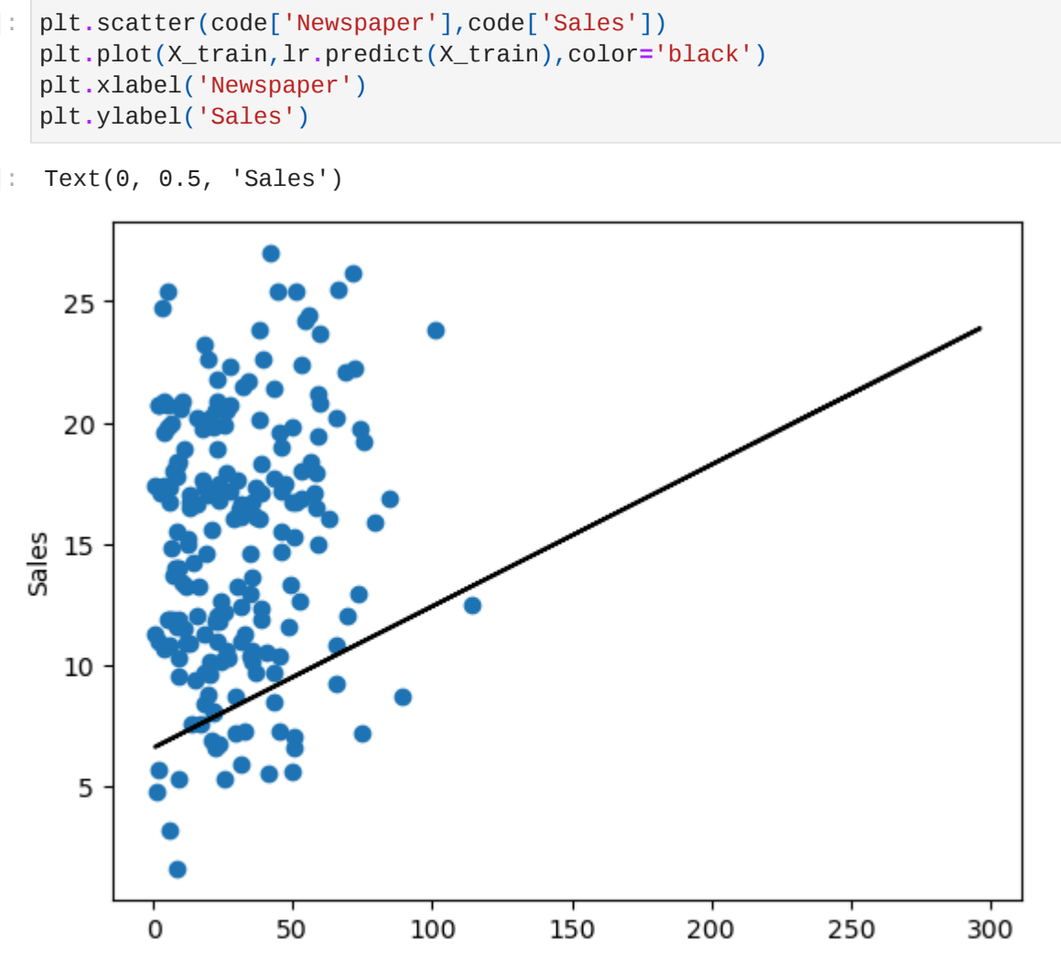
In [19]:
```
plt.scatter(code['Radio'],code['Sales'])
plt.plot(X_train,lr.predict(X_train),color='green')
plt.xlabel('Radio')
plt.ylabel('Sales')
```

Out[19]: Text(0, 0.5, 'Sales')



In [20]:
```
plt.scatter(code['Newspaper'],code['Sales'])
plt.plot(X_train,lr.predict(X_train),color='black')
plt.xlabel('Newspaper')
plt.ylabel('Sales')
```

Out[20]: Text(0, 0.5, 'Sales')



Model Evalution

In [21]:
```
from sklearn.metrics import mean_absolute_error,mean_squared_error
y_pred=lr.predict(X_test)
y_test.values
```

Out[21]:
```
array([17.1, 10.5,  8.7, 18.4, 19.6, 20.2, 12.6, 20.2, 12.6, 17.8,  9.2,
       19.4, 13.7,  6.6, 16.5, 12. ,  8.5, 16.5, 20.9, 19. , 20.5, 17. ,
       11.8, 10.8, 16.9, 16.7, 16.1, 11.3, 19.8, 16.6, 16.8, 16. , 15.6,
       13.2,  5.3, 12.4, 24.7, 20.2, 17.7, 21.2])
```

In [22]:
```
mean_absolute_error(y_test,y_pred)
```

Out[22]: 2.0246863980546763

In [23]:
```
mean_squared_error(y_test,y_pred)
```

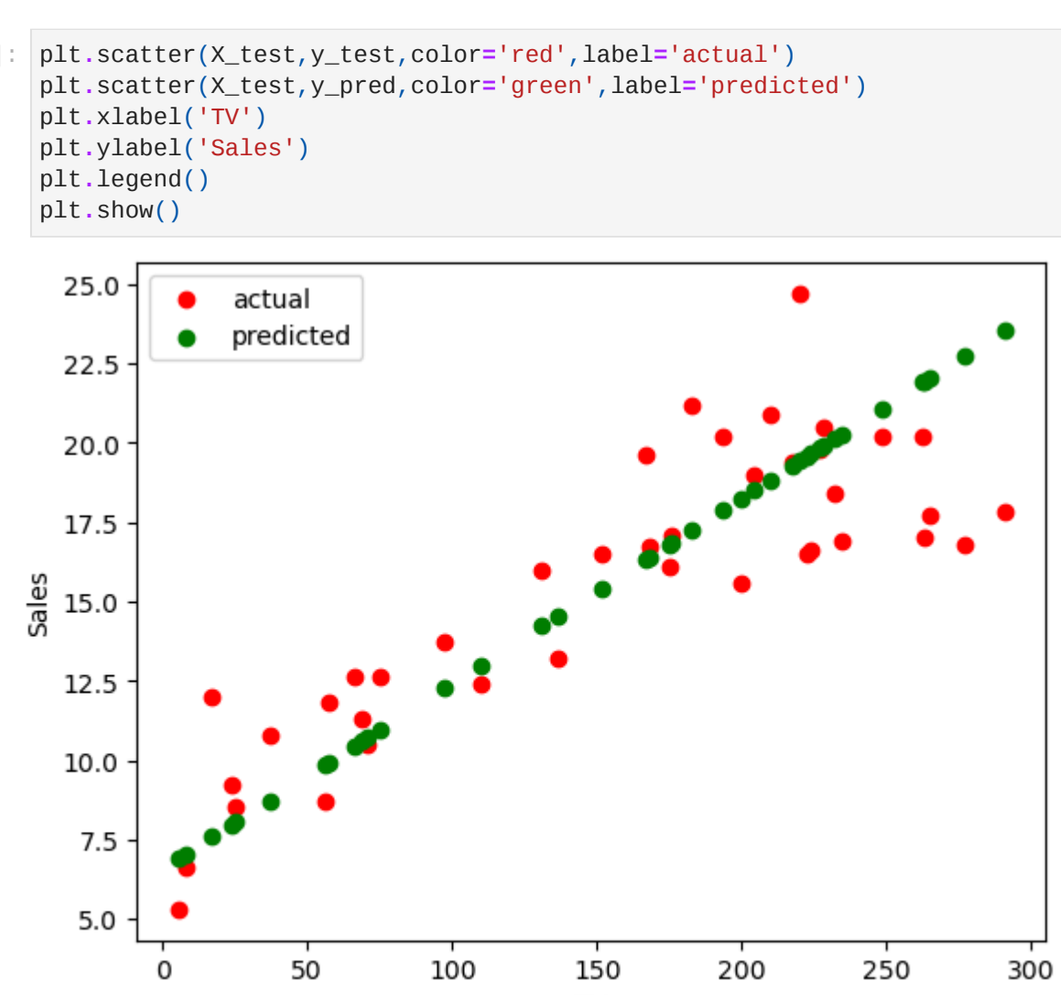
Out[23]: 6.7483191968371345

In [24]:
```
MSE=mean_squared_error(y_test,y_pred)
RMSE=MSE**0.5
data={'actual(y_test)':y_test,'predicted(y_pred)':y_pred}
df=pd.DataFrame(data)
df.head()
```

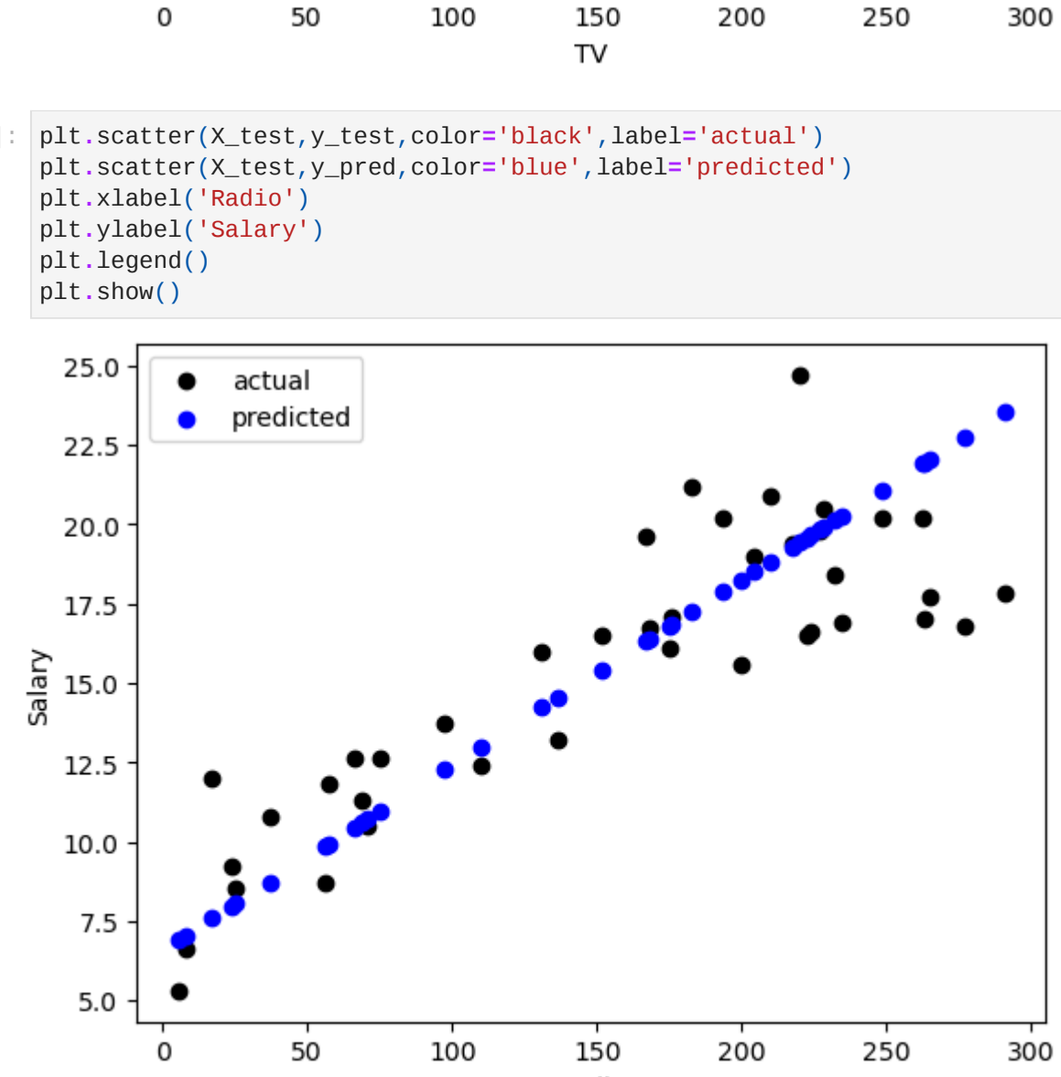
Out[24]:

	actual(y_test)	predicted(y_pred)
112	17.1	16.832915
29	10.5	10.696562
182	8.7	9.855806
199	18.4	20.125876
193	19.6	16.313281

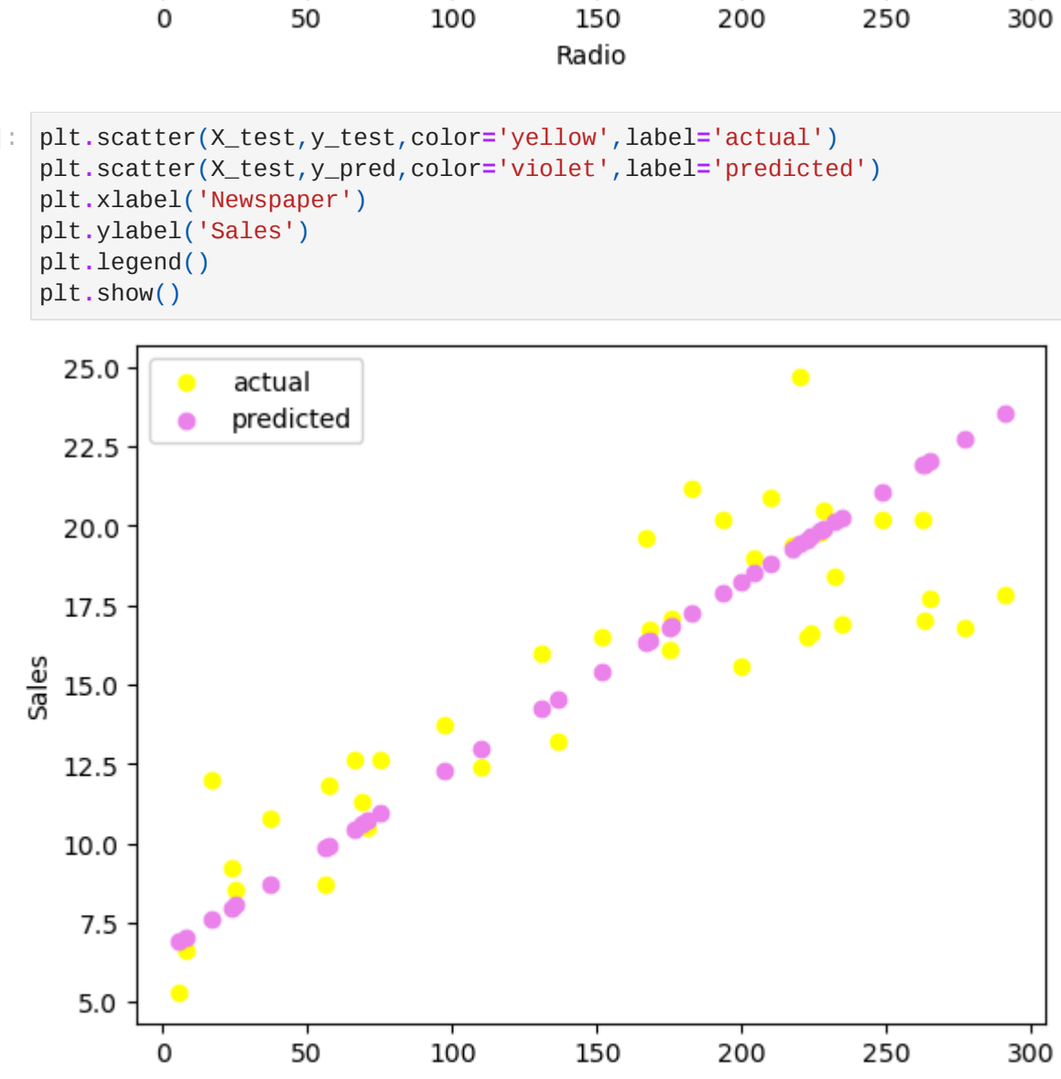
In [25]:
```
plt.scatter(X_test,y_test,color='red',label='actual')
plt.scatter(X_test,y_pred,color='green',label='predicted')
plt.xlabel('TV')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



In [26]:
```
plt.scatter(X_test,y_test,color='black',label='actual')
plt.scatter(X_test,y_pred,color='blue',label='predicted')
plt.xlabel('Radio')
plt.ylabel('Salary')
plt.legend()
plt.show()
```



In [27]:
```
plt.scatter(X_test,y_test,color='yellow',label='actual')
plt.scatter(X_test,y_pred,color='violet',label='predicted')
plt.xlabel('Newspaper')
plt.ylabel('Sales')
plt.legend()
plt.show()
```



In [ ]:
```
#Insights
#we have to import the advertising dataset based on the problem description
#the head() it takes the first five rows of the dataset and  describe() it calculates the count,mean,std,min,max of each column of the dataset
#30 above tv are the highest count of the plot and the least count of the tv is 15
#maximum sales on tv is 25 above the sales on tv should be increased
#the scatterplot defines the relationship bewteen two numerical variables and it display the plot
#by using scatterplot it display the x and y labels are the numerical variables
#we have to import linear regression and fit the x_train and y_train
#coefficient,intercept,predict are performed on the dataset by using linear regression
#model evaluation is implemented to predict the mean_absolute_error,mean_squared_error
#actual data and predicted data are performed in MSE and RMSE
```