TASK 4: SALES PREDICTION

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Importing required libraries

We are importing a lot of various required libraries to perform the specific tasks like numpy to deal with arrays or calculations , pandas for data frame or data structures etc. Following libraries are imported in the program:

- Numpy
- Pandas
- Warnings
- Matplotlib
- Sklearn
- Seaborn etc.

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

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


from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import mean_squared_error
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

Reading Dataset

```
Data = pd.read_csv("/content/drive/MyDrive/Dataset/advertising.csv")
Data.head(4)
```

	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

Data Pre-processing

It refers to the cleaning, transforming, and integrating of data in order to make it ready for analysis. The goal of data preprocessing is to improve the quality of the data and to make it more suitable for the specific data mining task.

```
Data.shape
```

```
(200, 4)
```

```
Data.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
```

```
Data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
Radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
Newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
Sales	200.0	15.1305	5.283892	1.6	11.000	16.00	19.050	27.0

Checking for null values and if any null is present then we will deal with missing values.

```
# Checking Null values
Data.isnull().sum()
```

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

```
Data.dtypes
```

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

```
Data.describe().T
```

	count	mean	std	min	25%	50%	75%	max
TV	200.0	147.0425	85.854236	0.7	74.375	149.75	218.825	296.4
Radio	200.0	23.2640	14.846809	0.0	9.975	22.90	36.525	49.6
Newspaper	200.0	30.5540	21.778621	0.3	12.750	25.75	45.100	114.0
Sales	200.0	15.1305	5.283892	1.6	11.000	16.00	19.050	27.0

```
def var_summary(x):
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),  x.std(), x.var(), x.min(), x.quantile(0.01), x.quantil
                  index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P1' , 'P5' ,'P10' ,'P25/Q1' ,'P50/Q2' ,'P75/Q3' ,'P9
```

```
def var_summary(x):
    uc = x.mean()+(2*x.std())
    lc = x.mean()-(2*x.std())

    for i in x:
        if i<lc or i>uc:
            count = 1
        else:
            count = 0
    outlier_flag = count
    return pd.Series([x.count(), x.isnull().sum(), x.sum(), x.mean(), x.median(),  x.std(), x.var(), x.min(), x.quantile(0.01), x.quantil
                  index=['N', 'NMISS', 'SUM', 'MEAN','MEDIAN', 'STD', 'VAR', 'MIN', 'P1' , 'P5' ,'P10' ,'P25' ,'P50' ,'P75' ,'P90' ,'P95'
```

```
#var_summary(advt.TV)
var_summary(Data.Newspaper)
```

```
N               200.000000
NMISS             0.000000
SUM            6110.800000
MEAN             30.554000
MEDIAN           25.750000
STD              21.778621
VAR             474.308326
MIN               0.300000
P1                0.999000
```

```
P5                3.600000
P10               5.990000
P25              12.750000
P50              25.750000
P75              45.100000
P90              59.070000
P95              71.825000
P99              89.515000
MAX             114.000000
LC              -13.003242
UC               74.111242
outlier_flag      0.000000
dtype: float64
```

```
#### Data Cleaning starts from here After my Extensive EDA
Data["Newspaper"] = Data.Newspaper.clip(lower = 10  , upper = 100)
var_summary(Data.Newspaper)
```

```
N               200.000000
NMISS             0.000000
SUM            6267.100000
MEAN             31.335500
MEDIAN           25.750000
STD              20.537900
VAR             421.805316
MIN              10.000000
P1               10.000000
P5               10.000000
P10              10.000000
P25              12.750000
P50              25.750000
P75              45.100000
P90              59.070000
P95              71.825000
P99              89.506000
MAX             100.000000
LC               -9.740299
UC               72.411299
outlier_flag      0.000000
dtype: float64
```

```
# Handling Outliers

Data['Sales'] = Data['Sales'].clip(lower = 10, upper = 100)
var_summary(Data.Sales)
```

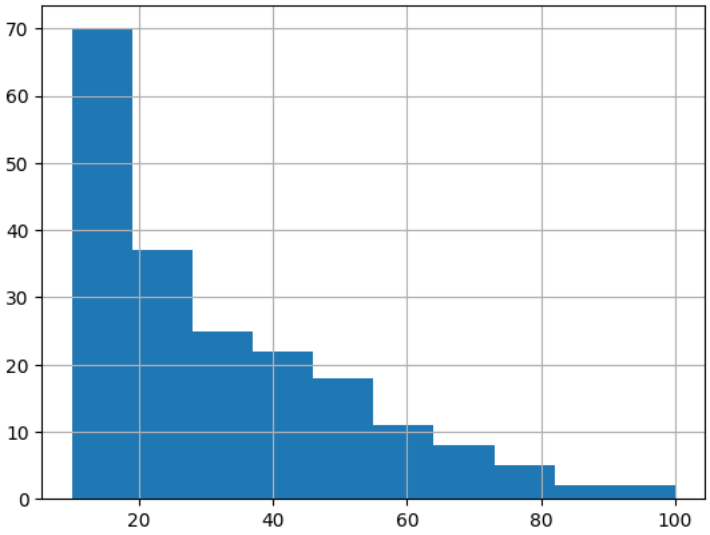
```
N               200.000000
NMISS             0.000000
SUM            3119.200000
MEAN             15.596000
MEDIAN           16.000000
STD               4.583725
VAR              21.010537
MIN              10.000000
P1               10.000000
P5               10.000000
P10              10.000000
P25              11.000000
P50              16.000000
P75              19.050000
P90              21.710000
P95              23.800000
P99              25.507000
MAX              27.000000
LC                6.428550
UC               24.763450
outlier_flag      0.000000
dtype: float64
```

```
Data['Sales']=Data['Sales'].fillna(Data['Sales'].mean())
sns.distplot(Data.Sales)
```

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

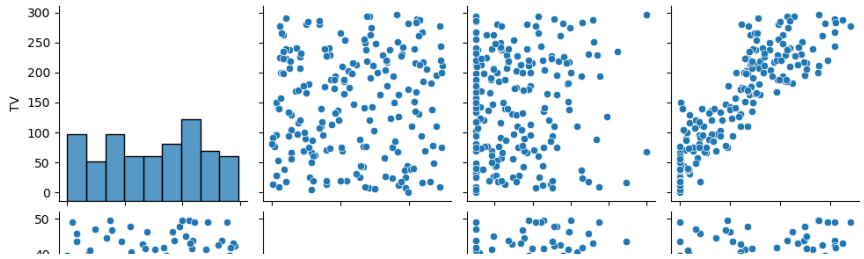

```
Data.Newspaper.hist()
```

```
<Axes: >
```




```
sns.pairplot ( Data)
```

```
<seaborn.axisgrid.PairGrid at 0x7ba81a832890>
```



Correlation

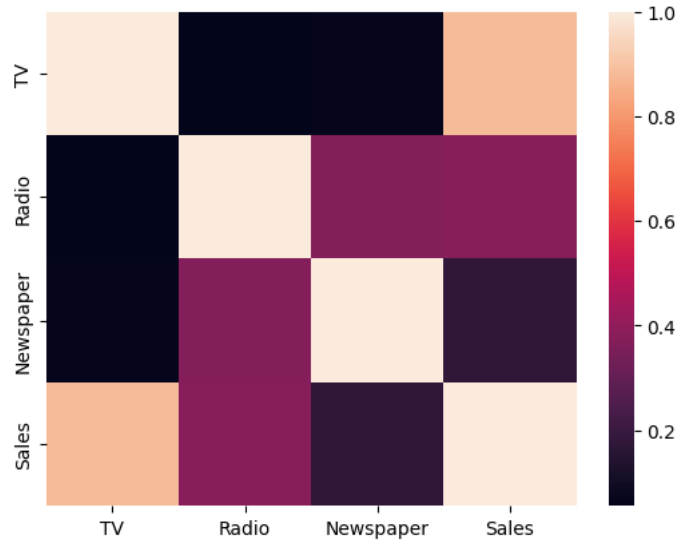
Correlation is a statistical measure that expresses the extent to which two variables are linearly related (meaning they change together at a constant rate). It's a common tool for describing simple relationships without making a statement about cause and effect. Here we are calculating the correlation between different variables with corr() command


```
Data.corr()
```

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.065510	0.883656
Radio	0.054809	1.000000	0.368934	0.378195
Newspaper	0.065510	0.368934	1.000000	0.174389
Sales	0.883656	0.378195	0.174389	1.000000

```
sns.heatmap( Data.corr())
```

```
<Axes: >
```




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

Splitting Training-Testing Dataset

We need to split a dataset into train and test sets to evaluate how well our machine learning model performs. We are splitting our dataset in training and testing dataset with the help of train_test_split() function.

```
x_train, x_test, y_train, y_test=train_test_split(x,y,random_state=2,test_size=15)
print(x.shape,x_train.shape,x_test.shape)
```

```
(200, 3) (185, 3) (15, 3)
```

** Fitting the model**

Model fitting is a measure of how well a machine learning model generalizes to similar data to that on which it was trained. On our data we are fitting linear regression model. Linear regression analysis is used to predict the value of a variable based on the value of another variable.

```
model=LinearRegression()
model.fit(x_train,y_train)
x_train_prediction=model.predict(x_train)
```

Model Evaluation By Calculating Root-Mean squared error

```
mean_square_error =mean_squared_error(y_train,x_train_prediction)
mean_square_error
```

```
2.305082878332641
```

```
root_mean_square_error =np.sqrt(mean_square_error)
root_mean_square_error
```

```
1.518249939348802
```