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 see the task of sales prediction with machine learning using python.

**Modules needed:

pandas: Pandas is an opensource library that allows you to perform data manipulation in Python. Pandas provide an easy way to create, manipulate and wrangle the data.

numpy: Numpy is the fundamental package for scientific computing with Python. numpy can be used as an efficient multi-dimensional container of generic data.

matplotlib: Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of formats.

seaborn: Seaborn is a Python data-visualization library that is based on matplotlib. Seaborn provides a high-level interface for drawing attractive and informative statistical graphics.

scipy: Scipy is a Python-based ecosystem of open-source software for mathematics, science, and engineering.

In [1]:
```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
#loading dataset
df = pd.read_csv('Advertising.csv')
```

In [3]:
```
#Displaying the dataset
print("Sales advertising data set: \n")
df
```

```
Sales advertising data set:
```

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]:
```
#First 7 rows of dataset
print("Top seven rows of dataset are: ")
df.head(7)
```

```
Top seven rows of dataset are:
```

Out[4]:

	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
5	6	8.7	48.9	75.0	7.2
6	7	57.5	32.8	23.5	11.8

In [5]:
```
#Last 5 rows of dataset
#If no number is given then it takes default number as 5
print("Last 5 rows of dataset are: ")
df.tail()
```

```
Last 5 rows of dataset are:
```

Out[5]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
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

In [6]:
```
#Information of dataset
print("Summarized information of the dataset: \n")
df.info()
```

```
Summarized information of the dataset:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Unnamed: 0  200 non-null    int64
 1   TV          200 non-null    float64
 2   Radio       200 non-null    float64
 3   Newspaper   200 non-null    float64
 4   Sales       200 non-null    float64
dtypes: float64(4), int64(1)
memory usage: 7.9 KB
```

In [7]:
```
#Summary statistics of dataset
#describe() function is used which gives all the summary statistics like mean
```


Out[7]:

	Unnamed: 0	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000	200.000000
mean	100.500000	147.042500	23.264000	30.554000	14.022500
std	57.879185	85.854236	14.846809	21.778621	5.217457
min	1.000000	0.700000	0.000000	0.300000	1.600000
25%	50.750000	74.375000	9.975000	12.750000	10.375000
50%	100.500000	149.750000	22.900000	25.750000	12.900000
75%	150.250000	218.825000	36.525000	45.100000	17.400000
max	200.000000	296.400000	49.600000	114.000000	27.000000

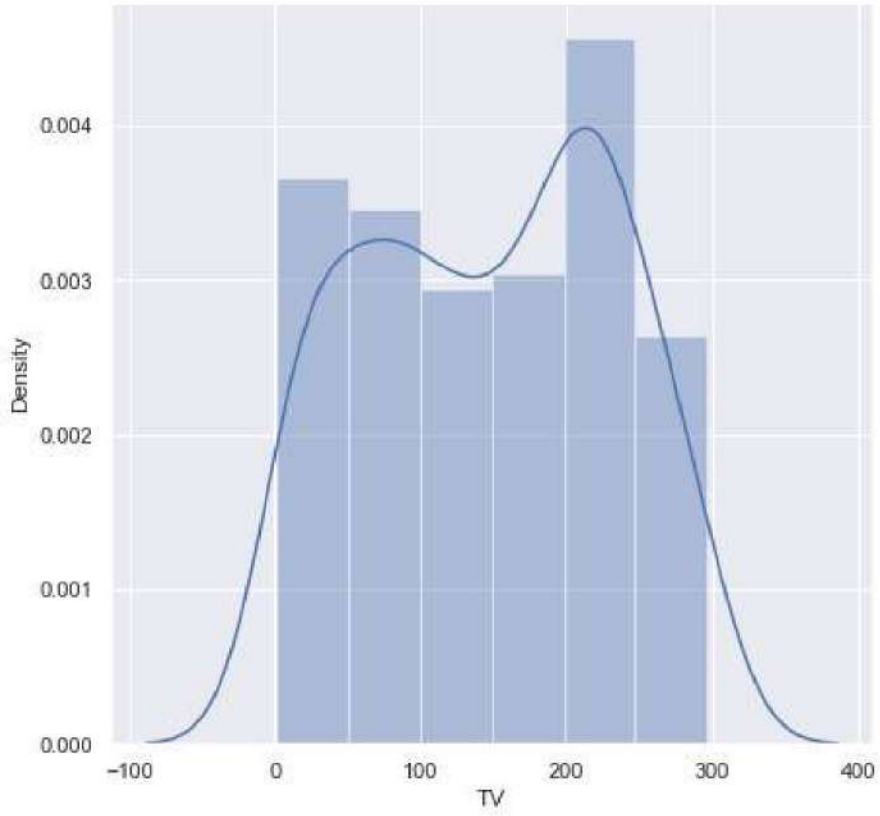
In [8]:
```
#Dimensions and size information about the dataset
#shape() function gives the dimensions of the dataset
print("Dimensions of the dataset are: ",df.shape)
print("Size of advertisement dataset are: ",df.size)
```

```
Dimensions of the dataset are:  (200, 5)
Size of advertisement dataset are:  1000
```

In [12]:

```
#TV distribution
plt.figure(figsize=(7,7))
sns.distplot(df['TV'])
plt.show()
```

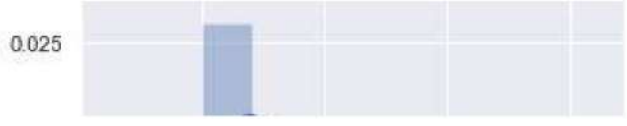
```
C:\Users\meghana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```

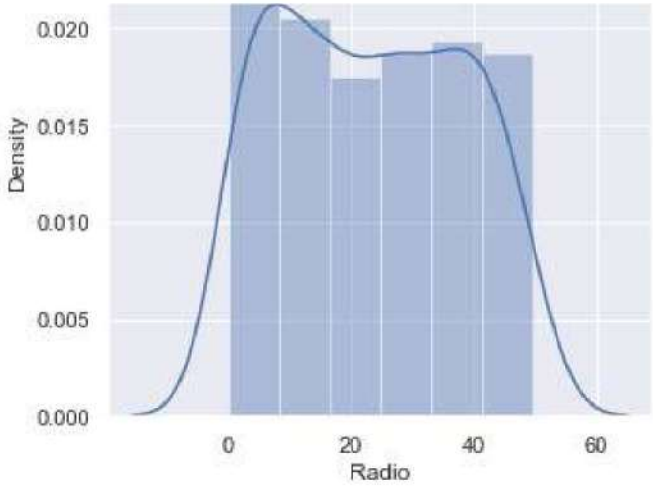


In [13]:

```
#Radio distribution
plt.figure(figsize=(5,5))
sns.distplot(df['Radio'])
plt.show()
```

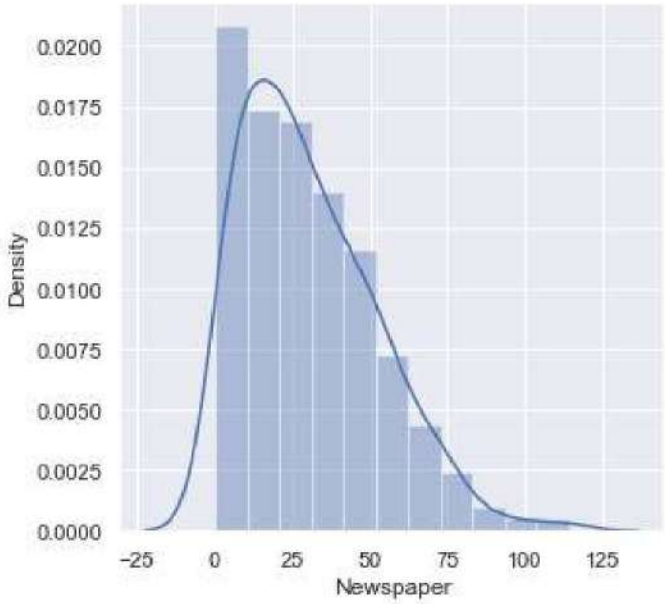
```
C:\Users\meghana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```





In [14]:
```
#Newspaper distribution
plt.figure(figsize=(5,5))
sns.distplot(df['Newspaper'])
plt.show()
```

```
C:\Users\meghana\anaconda3\lib\site-packages\seaborn\distributions.py:2619: Fut
ureWarning: `distplot` is a deprecated function and will be removed in a future
version. Please adapt your code to use either `displot` (a figure-level functio
n with similar flexibility) or `histplot` (an axes-level function for histogram
s).
  warnings.warn(msg, FutureWarning)
```



In [15]:
```
df.isnull().sum()
```

Out[15]:
```
Unnamed: 0    0
TV            0
Radio         0
Newspaper     0
Sales         0
dtype: int64
```

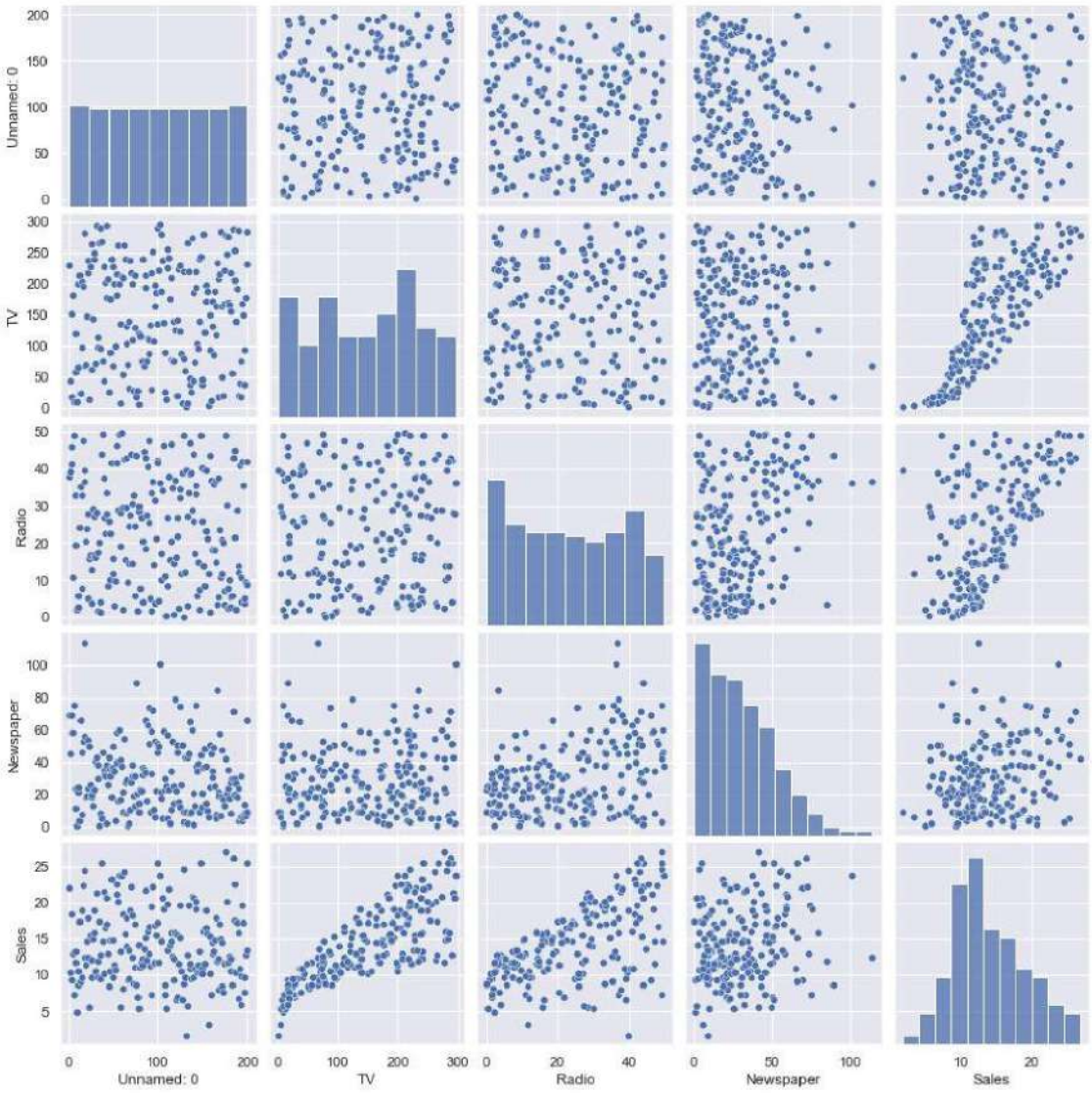
In [16]:

```
df.duplicated().value_counts()
```

Out[16]:
```
False    200
dtype: int64
```

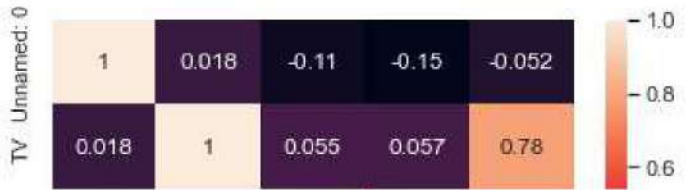
In [17]:
```
sns.pairplot(df)
```

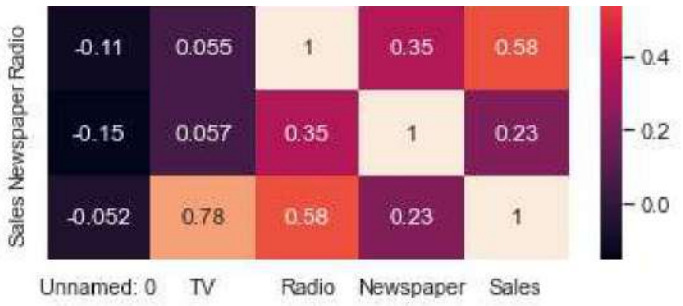
Out[17]: <seaborn.axisgrid.PairGrid at 0x29b6a638be0>



In [18]:
```
#heatmap
correlation=df.corr()
sns.heatmap(correlation, xticklabels=correlation.columns, yticklabels=correla
```

Out[18]: <AxesSubplot:>





Data Modelling

In [19]:
```
df1 = df.drop(['Unnamed: 0'], axis=1, inplace=True)
df
```

Out[19]:

	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

In [20]:
```
df.columns
```

Out[20]:
```
Index(['TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [21]:
```
X =np.array(df.drop(['Sales'], 1))
y =np.array(df['Sales'])
```

```
C:\Users\meghana\AppData\Local\Temp/ipykernel_15532/1751143456.py:1: FutureWarn
ing: In a future version of pandas all arguments of DataFrame.drop except for t
he argument 'labels' will be keyword-only
  X =np.array(df.drop(['Sales'], 1))
```

In [22]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test= train_test_split(X, y,train_size=0.8,test_s
```

In [23]:
```
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[23]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [24]:
```
y_predict = regressor.predict(X_test)
y_predict
```

Out[24]:
```
array([10.05739563,  7.4522807 ,  7.0197076 , 24.08029725, 12.01786259,
        6.53793858, 12.78286918, 15.10974587, 10.76974013, 16.34357951,
       22.88297477,  9.12924467, 10.46455672, 15.48743552, 11.58555633,
       12.17296914, 18.76551502, 10.78318566, 15.90515992, 17.30651279,
       24.06692057,  9.59834224, 15.13512211, 12.38591525,  5.71360885,
       15.24749314, 12.29402334, 20.9421167 , 13.40991558,  9.04348832,
       12.89239415, 21.40272028, 18.13802209, 21.17320803,  6.56974433,
        6.14114206,  7.89018394, 13.01541434, 14.68953791,  6.18835143])
```

In [25]:
```
# calculating the coefficient
coefficient = regressor.coef_
coefficient
```

Out[25]:
```
array([ 0.04458402,  0.19649703, -0.00278146])
```

In [26]:
```
# calculating the intercept
intercept = regressor.intercept_
intercept
```

Out[26]:
```
2.9948930304953283
```

In [27]:
```
# calculating the R squared value
from sklearn.metrics import r2_score
r2_score(y_test, y_predict)
```

Out[27]:
```
0.8601145185017867
```

In [50]:
```
forecast=pd.DataFrame(data={'Forecasted Sales': y_predict.flatten()})
forecast
```

Out[50]:

	Forecasted Sales
0	10.057396
1	7.452281

2	7.019708
3	24.080297
4	12.017863
5	6.537939
6	12.782869
7	15.109746
8	10.769740
9	16.343580
10	22.882975
11	9.129245
12	10.464557
13	15.487436
14	11.585556
15	12.172969
16	18.765515
17	10.783186
18	15.905160
19	17.306513
20	24.066921
21	9.598342
22	15.135122
23	12.385915
24	5.713609
25	15.247493
26	12.294023
27	20.942117
28	13.409916
29	9.043488
30	12.892394
31	21.402720
32	18.138022
33	21.173208
34	6.569744

35	6.141142
36	7.890184
37	13.015414