SALES PREDICTION

Importing All The Necessary Libraries

In [144...
```
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  #splits data into training and testing model
from sklearn.linear_model import LinearRegression
from sklearn.metrics import accuracy_score, mean_squared_error, r2_score
```

In [145...
```
#Loading the CSV file into a Pandas dataframe
df=pd.read_csv("C:/Users/HiSBL Aggarwal/Downloads/advertising.csv")
```

In [146...
```
df.head()
```

Out[146...

	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

In [147...
```
df.shape
```

Out[147...
```
(200, 4)
```

In [148...
```
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 [149...
```
df.describe()
```

Out[149...

	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

In [150...
```
df.isnull().sum()
```

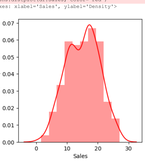
Out[150...
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

Data Visualization

In [151...
```
plt.figure(figsize=(4,4))
sns.distplot(df.Sales, color='red')
```

```
C:\Users\HiSBL Aggarwal\AppData\Local\Temp\ipykernel_10004\3744209507.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Sales, color='red')
```

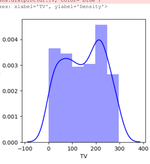
Out[151...
```
<Axes: xlabel='Sales', ylabel='Density'>
```



In [152...
```
plt.figure(figsize=(4,4))
sns.distplot(df.TV, color='blue')
```

```
C:\Users\HiSBL Aggarwal\AppData\Local\Temp\ipykernel_10004\191350807.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.TV, color='blue')
```

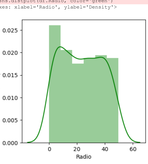
Out[152...
```
<Axes: xlabel='TV', ylabel='Density'>
```



In [153...
```
plt.figure(figsize=(4,4))
sns.distplot(df.Radio, color='green')
```

```
C:\Users\HiSBL Aggarwal\AppData\Local\Temp\ipykernel_10004\1642820241.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Radio, color='green')
```

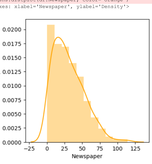
Out[153...
```
<Axes: xlabel='Radio', ylabel='Density'>
```



In [154...
```
plt.figure(figsize=(4,4))
sns.distplot(df.Newspaper, color='orange')
```

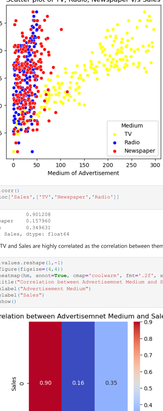
```
C:\Users\HiSBL Aggarwal\AppData\Local\Temp\ipykernel_10004\3360342774.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(df.Newspaper, color='orange')
```

Out[154...
```
<Axes: xlabel='Newspaper', ylabel='Density'>
```



In [155...
```
plt.figure(figsize=(5,5))
sns.scatterplot(data=df, x= 'TV', y='Sales', color='yellow', label="TV")
sns.scatterplot(data=df, x='Radio', y='Sales', color='blue', label='Radio')
sns.scatterplot(data=df, x='Newspaper', y='Sales', color='red', label='Newspaper')
plt.title("Scatter plot of TV, Radio, Newspaper v/s Sales")
plt.xlabel("Medium of Advertisement")
plt.ylabel("Sales")
plt.legend(title='Medium')
```

Out[155...
```
<matplotlib.legend.Legend at 0x237fdc54950>
```



In [156...
```
cm=df.corr()
ym.loc['Sales',['TV','Newspaper','Radio']]
y
```

Out[156...
```
TV           0.901208
Newspaper    0.157960
Radio        0.349631
Name: Sales, dtype: float64
```

Here, TV and Sales are highly correlated as the correlation between them is almost equal to 1

In [157...
```
hmy=y.values.reshape(1,-1)
plt.figure(figsize=(4,4))
sns.heatmap(hm, annot=True, cmap='coolwarm', fmt='.2f', xticklabels=['TV','Radio','Newspaper'])
plt.title("Correlation between Advertisemnet Medium and Sales")
plt.xlabel("Advertisement Medium")
plt.ylabel("Sales")
plt.show()
```


In [158...
```
#The features X and the target Y i.e. Sales
X=df[['TV','Radio','Newspaper']]
Y=df['Sales']
```

Splitting The Data Into Training And Testing Data

In [159...
```
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [160...
```
X_train.shape, X_test.shape, X.shape
```

Out[160...
```
((160, 3), (40, 3), (200, 3))
```

Model Training Using Linear Regression

In [161...
```
model=LinearRegression()
```

In [162...
```
model.fit(X_train, Y_train)
```

Out[162...
```
▾LinearRegression
LinearRegression()
```

In [163...
```
Y_pred=model.predict(X_test)
```

In [164...
```
mse=mean_squared_error(Y_test, Y_pred)
```

In [167...
```
print("Mean Squared Error: ", mse)
r2=r2_score(Y_test, Y_pred)
print("Root Squared Error: ",r2)
```

```
Mean Squared Error:  2.907756910271095
Root Squared Error:  0.9059011844150826
```

Here for the fact that TV and Sales are highly correlated, I have used them to train and predict the model.

In [168...
```
X=df[['TV']]
Y=df['Sales']
```

In [170...
```
X_train, X_test, Y_train, Y_test= train_test_split(X, Y, test_size=0.2, random_state=42)
```

In [171...
```
model=LinearRegression()
model.fit(X_train, Y_train)
```

Out[171...
```
▾LinearRegression
LinearRegression()
```

In [172...
```
Y_pred=model.predict(X_test)
mse=mean_squared_error(Y_test, Y_pred)
print("Mean Squared Error: ", mse)
r2=r2_score(Y_test, Y_pred)
print("Root Squared Error: ",r2)
```

```
Mean Squared Error:  6.101072906773963
Root Squared Error:  0.8025613534623698
```

In [ ]
```

```