Importing the library

In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

Importing the Dataset

In [3]:

```
advertising = pd.read_csv(r"C:\Users\91956\Downloads\advertising (1).csv")
advertising
```

Out[3]:

	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

EDA

Shape

In [4]:

```
advertising.shape
```

Out[4]:

```
(200, 4)
```

Columns and their types

In [5]:

```
advertising.columns
```

Out[5]:

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

In [6]:

```
advertising.dtypes
```

Out[6]:

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

In [5]:

```
advertising.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
```

Null value analysis

In [7]:

```
advertising.isna().sum()
```

Out[7]:

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

Duplicate values

In [8]:

```
advertising.duplicated().sum()
```

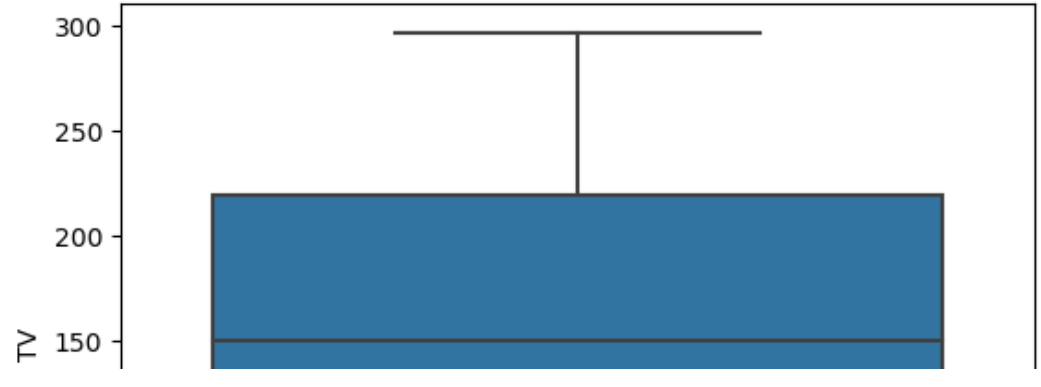
Out[8]:

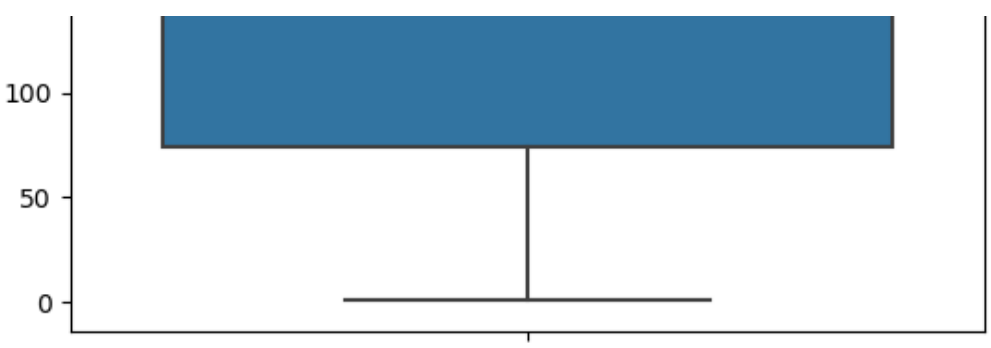
```
0
```

Outlier's detection

In [9]:

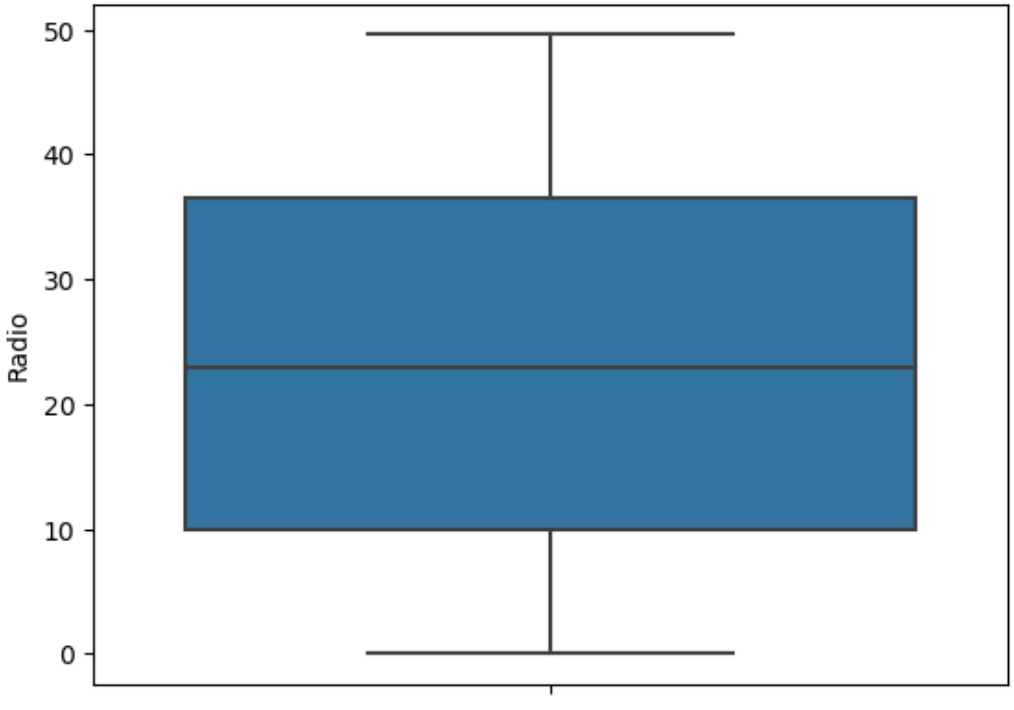
```
sns.boxplot(y='TV',data=advertising)
plt.show()
```





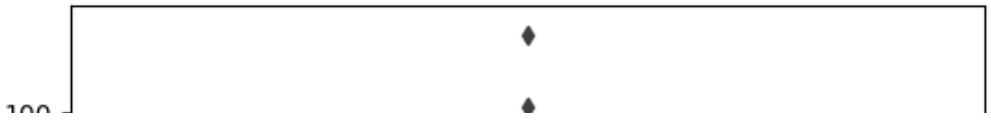
In [10]:

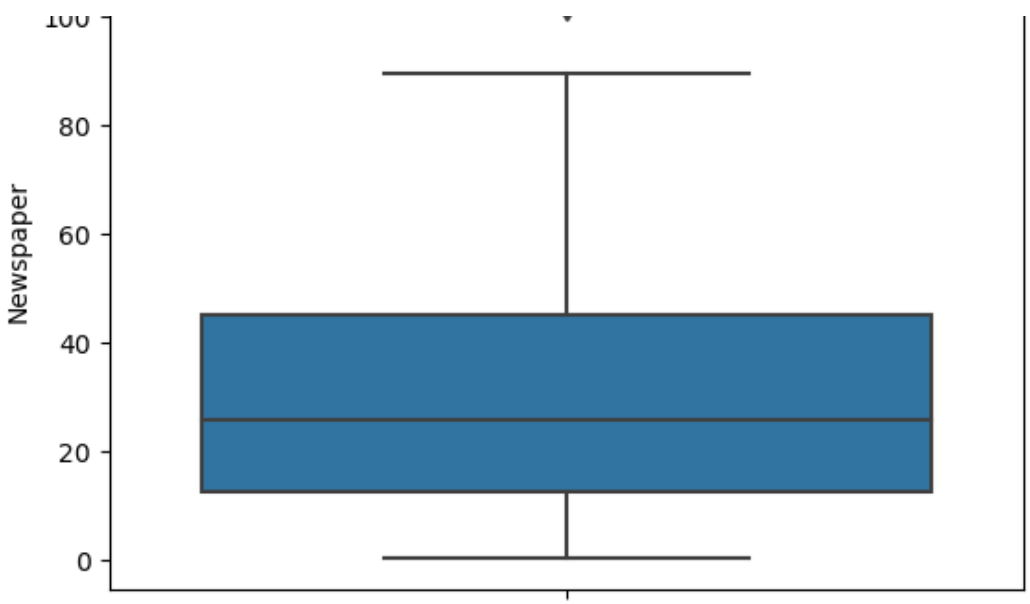
```
sns.boxplot(y='Radio',data=advertising)
plt.show()
```



In [11]:

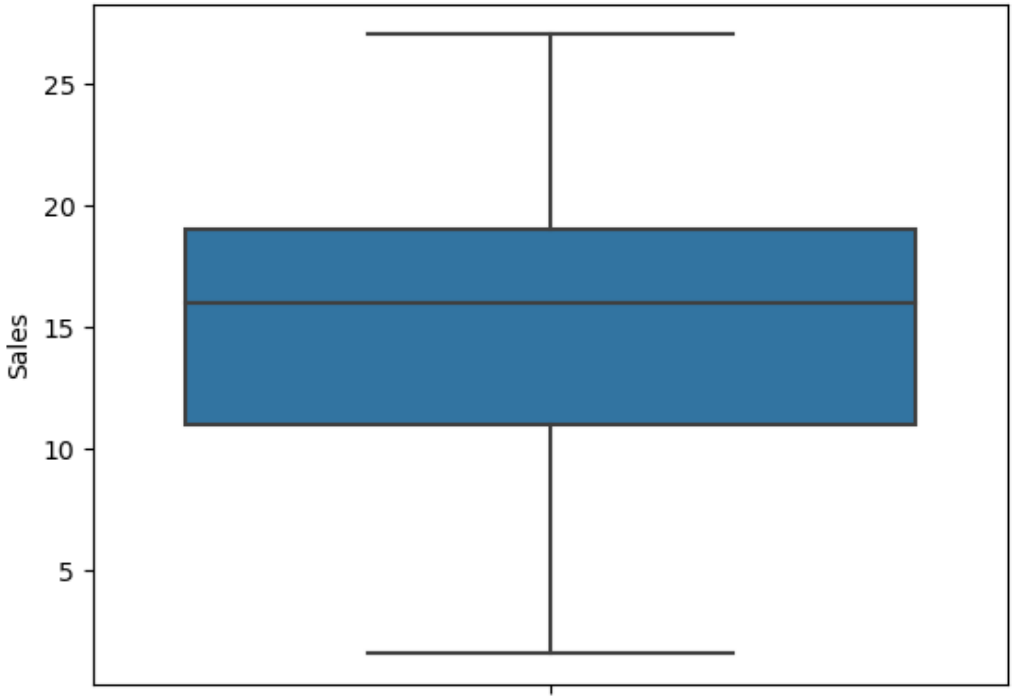
```
sns.boxplot(y='Newspaper',data=advertising)
plt.show()
```





In [12]:

```
sns.boxplot(y='Sales',data=advertising)
plt.show()
```



Statistical summary

In [6]:

```
advertising.describe()
```

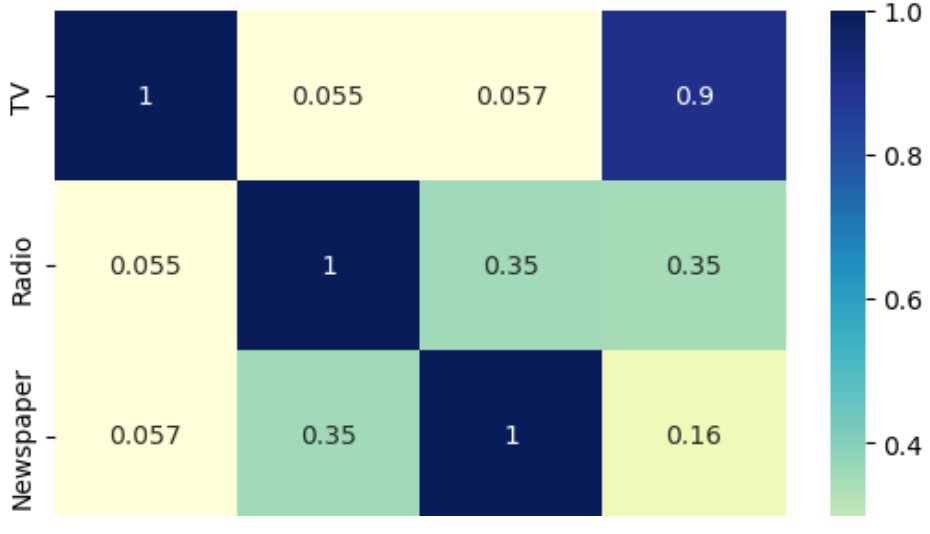
Out[6]:

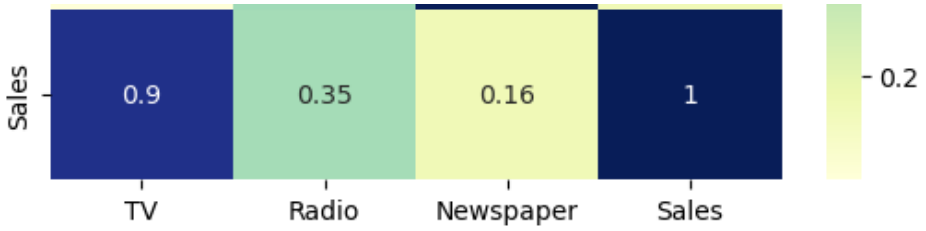
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000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

Correlation

In [7]:

```
sns.heatmap(advertising.corr(), cmap = "YlGnBu", annot = True)
plt.show()
```





Interpretation

In [ ]:

```
#TV and sales are highly correlated means that company should increase the advertising on TV(COMPANY COST) so that it will boost t
he sales of the company.
```

Machine learning

In [9]:

```
x=advertising[['TV', 'Radio', 'Newspaper']]
y=advertising['Sales']
```

Splitting the dataset into training and testing

In [11]:

```
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=7)
```

Importing the algorithm and the performance measure

In [10]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score,mean_squared_error
from math import sqrt
```

In [16]:

```
lm=LinearRegression()
lm.fit(x_train,y_train)
y_pred_test=lm.predict(x_test)
```

```
q=r2_score(y_test,y_pred_test)
rmse=sqrt(mean_squared_error(y_test,y_pred_test))
print('Value of r2 is', {q})
print('Value of rmse is ',{rmse} )
```

```
Value of r2 is {0.9259262684743649}
Value of rmse is  {1.4733877434160483}
```

In [17]:

```
from sklearn.neighbors import KNeighborsRegressor
knn=KNeighborsRegressor(n_neighbors=15)

knn.fit(x_train,y_train)
y_pred_knn=knn.predict(x_test)
q=r2_score(y_test,y_pred_knn)
rmse=sqrt(mean_squared_error(y_test,y_pred_knn))

print('Value of r2 is', {q})
print('Value of rmse is ',{rmse} )
```

```
Value of r2 is {0.8914010769553321}
Value of rmse is  {1.7840117089800103}
```

In [18]:

```
from sklearn.tree import DecisionTreeRegressor
dt=DecisionTreeRegressor(max_depth=5,random_state=7,min_samples_split=15)
dt.fit(x_train,y_train)
y_pred_dt=dt.predict(x_test)
q=r2_score(y_test,y_pred_dt)
rmse=sqrt(mean_squared_error(y_test,y_pred_dt))

print('Value of r2 is', {q})
print('Value of rmse is ',{rmse} )
```

```
Value of r2 is {0.8813068695481108}
Value of rmse is  {1.8650811523471227}
```

In [19]:

```
from sklearn.svm import SVR

svr=SVR(kernel='rbf')
svr.fit(x_train,y_train)
y_pred_svr=svr.predict(x_test)
q=r2_score(y_test,y_pred_svr)
rmse=sqrt(mean_squared_error(y_test,y_pred_svr))

print('Value of r2 is', {q})
```

```
print('Value of rmse is ',{rmse} )
```

```
Value of r2 is {0.8667119579014229}
Value of rmse is  {1.9764258956456702}
```