Oasis Infobyte

Task 5 : Sales Prediction using python

sales prediction means predicting how much of a product people will buy based on factors such as the amount you spend to advertise your product, the segment of people you advertise for, or the platform you are advertising on about your product.

Typically, a product and service-based business always need their Data Scientist to predict their future sales with every step they take to manipulate the cost of advertising their product. So let’s start the task of sales prediction with machine learning using Python.

1. Import all necessary

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

2. Import dataframe

In [2]:
```
df = pd.read_csv("Advertising.csv")
```

In [3]:
```
df
```

Out[3]:

	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 × 5 columns

In [4]:
```
df.drop(['Unnamed: 0'],axis=1,inplace=True)
```

In [5]:
```
df
```

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	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

3. Check for Null Values

In [6]:
```
df.isnull().sum()
```

Out[6]:
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

we don't have null values

4. Check for Dublicate row

In [7]:
```
df.duplicated().sum()
```

Out[7]:
```
0
```

we don't have dublicate row

5. Summery of data

In [8]:
```
df.describe()
```

Out[8]:

	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

In [9]:
```
df.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
```

6. Check column name

In [10]:
```
df.columns
```

Out[10]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

7. Check the datatype

In [11]:
```
df.dtypes
```

Out[11]:
```
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object
```

8.Shape of dataset

In [12]:
```
df.shape
```

Out[12]:
```
(200, 4)
```

9. Find Corelation of data

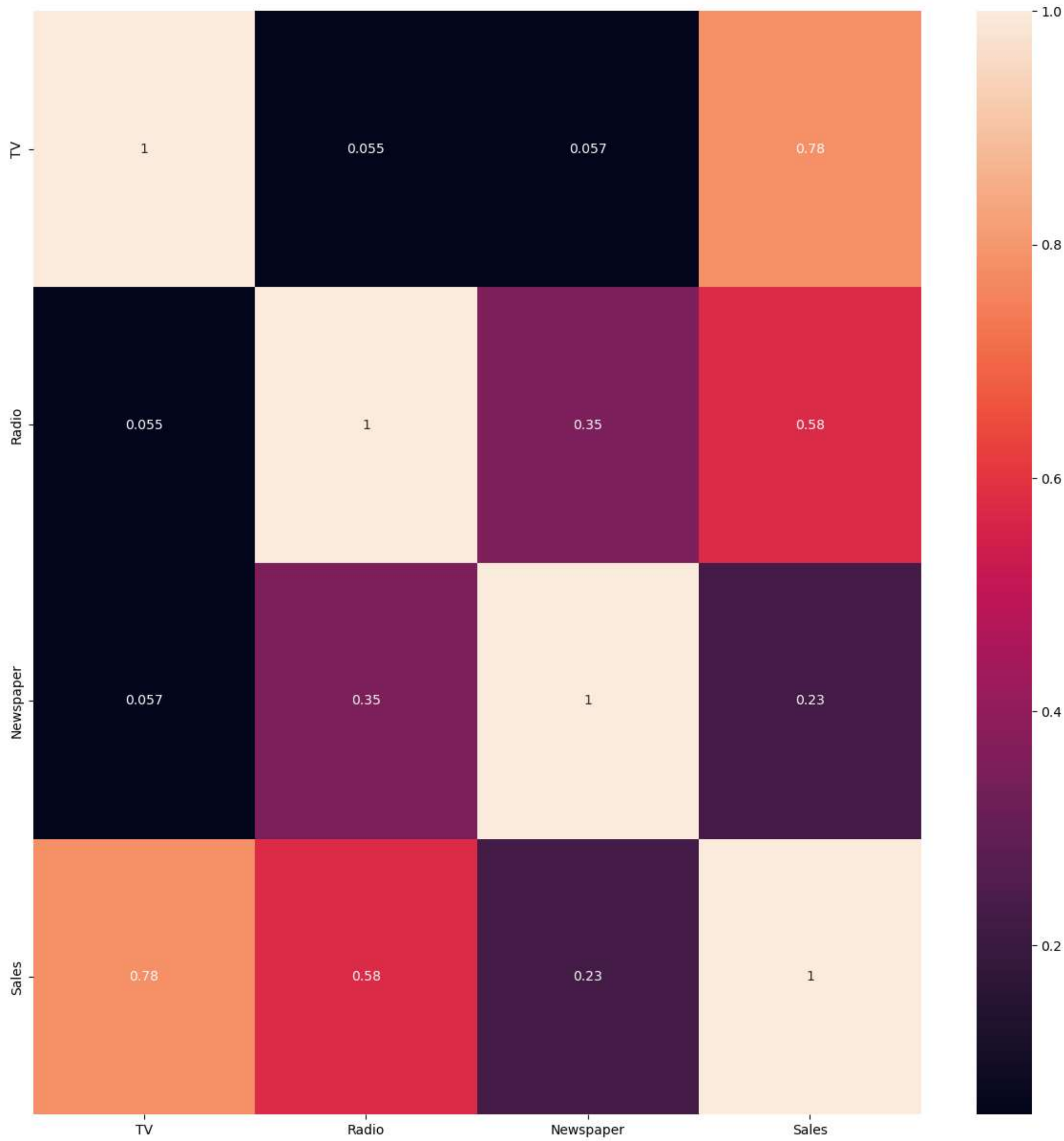
In [13]:
```
df.corr()
```

Out[13]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

In [14]:
```
plt.figure(figsize=(15,15))
sns.heatmap(df.corr() , annot=True)
```

Out[14]: <AxesSubplot:>



10. Analysing the 'Sales' column

In [15]:
```
df["Sales"]
```

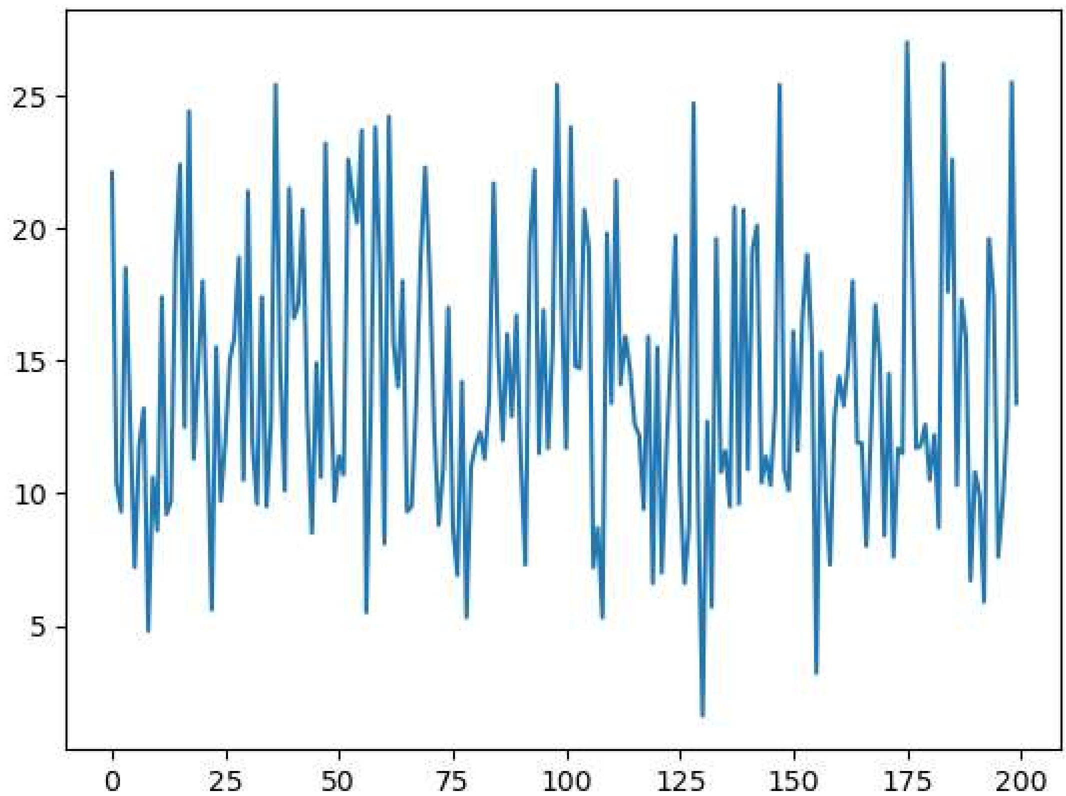
Out[15]:
```
0      22.1
1      10.4
2       9.3
3      18.5
4      12.9
       ...
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4
Name: Sales, Length: 200, dtype: float64
```

In [16]:
```
df["Sales"].describe()
```

Out[16]:
```
count    200.000000
mean      14.022500
std        5.217457
min        1.600000
25%       10.375000
50%       12.900000
75%       17.400000
max       27.000000
Name: Sales, dtype: float64
```

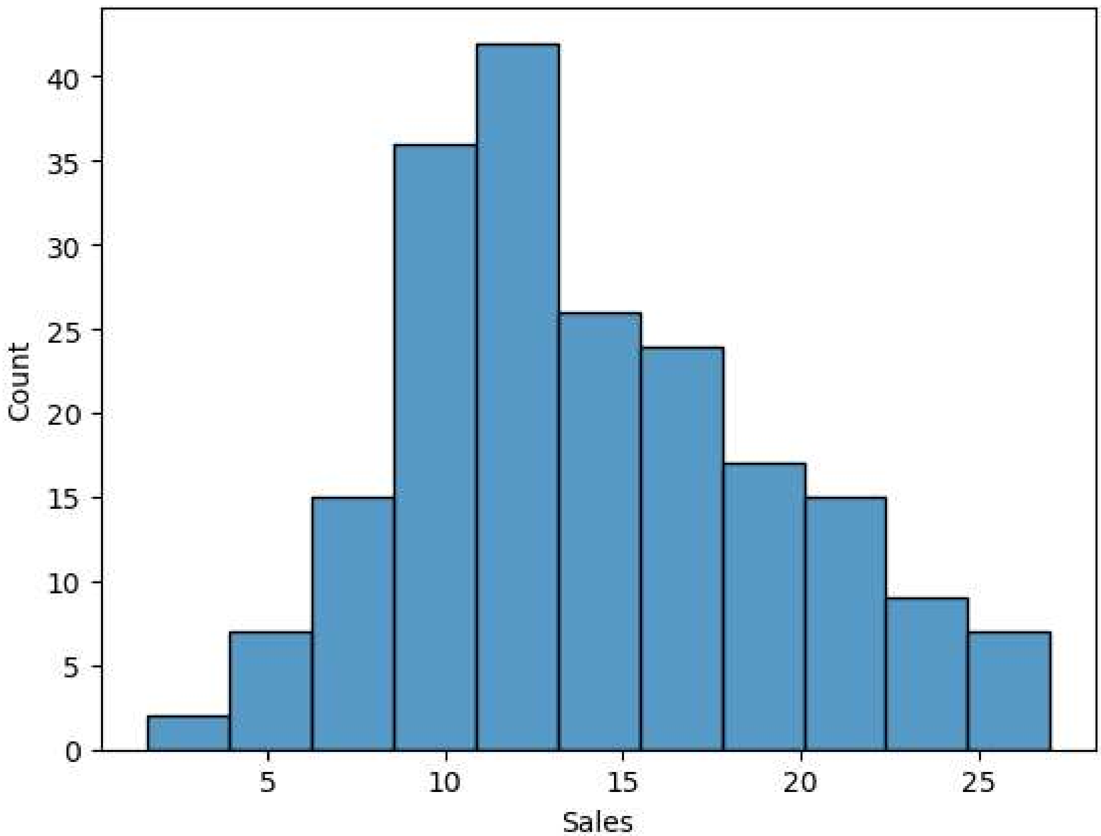
In [17]:
```
plt.plot(df["Sales"])
```

Out[17]: [<matplotlib.lines.Line2D at 0x1d2d4736dc0>]



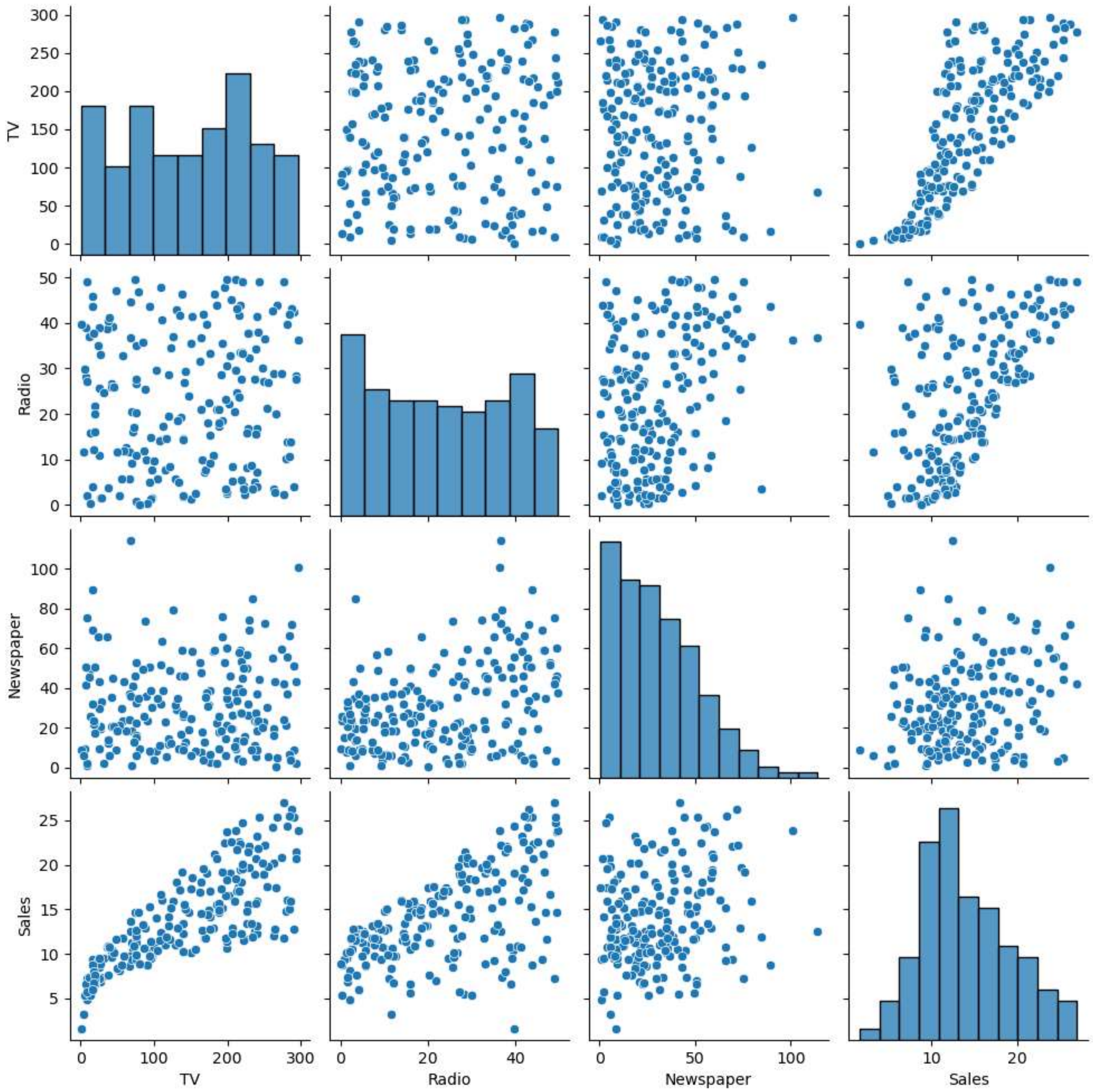
In [18]:
```
sns.histplot(df["Sales"])
```

Out[18]: <AxesSubplot:xlabel='Sales', ylabel='Count'>



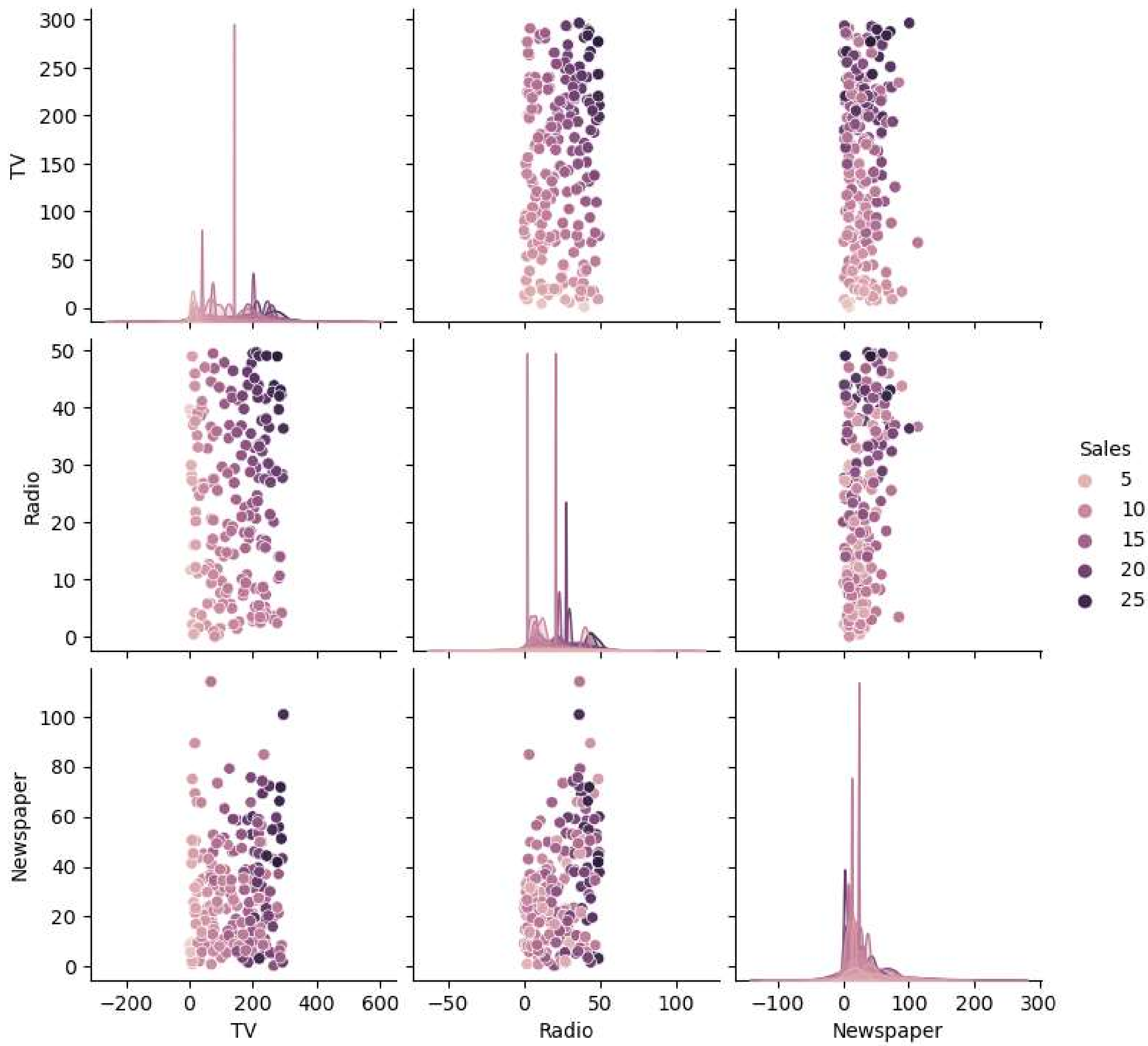
In [19]:
```
sns.pairplot(df )
```

Out[19]: <seaborn.axisgrid.PairGrid at 0x1d2d3356520>

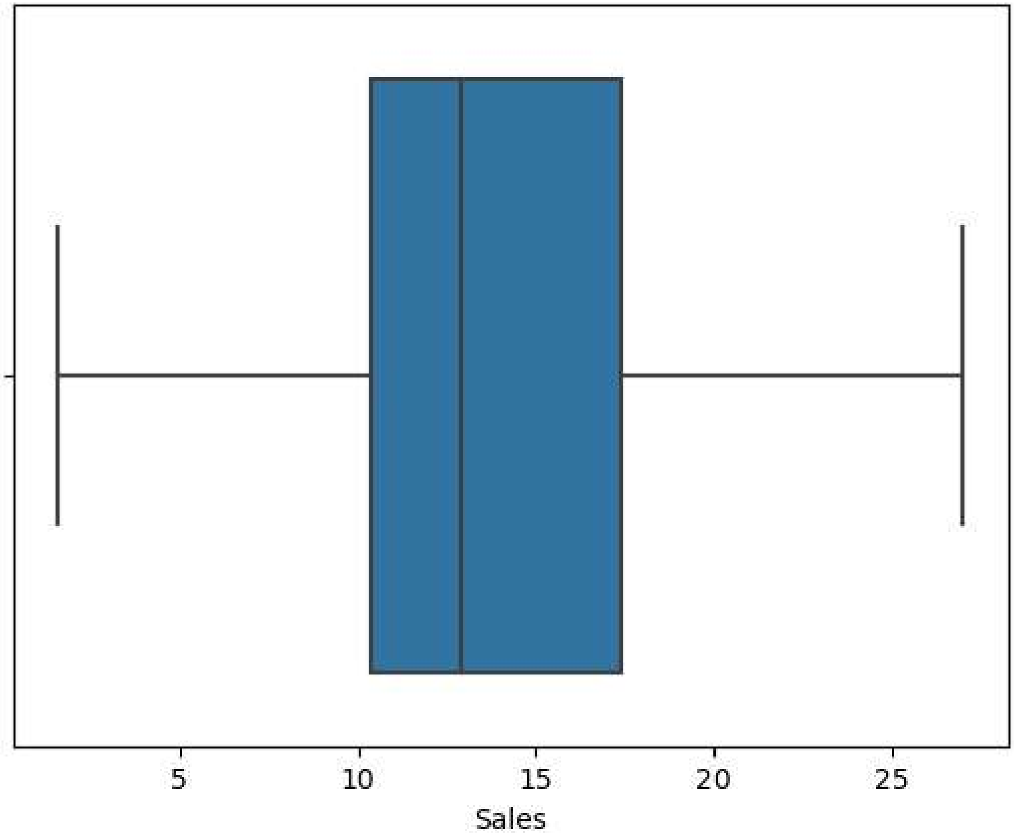


In [20]:
```
sns.pairplot(df , hue = "Sales")
```

Out[20]: <seaborn.axisgrid.PairGrid at 0x1d2d34d6df0>



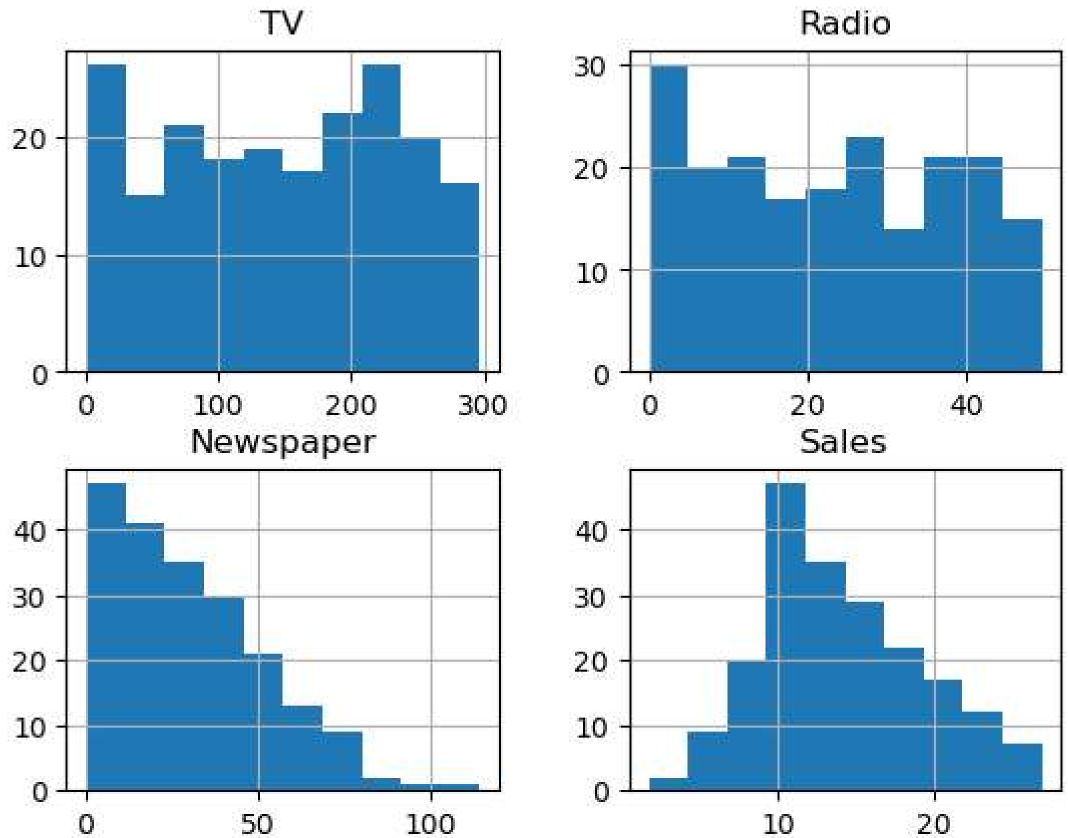
In [21]:
```
sns.boxplot(df['Sales'])
plt.show()
```



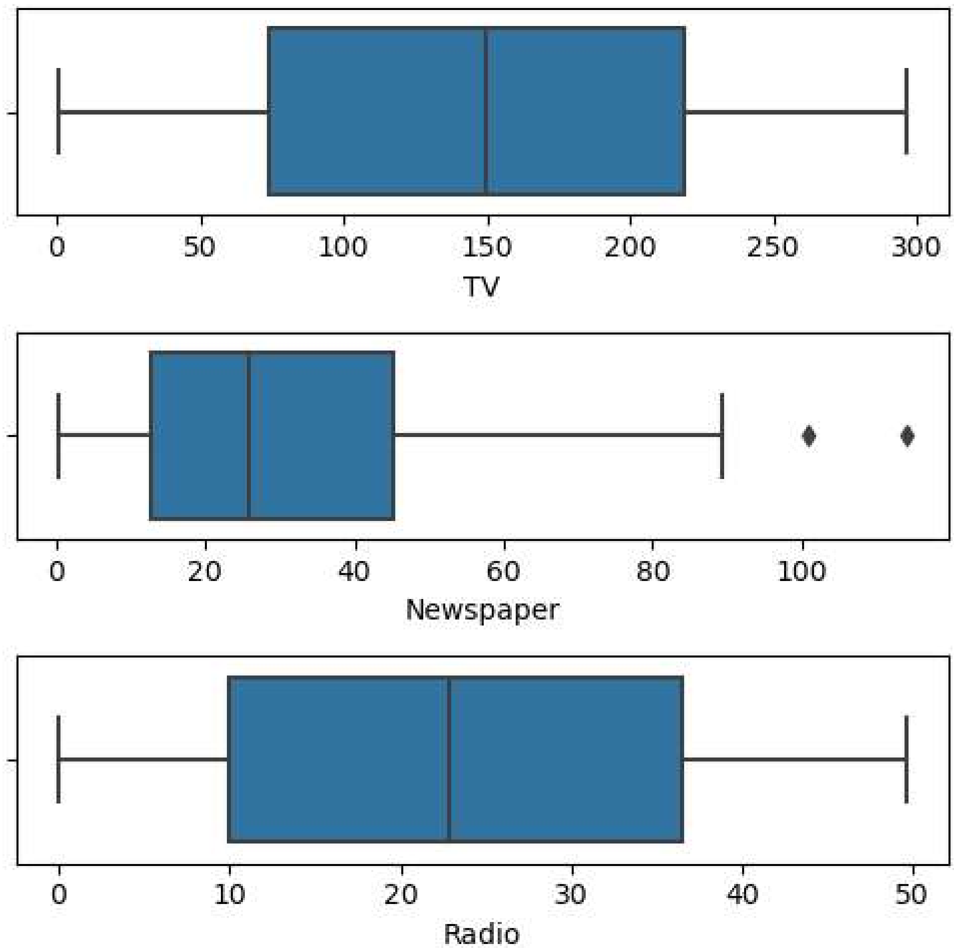
11. Analysing the "TV" , "Radio" , "Newspaper" column

In [22]:
```
df.hist()
```

Out[22]:
```
array([[<AxesSubplot:title={'center':'TV'}>,
        <AxesSubplot:title={'center':'Radio'}>],
       [<AxesSubplot:title={'center':'Newspaper'}>,
        <AxesSubplot:title={'center':'Sales'}>]], dtype=object)
```

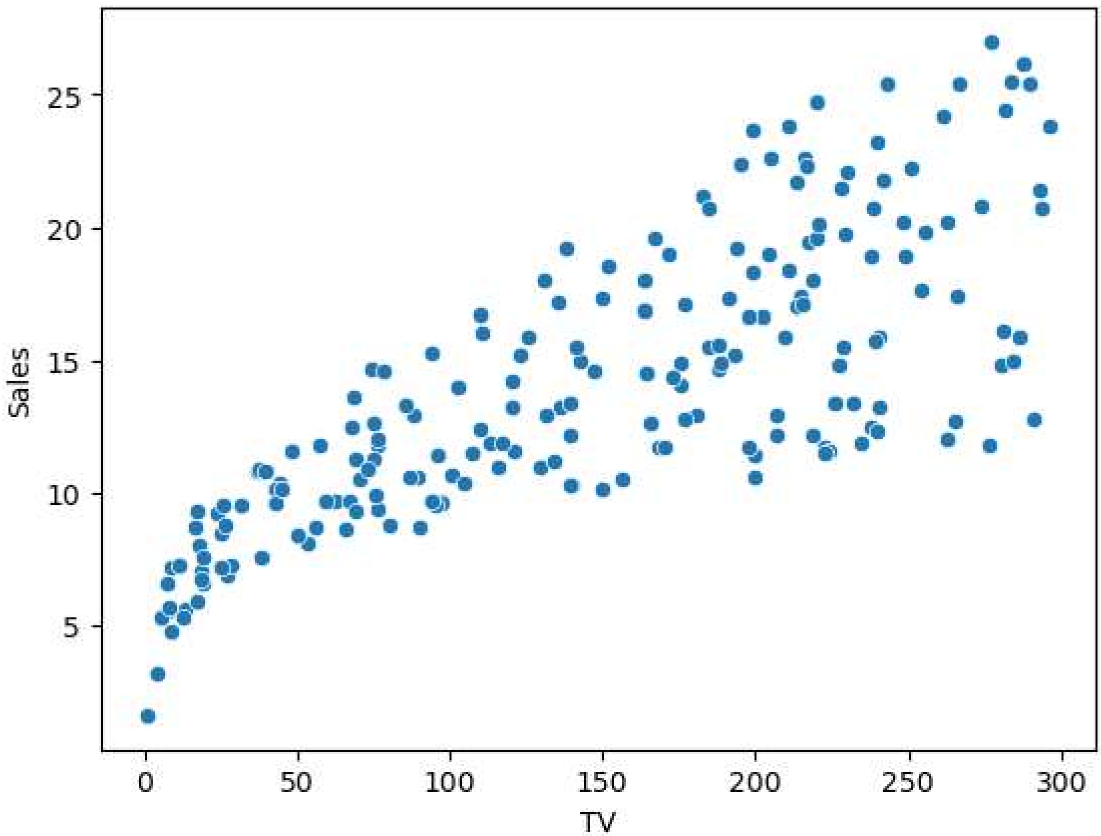


In [23]:
```
fig,axs=plt.subplots(3,figsize=(5,5))
plt1= sns.boxplot(df['TV'],ax=axs[0])
plt2= sns.boxplot(df['Newspaper'],ax=axs[1])
plt3= sns.boxplot(df['Radio'],ax=axs[2])
plt.tight_layout()
```



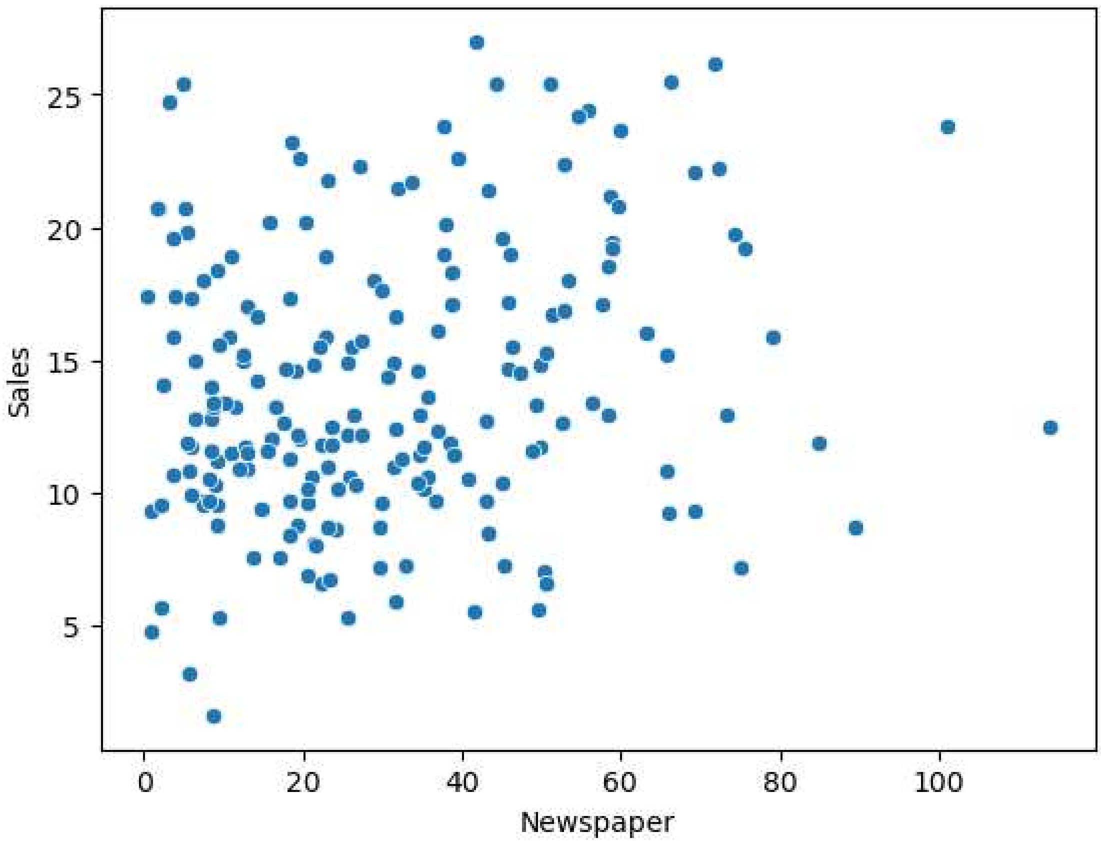
In [24]:
```
sns.scatterplot(x="TV" , y = "Sales" , data = df    )
```

Out[24]:
```
<AxesSubplot:xlabel='TV', ylabel='Sales'>
```



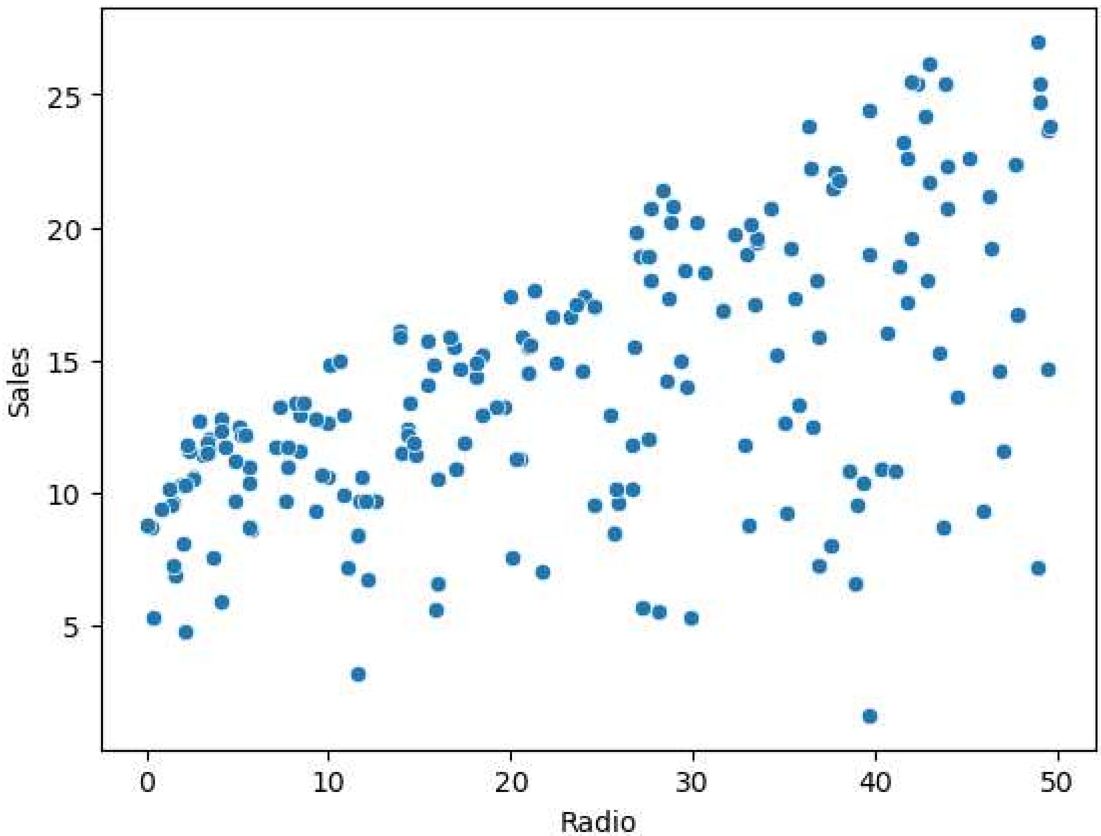
In [25]:
```
sns.scatterplot(x="Newspaper" , y = "Sales" , data = df    )
```

Out[25]: <AxesSubplot:xlabel='Newspaper', ylabel='Sales'>



In [26]:
```
sns.scatterplot(x="Radio" , y = "Sales" , data = df    )
```

Out[26]:
```
<AxesSubplot:xlabel='Radio', ylabel='Sales'>
```



In [27]:
```
df.columns
```

Out[27]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [28]:
```
df.dtypes
```

Out[28]:
```
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object
```

12. Split the data

In [29]:
```
x = df.drop("Sales" , axis = 1)
```

In [30]:
```
y = df["Sales"]
```

In [31]:
```
x
```

Out[31]:

	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 [32]:
```
y
```

Out[32]:
```
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
```

13. Let's Standardize the data

In [33]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [34]:
```
x_scaled = scaler.fit_transform(x)
```

In [35]:
```
from sklearn.model_selection import  train_test_split

x_train , x_test , y_train , y_test =  train_test_split(x,y,test_size=0.20)
```

In [36]:
```
x_test
```

Out[36]:

	TV	Radio	Newspaper
140	73.4	17.0	12.9
191	75.5	10.8	6.0
167	206.8	5.2	19.4
108	13.1	0.4	25.6
113	209.6	20.6	10.7
199	232.1	8.6	8.7
56	7.3	28.1	41.4
134	36.9	38.6	65.6
12	23.8	35.1	65.9
183	287.6	43.0	71.8
88	88.3	25.5	73.4
105	137.9	46.4	59.0
182	56.2	5.7	29.7
50	199.8	3.1	34.6
60	53.5	2.0	21.4
126	7.8	38.9	50.6
63	102.7	29.6	8.4
78	5.4	29.9	9.4
9	199.8	2.6	21.2
172	19.6	20.1	17.0
41	177.0	33.4	38.7
159	131.7	18.4	34.6
94	107.4	14.0	10.9
11	214.7	24.0	4.0
14	204.1	32.9	46.0
121	18.8	21.7	50.4
190	39.5	41.1	5.8
120	141.3	26.8	46.2
163	163.5	36.8	7.4
3	151.5	41.3	58.5
90	134.3	4.9	9.3
111	241.7	38.0	23.2
10	66.1	5.8	24.2
107	90.4	0.3	23.2
135	48.3	47.0	8.5
102	280.2	10.1	21.4

	TV	Radio	Newspaper
38	43.1	26.7	35.1
42	293.6	27.7	1.8
198	283.6	42.0	66.2
80	76.4	26.7	22.3

In [37]:
```
x_test
```

Out[37]:

	TV	Radio	Newspaper
140	73.4	17.0	12.9
191	75.5	10.8	6.0
167	206.8	5.2	19.4
108	13.1	0.4	25.6
113	209.6	20.6	10.7
199	232.1	8.6	8.7
56	7.3	28.1	41.4
134	36.9	38.6	65.6
12	23.8	35.1	65.9
183	287.6	43.0	71.8
88	88.3	25.5	73.4
105	137.9	46.4	59.0
182	56.2	5.7	29.7
50	199.8	3.1	34.6
60	53.5	2.0	21.4
126	7.8	38.9	50.6
63	102.7	29.6	8.4
78	5.4	29.9	9.4
9	199.8	2.6	21.2
172	19.6	20.1	17.0
41	177.0	33.4	38.7
159	131.7	18.4	34.6
94	107.4	14.0	10.9
11	214.7	24.0	4.0
14	204.1	32.9	46.0
121	18.8	21.7	50.4
190	39.5	41.1	5.8
120	141.3	26.8	46.2
163	163.5	36.8	7.4
3	151.5	41.3	58.5
90	134.3	4.9	9.3
111	241.7	38.0	23.2
10	66.1	5.8	24.2
107	90.4	0.3	23.2
135	48.3	47.0	8.5
102	280.2	10.1	21.4

	TV	Radio	Newspaper
38	43.1	26.7	35.1
42	293.6	27.7	1.8
198	283.6	42.0	66.2
80	76.4	26.7	22.3

In [38]:
```
y_train
```

Out[38]:
```
79     11.0
55     23.7
122    11.6
141    19.2
170     8.4
       ... 
186    10.3
150    16.1
173    11.7
148    10.9
161    13.3
Name: Sales, Length: 160, dtype: float64
```

In [39]:
```
y_test
```

Out[39]:
```
140    10.9
191     9.9
167    12.2
108     5.3
113    15.9
199    13.4
56      5.5
134    10.8
12      9.2
183    26.2
88     12.9
105    19.2
182     8.7
50     11.4
60      8.1
126     6.6
63     14.0
78      5.3
9      10.6
172     7.6
41     17.1
159    12.9
94     11.5
11     17.4
14     19.0
121     7.0
190    10.8
120    15.5
163    18.0
3      18.5
90     11.2
111    21.8
10      8.6
107     8.7
135    11.6
102    14.8
38     10.1
42     20.7
198    25.5
80     11.8
Name: Sales, dtype: float64
```

14. Apply models

In [40]:
```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

In [41]:
```
model.fit(x_train, y_train)
```

Out[41]:
```
LinearRegression()
```

In [42]:
```
y_Pred = model.predict(x_test)
```

15. check model

In [43]:
```
from sklearn.metrics import r2_score
```

In [44]:
```
r2 = r2_score(y_test,y_Pred)
r2
```

Out[44]:
```
0.895364233937922
```

In [45]:
```
from sklearn.metrics import mean_squared_error,mean_absolute_error
```

In [46]:
```
MAE = mean_absolute_error(y_test,y_Pred)
MAE
```

Out[46]:
```
1.3839297223907647
```

In [47]:
```
np.sqrt(MAE) # root mean squared error
```

Out[47]:
```
1.1764054243290298
```

In [48]:
```
MSE = mean_squared_error(y_test,y_Pred)
MSE
```

Out[48]:
```
2.8186233324030754
```

In [49]:
```
model.score(x,y)
```

Out[49]:
```
0.8967135509652482
```

In [50]:
```
model.coef_
```

Out[50]:
```
array([ 0.04513717,  0.1952601 , -0.00458532])
```

In [51]:
```
model.intercept_
```

Out[51]:
```
2.9991170116709363
```

16 . Summry

In [52]:
```
import statsmodels.api as sm
x_train_Sm =sm.add_constant(x_train)
x_train_Sm =sm.add_constant(x_train)

ls=sm.OLS(y_train,x_train).fit()
print(ls.summary())
```

```
                            OLS Regression Results
==============================================================================
==========
Dep. Variable:                  Sales   R-squared (uncentered):
0.982
Model:                            OLS   Adj. R-squared (uncentered):
0.982
Method:                 Least Squares   F-statistic:
2931.
Date:                Thu, 06 Apr 2023   Prob (F-statistic):
1.43e-137
Time:                        15:53:35   Log-Likelihood:
-339.06
No. Observations:                 160   AIC:
684.1
Df Residuals:                     157   BIC:
693.4
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
=
                 coef    std err          t      P>|t|      [0.025      0.97
5]
------------------------------------------------------------------------------
-
TV             0.0536      0.001     35.900      0.000       0.051       0.05
7
Radio          0.2278      0.010     21.738      0.000       0.207       0.24
8
Newspaper      0.0125      0.007      1.678      0.095      -0.002       0.02
7
==============================================================================
=
Omnibus:                        9.049   Durbin-Watson:                   2.01
8
Prob(Omnibus):                  0.011   Jarque-Bera (JB):               13.15
8
Skew:                          -0.308   Prob(JB):                      0.0013
9
Kurtosis:                       4.263   Cond. No.                         12.
7
==============================================================================
=

Notes:
[1] R² is computed without centering (uncentered) since the model does not co
ntain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correc
tly specified.
```

In [ ]:

In [ ]:

In [ ]: