Sales Prediction

Using linear regression,decision tree and random forest

Problem Statement

Build a model which predicts sales based on the money spent on different platforms for marketing.

Data

In this notebook, we'll build a different machine learning techniques model to predict `Sales` using an appropriate predictor variable.

Reading and Understanding the Data

In [1]:
```
# Import the numpy and pandas package

import numpy as np
import pandas as pd

# Data Visualisation
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
df = pd.read_csv(r"C:\Users\kanis\Downloads\advertising.csv")
df.head()
```

Out[2]:

	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

The dataset comprises information regarding advertising expenditures across various media channels such as TV, radio, and newspaper, along with corresponding sales figures derived from these marketing efforts. The data provides insights into the relationships between advertising investments in each channel (TV, radio, and newspaper) and the resulting sales outcomes.

Data Inspection

In [3]:
```
df.shape
```

Out[3]:
```
(200, 4)
```

In [4]:
```
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
```

In [5]:
```
df.describe()
```

Out[5]:

	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

Data Cleaning

In [6]:
```
# Checking Null values
df.isnull().sum()
# There are no NULL values in the dataset, hence it is clean.
```

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

In [7]:
```
df.duplicated().sum()#checking duplicate values
```

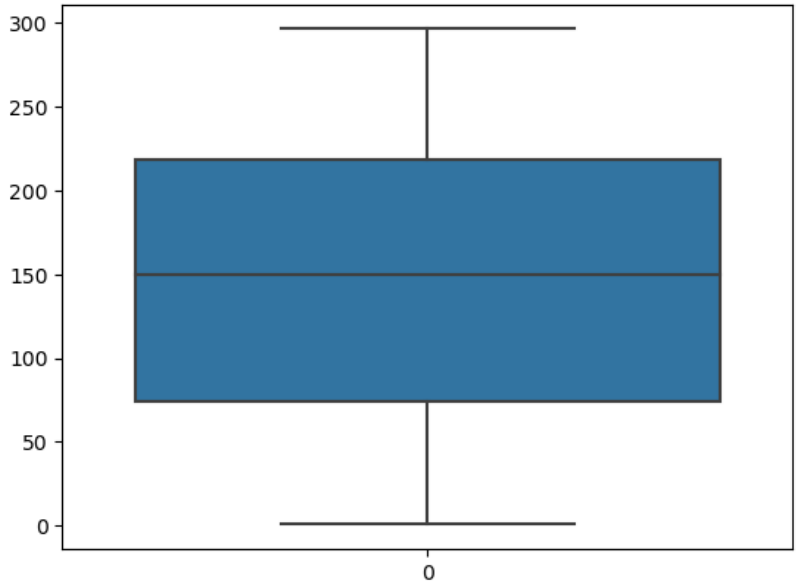
Out[7]:
```
0
```

no dublicate if there were any duplicate we would have simply drop the duplicate

EDA(Explotatory data analysis)

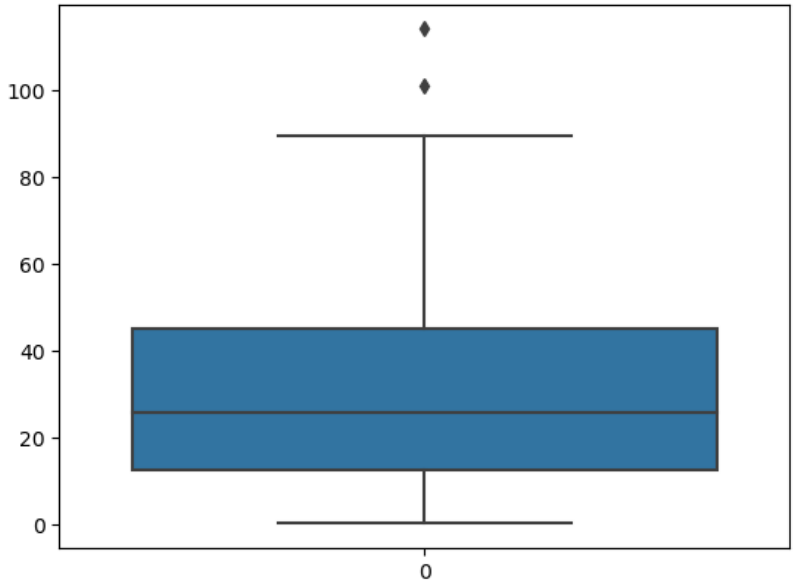
In [8]:
```
sns.boxplot(df['TV'])
```

Out[8]:
```
<Axes: >
```



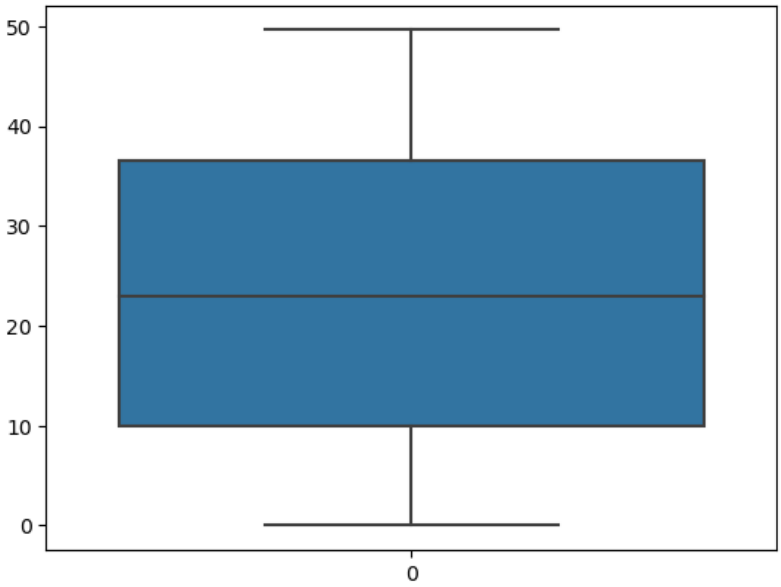
In [9]:
```
sns.boxplot(df['Newspaper'])
```

Out[9]: <Axes: >



In [10]:
```
sns.boxplot(df['Radio'])
```

Out[10]: <Axes: >

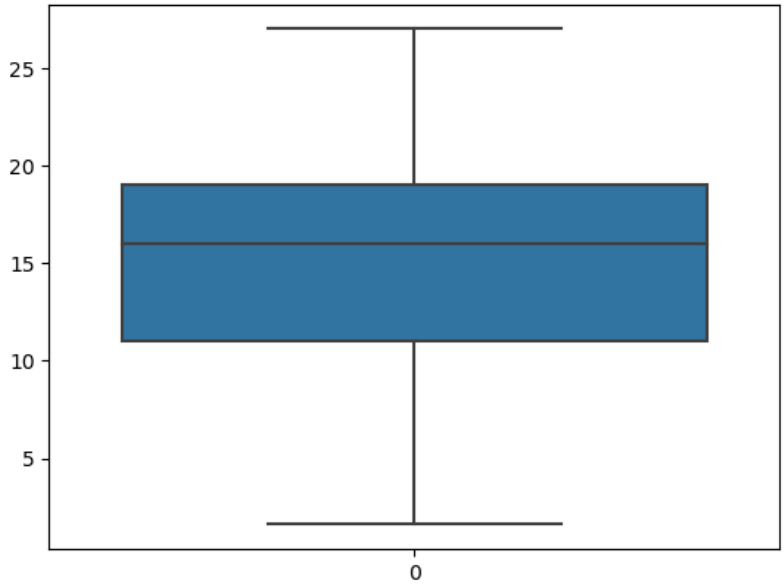


In [38]:
```
# There are  also no considerable outliers present in the data.
```

Exploratory Data Analysis

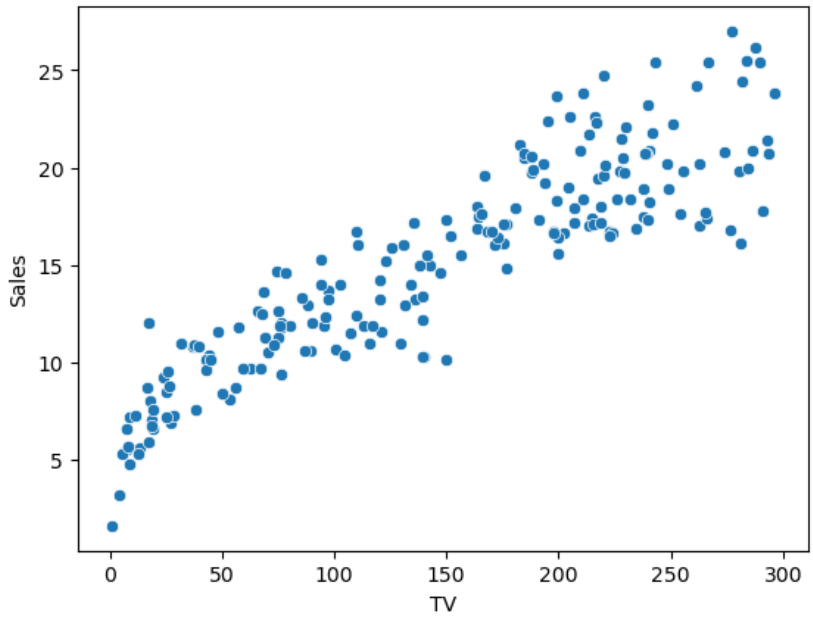
Sales (Target Variable)

In [12]:
```
sns.boxplot(df['Sales'])
plt.show()
```



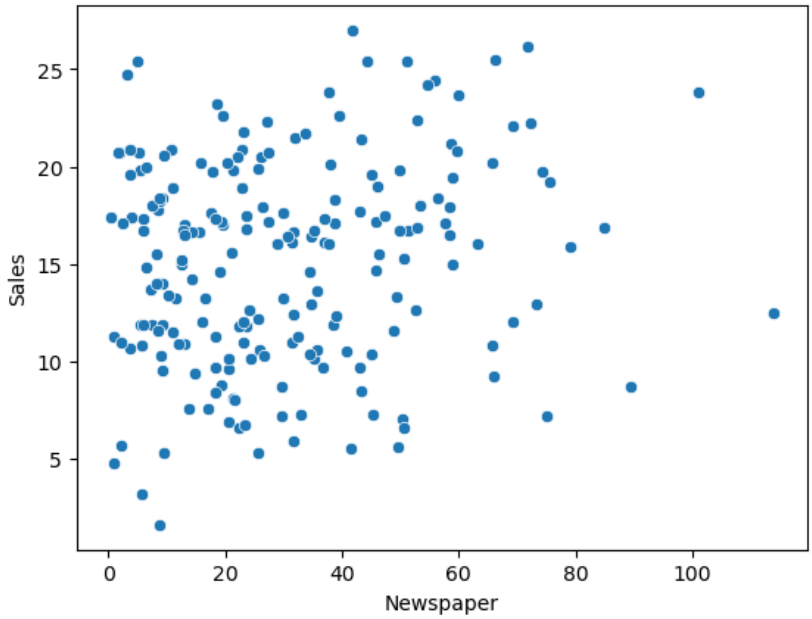
In [13]:
```
sns.scatterplot(x=df['TV'],y=df['Sales'])
```

Out[13]:
```
<Axes: xlabel='TV', ylabel='Sales'>
```



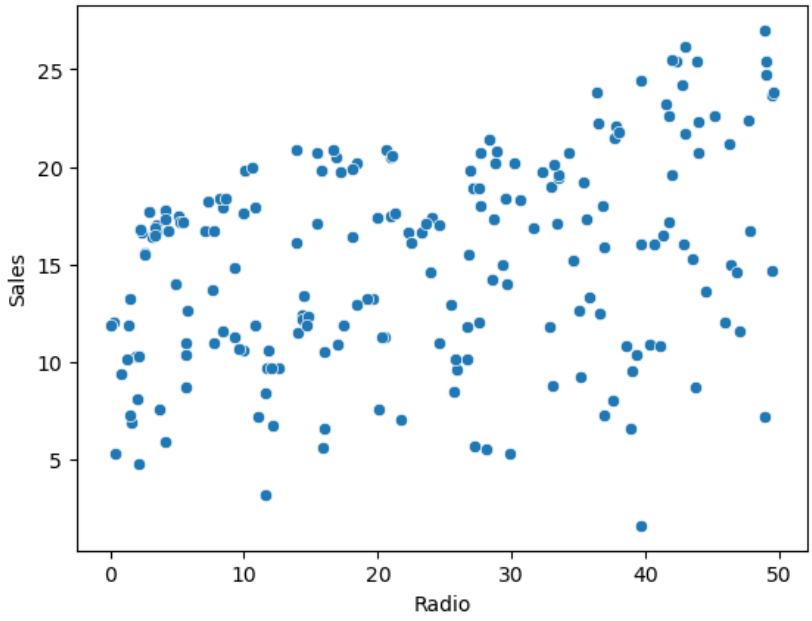
In [14]:
```
sns.scatterplot(x=df['Newspaper'],y=df['Sales'])
```

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

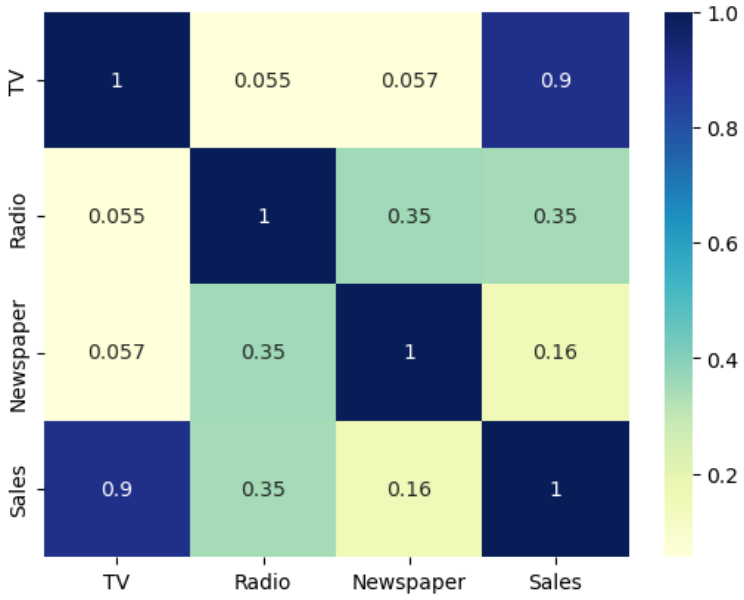


In [15]:
```
sns.scatterplot(x=df['Radio'],y=df['Sales'])
```

Out[15]:
```
<Axes: xlabel='Radio', ylabel='Sales'>
```



In [16]:
```
# Let's see the correlation between different variables.
sns.heatmap(df.corr(), cmap="YlGnBu", annot = True)
plt.show()
```



Model Building

Performing Simple Linear Regression

In [17]:
```
x =df.drop(['Sales'],axis=1) #dropping highly co-related features.
x #new datas set.
```

Out[17]:

	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 [18]:
```
y=df['Sales']
y
```

Out[18]:
```
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
```

Train-Test Split

You now need to split our variable into training and testing sets. You'll perform this by importing `train_test_split` from the `sklearn.model_selection` library. It is usually a good practice to keep 70% of the data in your train dataset and the rest 30% in your

In [19]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, train_size = 0.7, test_size = 0.3, random_state = 100
```

In [20]:
```
# Let's now take a look at the train dataset

x_train.head()
```

Out[20]:

	TV	Radio	Newspaper
74	213.4	24.6	13.1
3	151.5	41.3	58.5
185	205.0	45.1	19.6
26	142.9	29.3	12.6
90	134.3	4.9	9.3

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

Out[21]:
```
74     17.0
3      16.5
185    22.6
26     15.0
90     14.0
Name: Sales, dtype: float64
```

In [22]:
```
# Importing linear regression library
from sklearn.linear_model import LinearRegression
```

In [23]:
```
# loading linearregression model
reg_model=LinearRegression()
```

In [24]:
```
reg_model.fit(x_train,y_train)
```

Out[24]:
```
LinearRegression()
```

In [25]:
```
#Model evaluation for training set
y_predict = reg_model.predict(x_train)
```

In [26]:
```
#Importing metrices from sklearn library
from sklearn import metrics
#Metrix gives r2 score to cross verify accuracy of model
```

In [27]:
```
reg_model.score(x_train, y_train)
```

Out[27]:
```
0.91049938193816
```

In [28]:
```
#cheking the r2_score to y_test,y_predict to predict the accuracy.
r_square=metrics.r2_score(y_train,y_predict)
print('R-square error for decision tree regressor is:',round(100*(r_square),3))
```

```
R-square error for decision tree regressor is: 91.05
```

Decission tree

In [29]:
```
#Importing DecisionTreeRegressor from sklearn.tree library.
from sklearn.tree import DecisionTreeRegressor
#Calling DecisionTreeRegressor with max_depth as 3 and calling it to dt.
dt= DecisionTreeRegressor(max_depth=3)
```

In [30]:
```
#Fitting x_train,y_train to dt.
dt.fit(x_train,y_train)
```

Out[30]:
```
DecisionTreeRegressor(max_depth=3)
```

In [31]:
```
# Score of train model
print(dt.score(x_train, y_train))
```

```
0.9237579359310554
```

In [32]:
```
#Model evaluation for training set
yd_predict = dt.predict(x_train)
```

In [33]:
```
#cheking the r2_score to y_test,y_predict to predict the accuracy.
r_square=metrics.r2_score(y_train,yd_predict)
print('R-square error for decision tree regressor is:',round(100*(r_square),3))
```

```
R-square error for decision tree regressor is: 92.376
```

Random Forest

In [34]:
```
#importing Random forest regressor from sklearn.ensemble and it to rfr with random_state 5 and fitting it to X_t
from sklearn.ensemble import RandomForestRegressor
#importing metrices from sklearn library
from sklearn import metrics
rfr =RandomForestRegressor(random_state=5)
rfr.fit(x_train,y_train)
```

Out[34]:
```
RandomForestRegressor
RandomForestRegressor(random_state=5)
```

In [35]:
```
# Score of train model
print(rfr.score(x_train, y_train))
```

```
0.9916386982783094
```

In [36]:
```
#by using rfr. predict function x)test and calling it to y_predict to predict a value
y1_pred =rfr.predict(x_train)
```

In [37]:
```
#cheking the r2_score to y_test,y_predict to predict the accuracy.
r_square=metrics.r2_score(y_train,y1_pred)
print('R-square error for decision tree regressor is:',round(100*(r_square),3))
```

```
R-square error for decision tree regressor is: 99.164
```

In [ ]:
```
```