Advertising Sales Channel Prediction

Problem Statement:

When a company enters a market, the distribution strategy and channel it uses are keys to its success in the market, as well as market know-how and customer knowledge and understanding. Because an effective distribution strategy under efficient supply-chain management opens doors for attaining competitive advantage and strong brand equity in the market, it is a component of the marketing mix that cannot be ignored .

The distribution strategy and the channel design have to be right the first time. The case study of Sales channel includes the detailed study of TV, radio and newspaper channel. The predict the total sales generated from all the sales channel.

import the libraries

In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

import dataset

In [2]:
```
df=pd.read_csv('Advertising_Sales_Channel_Prediction.csv')
df.head()
```

Out[2]:

	TV	radio	newspaper	sales
1	230.1	37.8	69.2	22.1
2	44.5	39.3	45.1	10.4
3	17.2	45.9	69.3	9.3
4	151.5	41.3	58.5	18.5
5	180.8	10.8	58.4	12.9

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

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

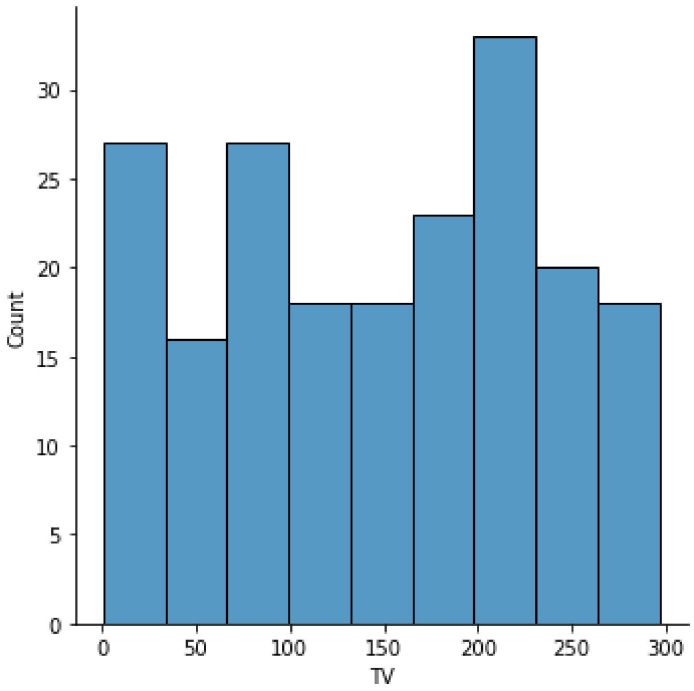
In [4]:
```
df.info()
```

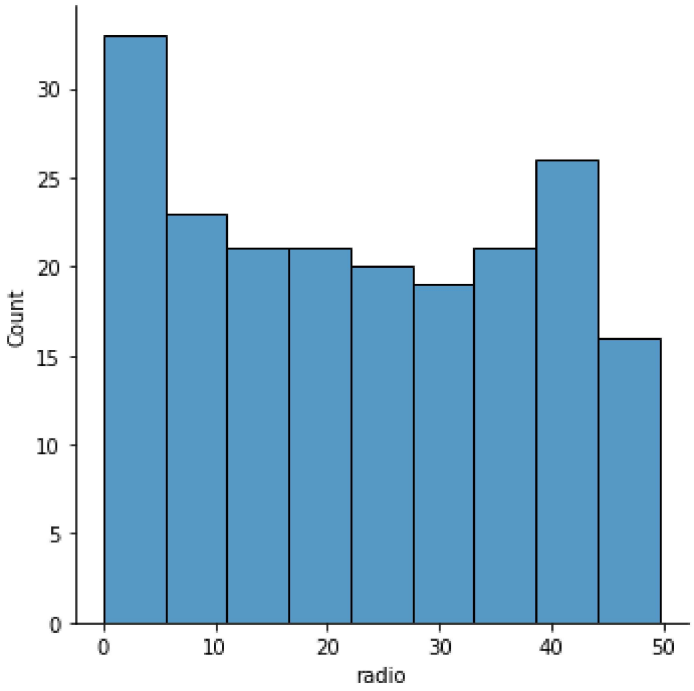
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   TV         200 non-null    float64
 1   radio      200 non-null    float64
 2   newspaper  200 non-null    float64
 3   sales      200 non-null    float64
dtypes: float64(4)
memory usage: 7.8 KB
```

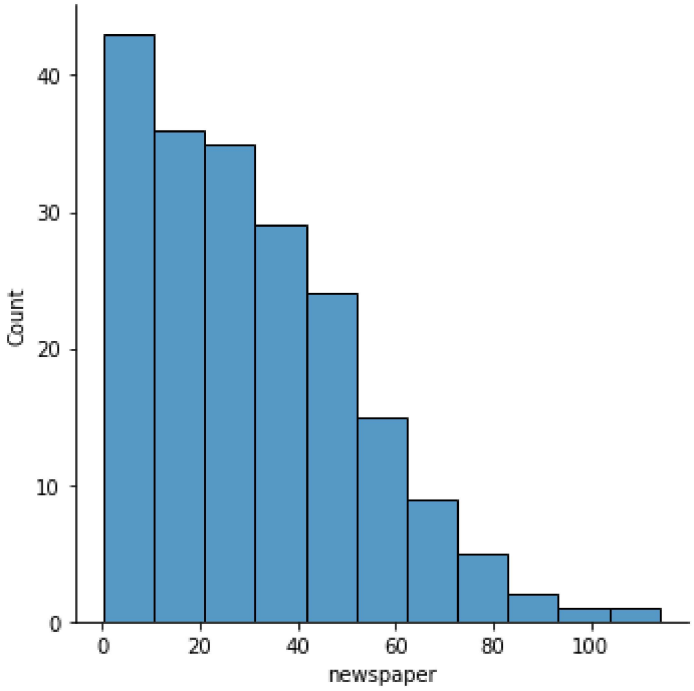
In [5]:
```
df.isnull().sum()
```

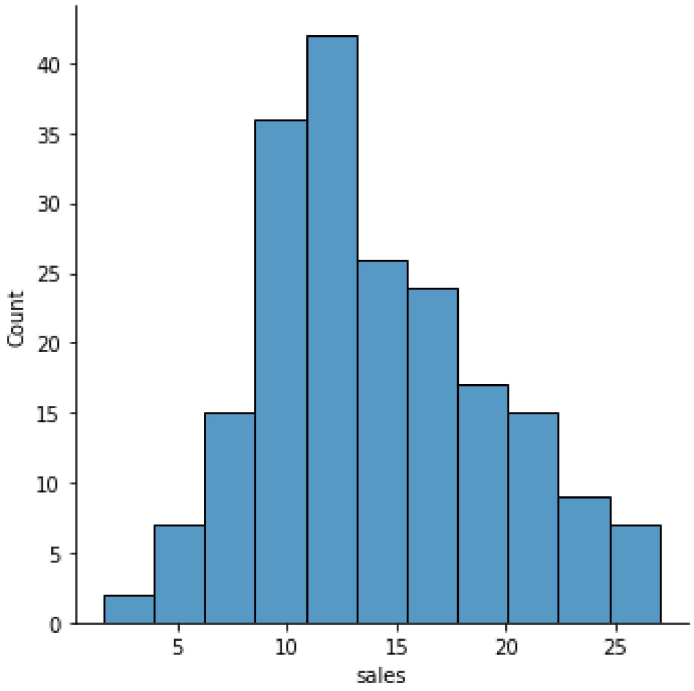
Out[5]:
```
TV           0
radio        0
newspaper    0
sales        0
dtype: int64
```

In [6]:
```
#creating a for loop to get the distribution plot for all columns
for column in df:
    sns.displot(x=column,data=df)
```



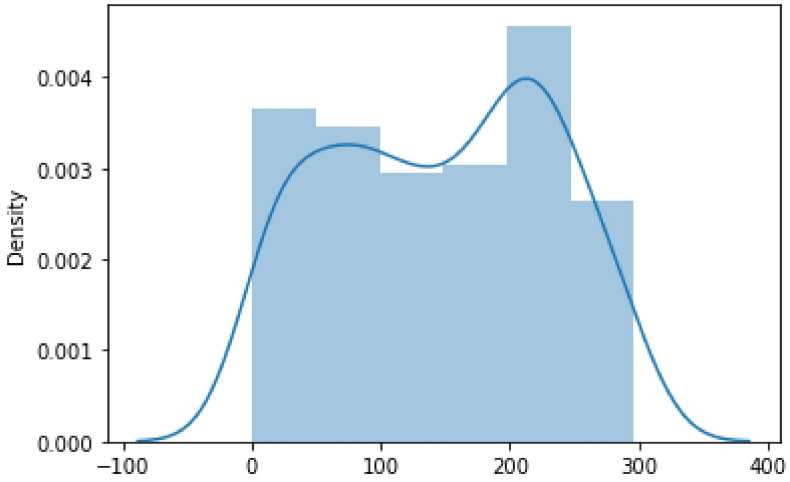

Count
radio
0
5
10
15
20
25
30
0
10
20
30
40
50


Count
newspaper
0
10
20
30
40
0
20
40
60
80
100



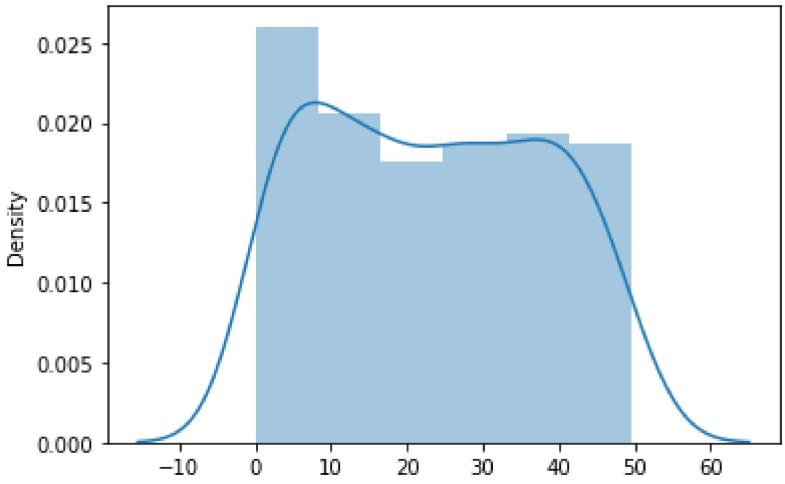
In [7]:
```
sns.distplot(x=df.TV)
```

Out[7]:
```
<AxesSubplot:ylabel='Density'>
```



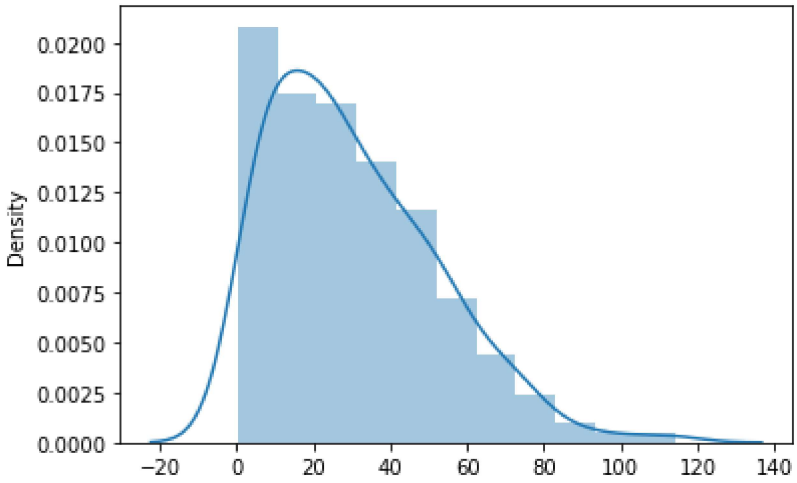
In [8]:
```
sns.distplot(x=df.radio)
```

Out[8]:
```
<AxesSubplot:ylabel='Density'>
```



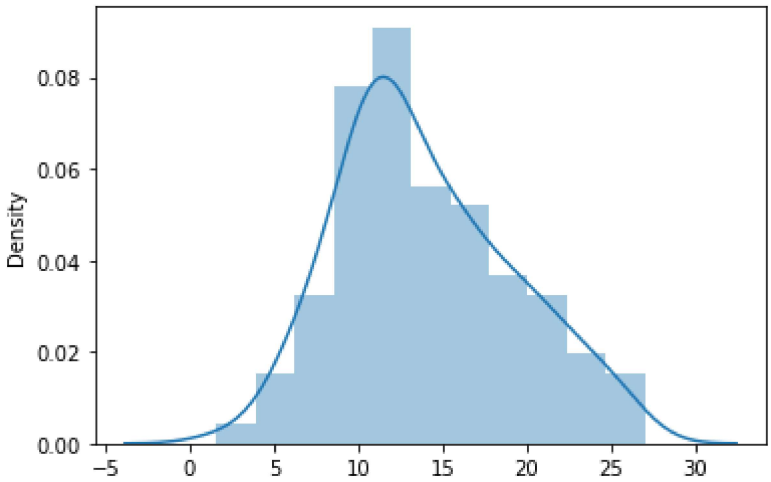
In [9]:
```
sns.distplot(x=df.newspaper)
```

Out[9]:
```
<AxesSubplot:ylabel='Density'>
```



In [10]:
```
sns.distplot(x=df.sales)
```

Out[10]:
```
<AxesSubplot:ylabel='Density'>
```



In [11]:
```
df.describe()
```

Out[11]:

	TV	radio	newspaper	sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	14.022500
std	85.854236	14.846809	21.778621	5.217457
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	10.375000
50%	149.750000	22.900000	25.750000	12.900000
75%	218.825000	36.525000	45.100000	17.400000
max	296.400000	49.600000	114.000000	27.000000

Correlation Matrix

In [12]:
```
#Contructing a heat mat to visualize the correlation matrix
plt.figure(figsize=(15,10))
sns.heatmap(df.corr(),annot=True, linewidths=0.1,linecolor="black",fmt="0.2f")
```

Out[12]:
```
<AxesSubplot:>
```

Splitting the data and Target

In [13]:
```
x=df[['TV','radio','newspaper']]
y=df['sales']
```

Splitting Training and Test Data

In [14]:
```
from sklearn.model_selection import train_test_split

x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.1,random_state=2)
```

Model training

In [15]:
```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
```

Out[15]:
```
LinearRegression()
```

In [16]:
```
print("intercept:",lr.intercept_)
print("cofficients:",lr.coef_)
lr.score(x_test,y_test)
```

```
intercept: 2.933283514829732
cofficients: [ 4.58551301e-02  1.89190950e-01 -1.54837230e-04]
```

Out[16]:
```
0.8169646763804375
```

In [17]:
```
y_pred_lr=lr.predict(x_test)
print('prediction for test set: {}'.format(y_pred_lr))
```

```
prediction for test set: [13.90319889  9.19139355  6.58413158 15.20195429 18.52738171
15.26343532
  7.05786806 20.42566364 12.99052713 17.0377366  10.6550342  19.24470677
  8.84089509 10.64264674 13.7726853  12.39512616  8.93975025 17.68486399
 16.4401956  18.50957532]
```

In [18]:
```
lr_diff=pd.DataFrame({'Actual values':y_test,'predicted values':y_pred_lr})
lr_diff
```

Out[18]:

	Actual values	predicted values
113	14.1	13.903199
30	10.5	9.191394
183	8.7	6.584132
200	13.4	15.201954
194	19.6	18.527382
86	15.2	15.263435
11	8.6	7.057868
55	20.2	20.425664
116	12.6	12.990527
36	12.8	17.037737
13	9.2	10.655034
93	19.4	19.244707
14	9.7	8.840895
127	6.6	10.642647
175	11.5	13.772685
3	9.3	12.395126
45	8.5	8.939750
4	18.5	17.684864
114	15.9	16.440196
15	19.0	18.509575

In [19]:

```
from sklearn import metrics
mean_abs=metrics.mean_absolute_error(y_test,y_pred_lr)
mean_abs_squr=metrics.mean_squared_error(y_test,y_pred_lr)
root_mean_sqrerr=np.sqrt(metrics.mean_squared_error(y_test,y_pred_lr))

print('R squared: {:.4f}'.format(lr.score(x,y)*100))
print('Mean Absolute Error:',mean_abs)
print('Mean Square Error:',mean_abs_squr)
print('Root Mean Square Error:',root_mean_sqrerr)
```

```
R squared: 89.7093
Mean Absolute Error: 1.3560370133709394
Mean Square Error: 3.3263512708523053
Root Mean Square Error: 1.8238287394523383
```