Sales Prediction

```
# Importing essential libraries

# for numerical operations
import numpy as np
import pandas as pd
# for Vizualisation
import matplotlib.pyplot as plt
import seaborn as sns
```

Importing and Understanding Dataset

```
# loading dataset
advertise=pd.read_csv('/content/drive/MyDrive/Datasets/CodSoft/advertising.xls')
advertise
```

	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

```
# print no. of rows and columns(order)
advertise.shape
```

```
(200, 4)
```

```
# print total no. of elements(size)
advertise.size
```

```
800
```

```
# printing first five rows
advertise.head()
```

	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

```
# printing last five rows
advertise.tail()
```

	TV	Radio	Newspaper	Sales
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

```
# print column labels
advertise.columns
```

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

```
# print datatype of each column
advertise.dtypes
```

```
TV           float64
Radio        float64
Newspaper    float64
Sales        float64
dtype: object
```

```
# print information about dataframe
advertise.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)
```

```
dtypes: float64(4)
memory usage: 6.4 KB
```

```
# print statistical summary of dataframe
advertise.describe()
```

	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

```
# checking duplicate rows
advertise.duplicated().sum()
```

```
0
```

```
# checking for missing values
advertise.isna().sum()
```

```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

```
# separating i/p and output
x=advertise.drop(['Sales'],axis=1)
y=advertise['Sales']
print("Dimensions \nX :",x.ndim,"\nY :",y.ndim)
```

```
Dimensions
X : 2
Y : 1
```

```
x
```

	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

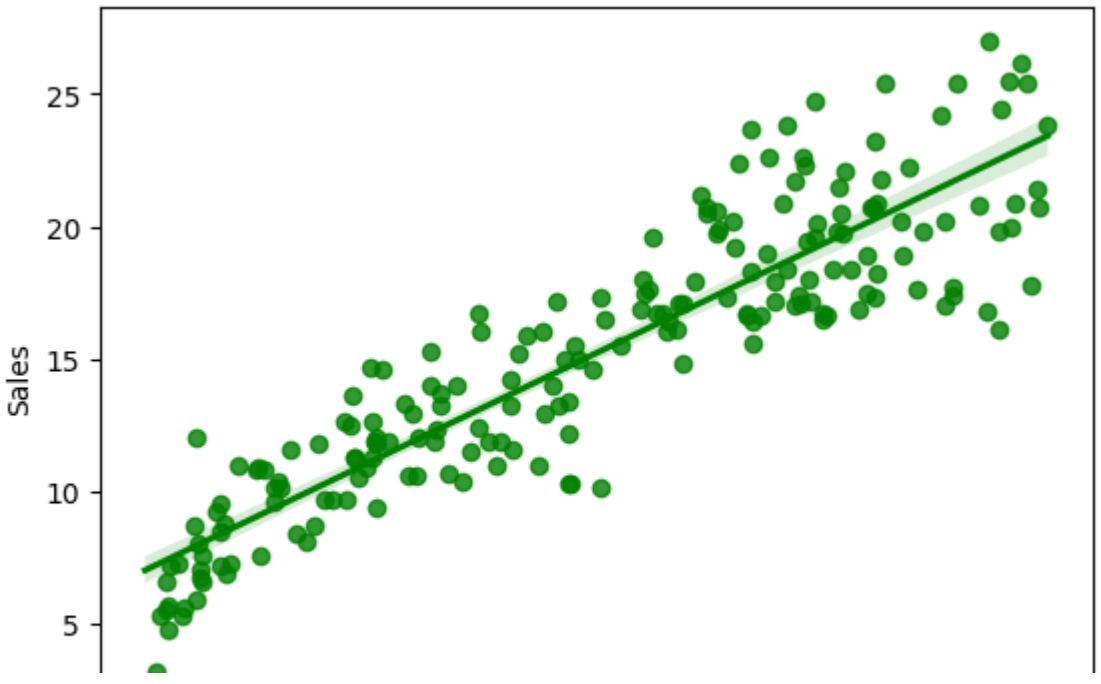
```
y
```

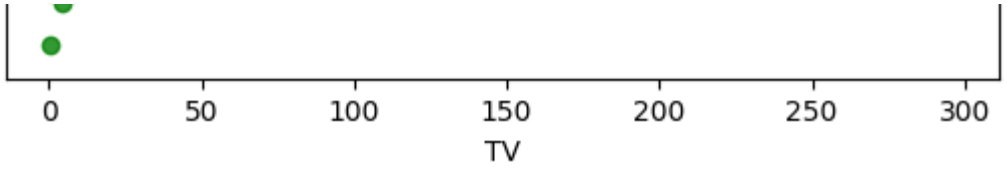
```
0      22.1
1      10.4
2      12.0
3      16.5
4      17.9
       ...
195     7.6
196    14.0
197    14.8
198    25.5
199    18.4
Name: Sales, Length: 200, dtype: float64
```

Data Visualization

```
# Plotting TV against Sales
sns.regplot(x=advertise['TV'],y=advertise['Sales'],color='green')
```

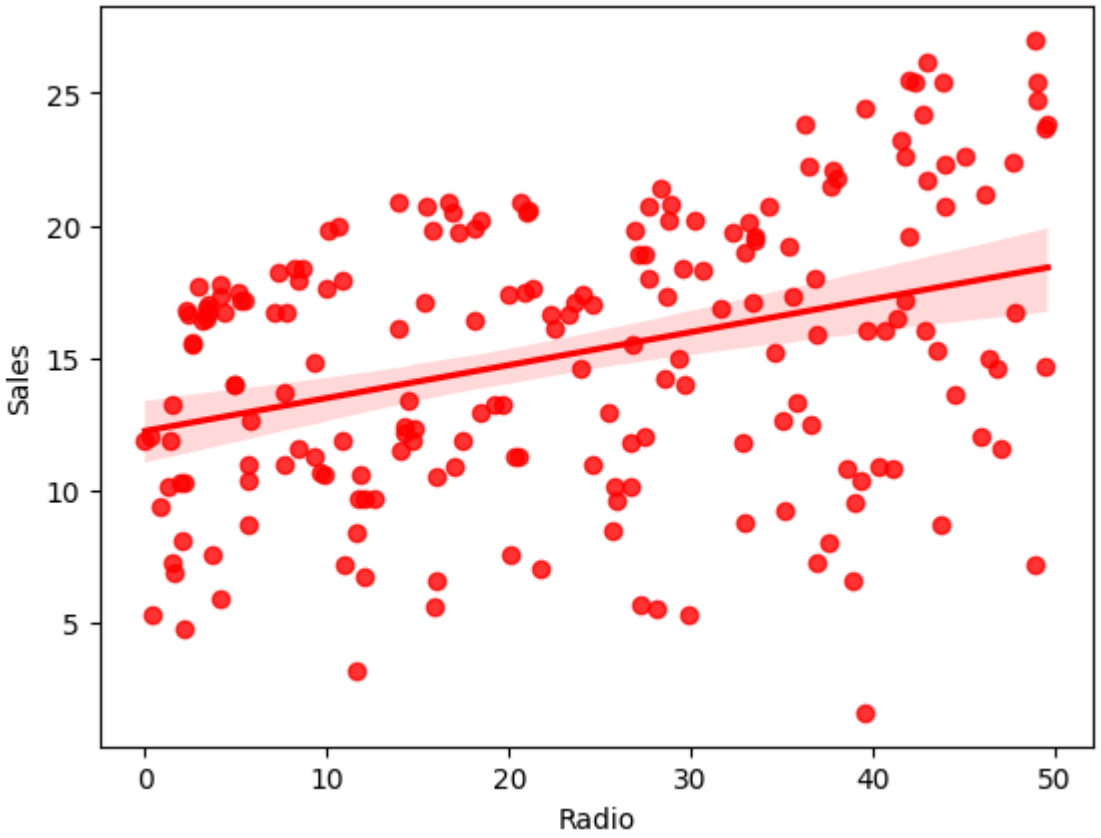
```
<Axes: xlabel='TV', ylabel='Sales'>
```





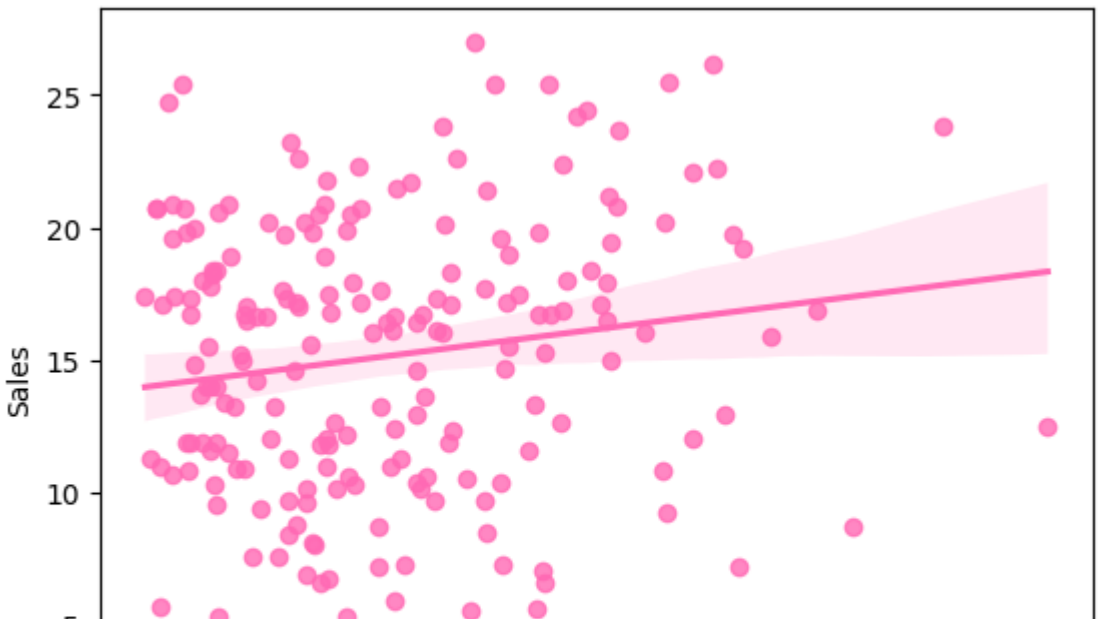

```
# Plotting Radio against Sales
sns.regplot(x=advertise['Radio'],y=advertise['Sales'],color='red')
```

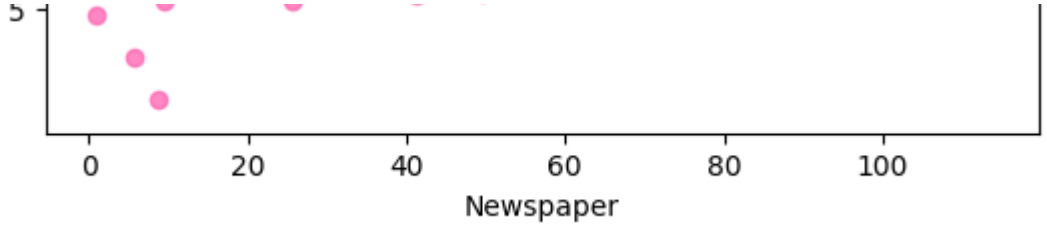
```
<Axes: xlabel='Radio', ylabel='Sales'>
```



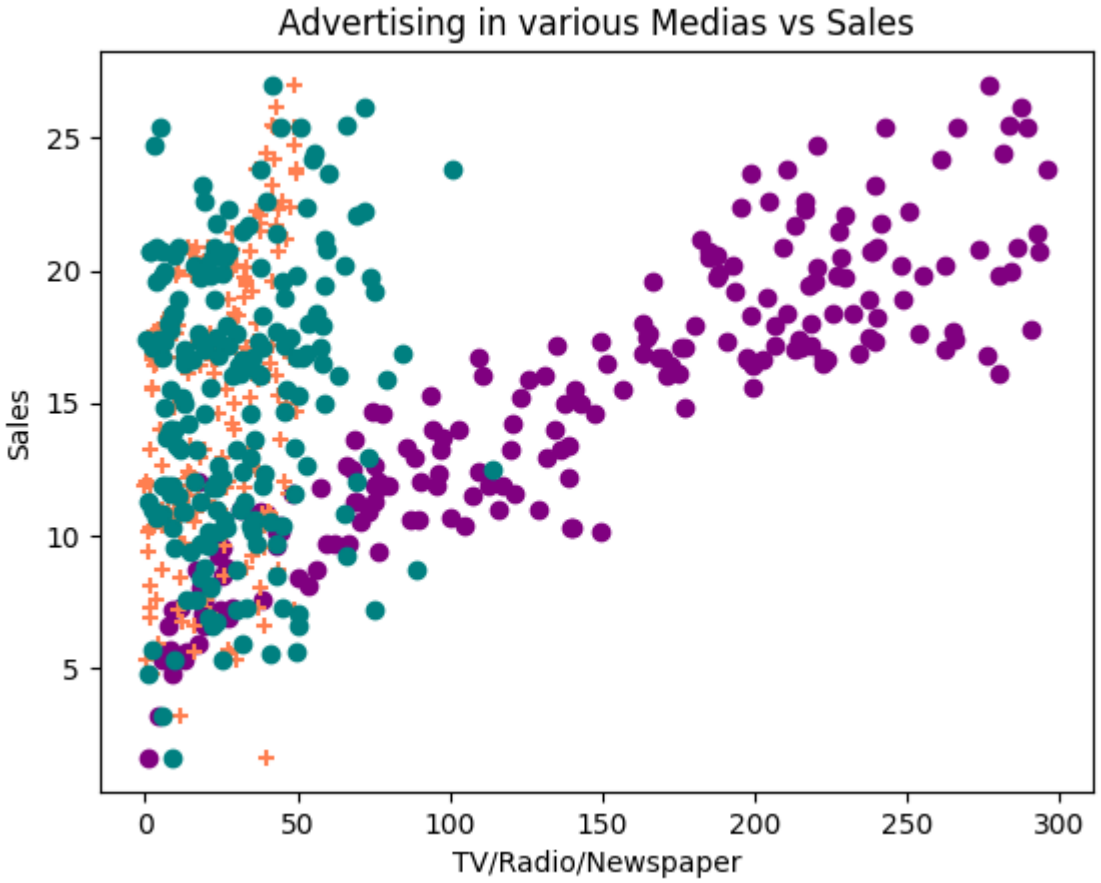

```
# Plotting Newspaper against Sales
sns.regplot(x=advertise['Newspaper'],y=advertise['Sales'],color='hotpink')
```

```
<Axes: xlabel='Newspaper', ylabel='Sales'>
```






```
# Scatter Plot
plt.scatter(advertise['TV'],y,marker='o',color='purple')
plt.scatter(advertise['Radio'],y,marker='+',color='coral')
plt.scatter(advertise['Newspaper'],y,color='teal')
plt.xlabel("TV/Radio/Newspaper")
plt.ylabel("Sales")
plt.title("Advertising in various Medias vs Sales")
plt.show()
```

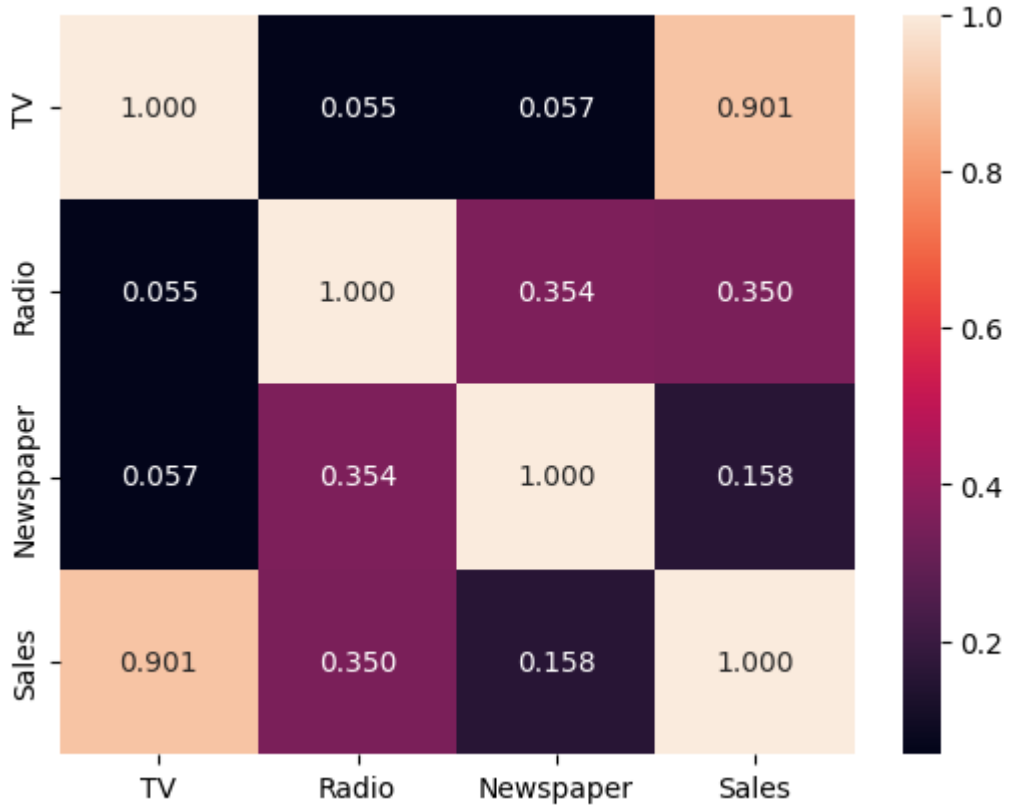



```
# relationship between each column
corr_matrix=advertise.corr()
corr_matrix
```

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.901208
Radio	0.054809	1.000000	0.354104	0.349631
Newspaper	0.056648	0.354104	1.000000	0.157960
Sales	0.901208	0.349631	0.157960	1.000000

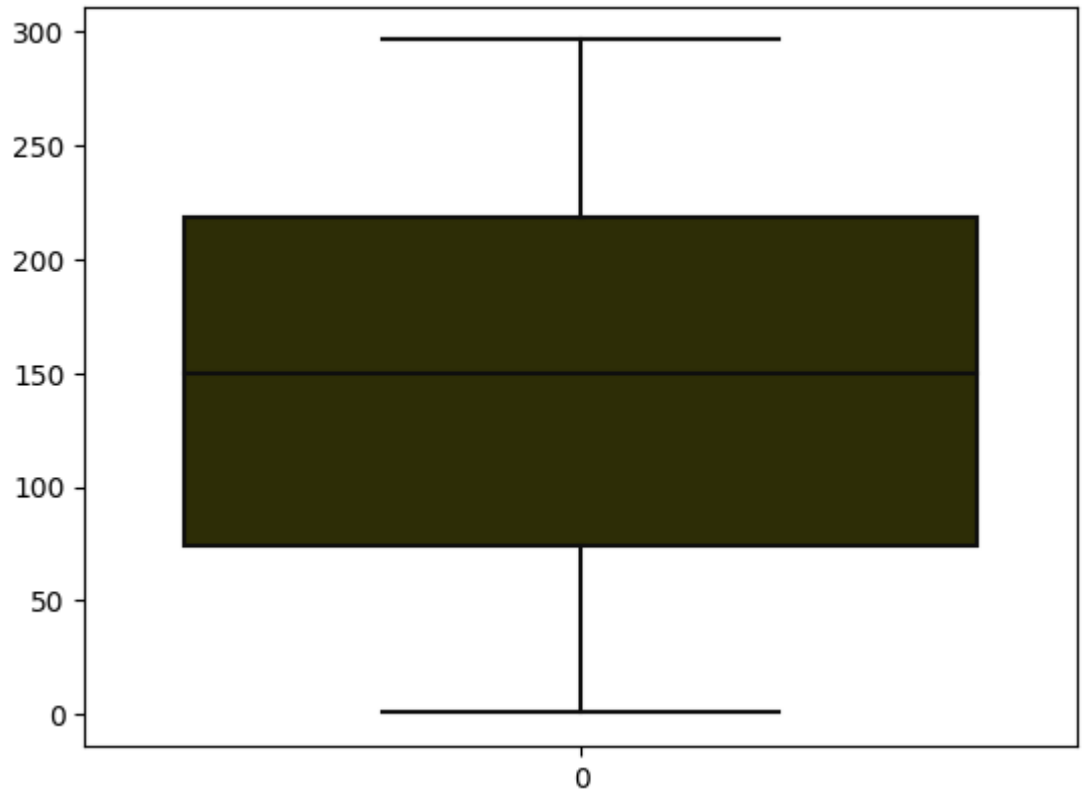
```
# plotting correlation(heatmap)
sns.heatmap(corr_matrix,annot=True,fmt='.3f')
```

```
<Axes: >
```



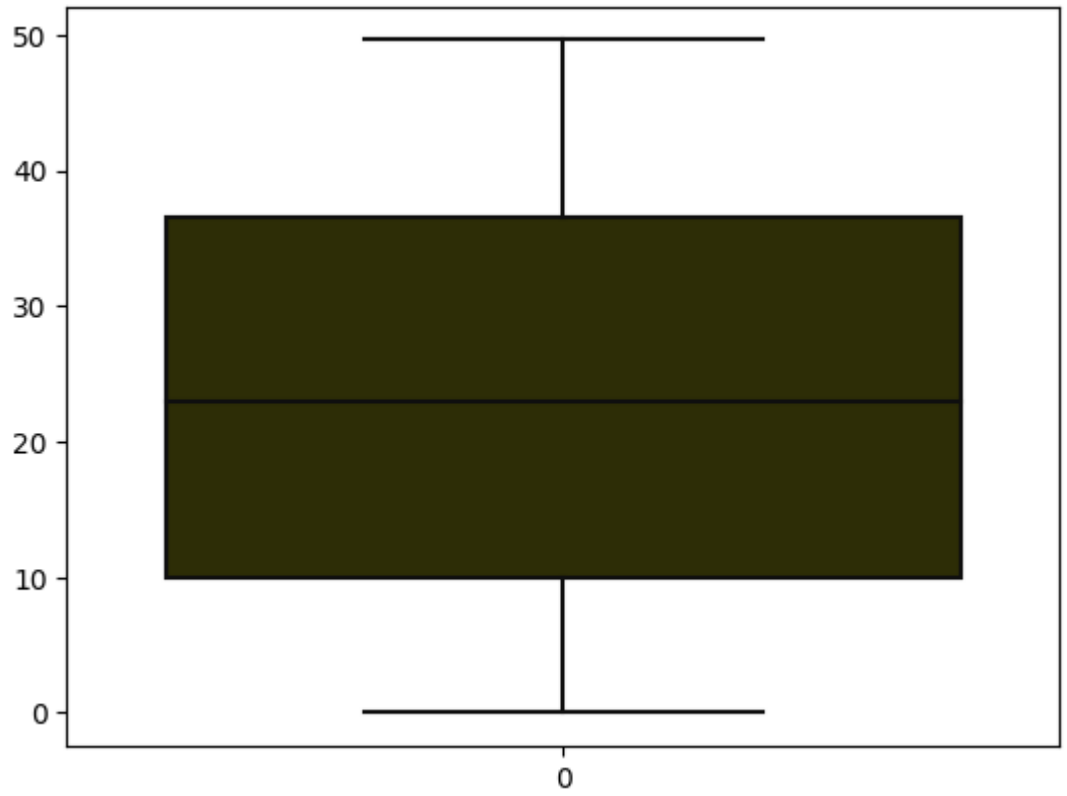

```
# checking for outliers
sns.boxplot(advertise['TV'],color='#333300')
```

```
<Axes: >
```



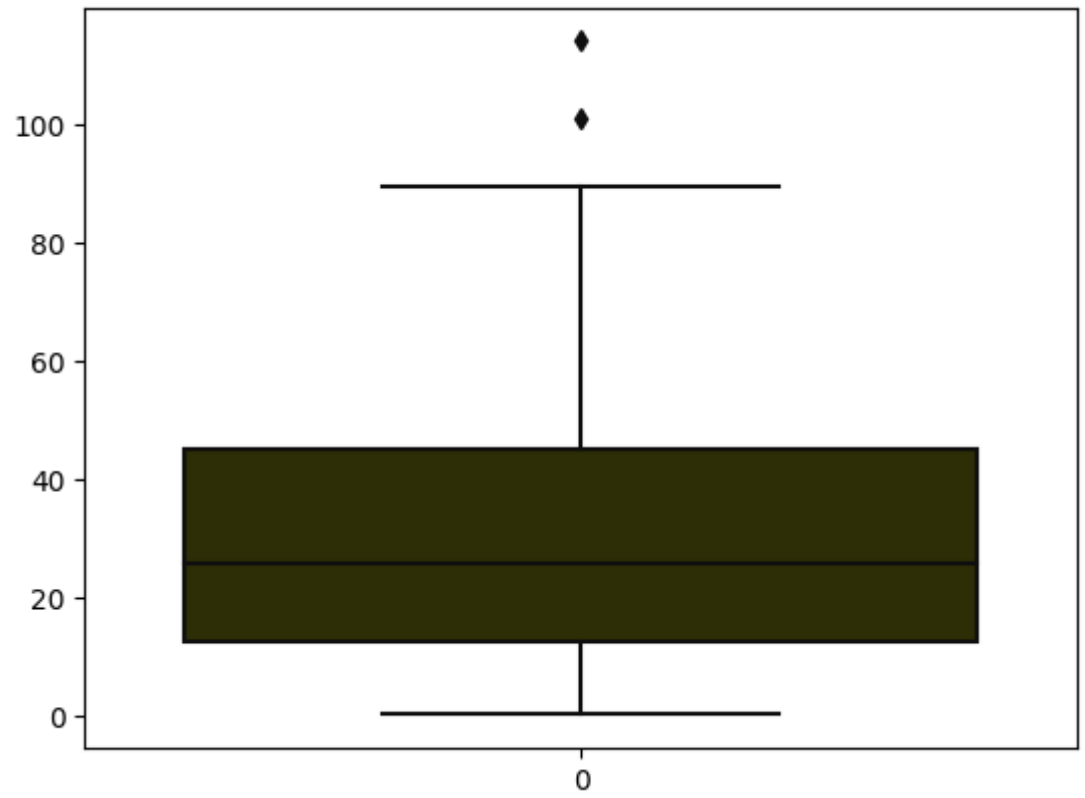
```
sns.boxplot(advertise['Radio'],color='#333300')
```

```
<Axes: >
```



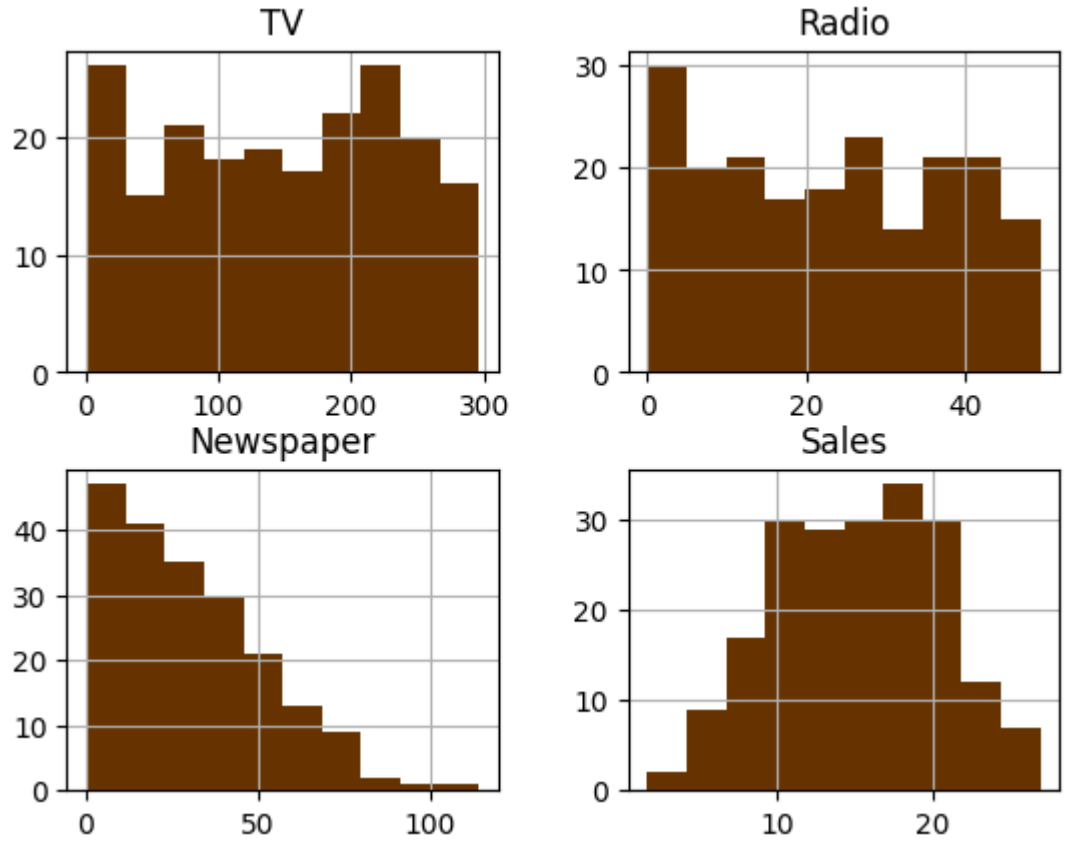

```
sns.boxplot(advertise['Newspaper'],color='#333300')
```

```
<Axes: >
```




```
advertise.hist(color='#663300')
```

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




```
# splitting dataset into training and testing data
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.30,random_state=42)
```

```
x_train
```

	TV	Radio	Newspaper
169	284.3	10.6	6.4
97	184.9	21.0	22.0
31	112.9	17.4	38.6
12	23.8	35.1	65.9
35	290.7	4.1	8.5
...	...	...	...
106	25.0	11.0	29.7
14	204.1	32.9	46.0
92	217.7	33.5	59.0
179	165.6	10.0	17.6
102	280.2	10.1	21.4

140 rows × 3 columns

```
x_test
```

	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
66	31.5	24.6	2.2
182	56.2	5.7	29.7
165	234.5	3.4	84.8
78	5.4	29.9	9.4
186	139.5	2.1	26.6
177	170.2	7.8	35.2
56	7.3	28.1	41.4
152	197.6	23.3	14.2
82	75.3	20.3	32.5
68	237.4	27.5	11.0
124	229.5	32.3	74.2
16	67.8	36.6	114.0
148	38.0	40.3	11.9
93	250.9	36.5	72.3
65	69.0	9.3	0.9
60	53.5	2.0	21.4
84	213.5	43.0	33.8
67	139.3	14.5	10.2
125	87.2	11.8	25.9

132	8.4	27.2	2.1
9	199.8	2.6	21.2
18	69.2	20.5	18.3
55	198.9	49.4	60.0
75	16.9	43.7	89.4
150	280.7	13.9	37.0
104	238.2	34.3	5.3
135	48.3	47.0	8.5
137	273.7	28.9	59.7
164	117.2	14.7	5.4
76	27.5	1.6	20.7
79	116.0	7.7	23.1
197	177.0	9.3	6.4
38	43.1	26.7	35.1
24	62.3	12.6	18.3
122	224.0	2.4	15.6
195	38.2	3.7	13.8
29	70.6	16.0	40.8
19	147.3	23.9	19.1
143	104.6	5.7	34.4
86	76.3	27.5	16.0
114	78.2	46.8	34.5
173	168.4	7.1	12.8
5	8.7	48.9	75.0
126	7.8	38.9	50.6
117	76.4	0.8	14.8
73	129.4	5.7	31.3
140	73.4	17.0	12.9
98	289.7	42.3	51.2
172	19.6	20.1	17.0
96	197.6	3.5	5.9

```
y_train
```

```
169    20.0
97     20.5
31     11.9
12      9.2
35     17.8
       ...
106     7.2
14     19.0
92     19.4
179    17.6
102    19.8
Name: Sales, Length: 140, dtype: float64
```

y_test

```
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    16.5
45     16.1
66     11.0
182     8.7
165    16.9
78      5.3
186    10.3
177    16.7
56      5.5
152    16.6
82     11.3
68     18.9
124    19.7
16     12.5
148    10.9
93     22.2
65     11.3
60      8.1
84     21.7
67     13.4
125    10.6
132     5.7
9      15.6
18     11.3
55     23.7
75      8.7
150    16.1
104    20.7
135    11.6
137    20.8
164    11.9
76      6.9
79     11.0
197    14.8
38     10.1
24      9.7
```

```
122    16.6
195     7.6
29     10.5
19     14.6
143    10.4
86     12.0
114    14.6
173    16.7
5       7.2
126     6.6
117     9.4
73     11.0
140    10.9
98     25.4
```

Model Creation

1. Multiple Linear Regression

```
from sklearn.linear_model import LinearRegression
# create linear regression object
lr=LinearRegression()
# train the model using training set
lr.fit(x_train,y_train)
# predicting output(sales) using testing set
y_pred_lr=lr.predict(x_test)
y_pred_lr
```

```
array([17.15991908, 20.53369503, 23.68914396,  9.5191455 , 21.60736836,
       12.78101318, 21.08636345,  8.76054246, 17.11499951, 16.68789636,
        8.97584663,  8.57645026, 18.33212325,  8.17863567, 12.64605571,
       14.94486946,  8.34939536, 17.83858948, 11.12172174, 20.37740648,
       20.9483297 , 13.04035779, 11.01360656, 22.51142595,  9.40369784,
        7.98591291, 20.86943368, 13.77882255, 10.83407064,  8.00419229,
       15.88597618, 10.7027424 , 20.9521718 , 10.84679243, 21.50720813,
       21.07347295, 12.22673775, 22.85273767, 12.57698182,  6.54597206,
       11.93411853, 15.23490068, 10.07411153,  9.52159696, 17.11786382,
        7.28032677, 10.49404864, 15.24356754, 11.20742176, 11.78392665,
       14.01472163, 14.59884572, 10.82722434,  9.55839415,  9.03749681,
       12.51183313, 10.52551021, 25.01900824,  7.99334943, 15.73916263])
```

```
# actual output(sales)
y_test
```

```
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    16.5
45     16.1
```

```
66     11.0
182     8.7
165    16.9
78      5.3
186    10.3
177    16.7
56      5.5
152    16.6
82     11.3
68     18.9
124    19.7
16     12.5
148    10.9
93     22.2
65     11.3
60      8.1
84     21.7
67     13.4
125    10.6
132     5.7
9      15.6
18     11.3
55     23.7
75      8.7
150    16.1
104    20.7
135    11.6
137    20.8
164    11.9
76      6.9
79     11.0
197    14.8
38     10.1
24      9.7
122    16.6
195     7.6
29     10.5
19     14.6
143    10.4
86     12.0
114    14.6
173    16.7
5       7.2
126     6.6
117     9.4
73     11.0
140    10.9
98     25.4
```

```
# Performance Evaluation
from sklearn.metrics import mean_absolute_percentage_error,mean_squared_error,r2_score
print("Mean Absolute Percentage Error :",mean_absolute_percentage_error(y_test,y_pred_
print("Mean Squared Error :",mean_squared_error(y_test,y_pred_lr))
print("R2 Score :",r2_score(y_test,y_pred_lr))
```

```
Mean Absolute Percentage Error : 0.10536440823029307
Mean Squared Error : 2.541624036229147
R2 Score : 0.9091484341849799
```

2. Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
# create decision tree regression object
dtr=DecisionTreeRegressor()
dtr.fit(x_train,y_train)
y_pred_dtr=dtr.predict(x_test)
y_pred_dtr
```

```
array([18. , 23.8, 19.6,  5.6, 23.8, 15.3, 22.6,  9.7, 17. , 17.1,  8.8,
        9.7, 16.7,  1.6, 13.2, 17.9,  4.8, 17. , 11.9, 20.9, 19.8, 15.3,
       10.8, 22.1,  9.7,  9.7, 22.6, 13.2, 11.9,  4.8, 16.4, 13.2, 23.8,
       12. , 20.1, 20.9, 10.4, 19.8, 13.2,  6.6, 13.2, 17.6,  9.6,  9.7,
       17. ,  9.7, 12.3, 10.1, 13.2, 13.3, 13.6, 17.6,  4.8,  4.8, 11.9,
       13.2, 13.2, 25.5,  6.6, 16.4])
```

```
# Performance Evaluation
print("Mean Absolute Percentage Error :",mean_absolute_percentage_error(y_test,y_pred_
print("Mean Squared Error :",mean_squared_error(y_test,y_pred_dtr))
print("R2 Score :",r2_score(y_test,y_pred_dtr))
```

```
Mean Absolute Percentage Error : 0.13139753228670886
Mean Squared Error : 3.090333333333333
R2 Score : 0.8895345581322749
```

3. Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor
# create random forest regression object
rfr=RandomForestRegressor()
rfr.fit(x_train,y_train)
y_pred_rfr=rfr.predict(x_test)
y_pred_rfr
```

```
array([17.504, 22.622, 19.844,  6.458, 23.075, 14.001, 22.671,  9.47 ,
       17.188, 17.001,  8.525, 10.614, 17.489,  4.627, 11.975, 17.178,
        5.949, 17.783, 12.18 , 19.804, 19.814, 13.501, 10.675, 21.881,
       10.928, 10.368, 22.784, 12.428, 11.84 ,  5.508, 16.697, 11.572,
       23.112,  9.859, 19.786, 20.15 , 11.017, 19.591, 12.601,  7.419,
       12.45 , 17.428, 10.013, 10.249, 17.024,  9.482, 11.193, 13.679,
       12.487, 13.062, 13.962, 17.527,  6.823,  6.084, 11.985, 12.613,
       12.167, 25.274,  7.051, 17.027])
```

```
# Performance Evaluation
print("Mean Absolute Percentage Error :",mean_absolute_percentage_error(y_test,y_pred_
print("Mean Squared Error :",mean_squared_error(y_test,y_pred_rfr))
print("R2 Score :",r2_score(y_test,y_pred_rfr))
```

```
Mean Absolute Percentage Error : 0.08518143591305863
Mean Squared Error : 1.4959507166666661
R2 Score : 0.9465265267191489
```