Oasis Infobyte - OIBSIP - Data Science

Task 3 : Sales Prediction

Intern - Shravani Mahesuni

Problem Statement : Sales forecasting is the art of estimating how much of a product consumers will purchase depending on variables like the amount of money spent on promoting your product, the target market you target, or the platform you are using for your advertising.Typically, a product- and service-based company needs a data scientist to forecast future sales with each decision they make to control the expense of product promotion.

In [69]:
```
#importing python libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score,confusion_matrix
```

Loading or Reading Dataset

In [8]:
```
#loading dataset and print whole data preview
df = pd.read_csv('Advertising.csv',encoding='latin1')
df
```

Out[8]:

	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 [9]:
```
# Print first 5 rows of data
df.head(5)
```

Out[9]:

	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

In [10]:
```
# get number of rows and columns
df.shape
```

Out[10]:
```
(200, 5)
```

In [11]:
```
# get coulmn names
df.columns
```

Out[11]:
```
Index(['Unnamed: 0', 'TV', 'Radio', 'Newspaper', 'Sales'], dtype='object')
```

In [12]:
```
# scrutinizing rows and columns
print('rows-->',df.shape[0])
print('columns-->',df.shape[1])
```

```
rows--> 200
columns--> 5
```

As you can see there are missing entries in some columns that's not at all required for analyzing dataset, Therefore, these null values fields should be eliminated.

In [14]:
```
#removing insignificant columns
df = df.drop(columns=['Unnamed: 0'])
```

In [15]:
```
df
```

Out[15]:

	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

Now one column got reduced accurately !

In [16]:
```
df.shape
```

Out[16]:
```
(200, 4)
```

In [17]:
```
df.describe()
```

Out[17]:

	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

In [18]:
```
x = df.iloc[:,0:-1]
```

In [19]:
```
x
```

Out[19]:

	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 [24]:
```
y = df.iloc[:,-1]
```

In [25]:
```
# sales data prediction
y
```

Out[25]:
```
0      22.1
1      10.4
2       9.3
3      18.5
4      12.9
       ...
195     7.6
196     9.7
197    12.8
198    25.5
199    13.4
Name: Sales, Length: 200, dtype: float64
```

In [26]:
```
x.iloc[:,0]
```

Out[26]:
```
0      230.1
1       44.5
2       17.2
3      151.5
4      180.8
       ...
195     38.2
196     94.2
197    177.0
198    283.6
199    232.1
Name: TV, Length: 200, dtype: float64
```

In [27]:
```
#displaying first 2 rows
df.head(2)
```

Out[27]:

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4

In [28]:
```
#Identify the pairwise correlation between each column in the DataFrame.
df.corr()
```

Out[28]:

	TV	Radio	Newspaper	Sales
TV	1.000000	0.054809	0.056648	0.782224
Radio	0.054809	1.000000	0.354104	0.576223
Newspaper	0.056648	0.354104	1.000000	0.228299
Sales	0.782224	0.576223	0.228299	1.000000

In [31]:
```
# integrating statistical graphics
sns.pairplot(data=df)
plt.show()
```

Now we need to test hypothesis to estimate performance .

Train_test_split

In [36]:
```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.2,random_state=43)
```

In [37]:
```
xtrain
```

Out[37]:

	TV	Radio	Newspaper
116	139.2	14.3	25.6
138	43.0	25.9	20.5
155	4.1	11.6	5.7
82	75.3	20.3	32.5
160	172.5	18.1	30.7
...	...	...	...
58	210.8	49.6	37.7
21	237.4	5.1	23.5
49	66.9	11.7	36.8
64	131.1	42.8	28.9
68	237.4	27.5	11.0

160 rows × 3 columns

In [38]:
```
xtest
```

Out[38]:

	TV	Radio	Newspaper
56	7.3	28.1	41.4
37	74.7	49.4	45.7
67	139.3	14.5	10.2
79	116.0	7.7	23.1
80	76.4	26.7	22.3
188	286.0	13.9	3.7
183	287.6	43.0	71.8
10	66.1	5.8	24.2
128	220.3	49.0	3.2
62	239.3	15.5	27.3
65	69.0	9.3	0.9
17	281.4	39.6	55.8
133	219.8	33.5	45.1
195	38.2	3.7	13.8
146	240.1	7.3	8.7
38	43.1	26.7	35.1
173	168.4	7.1	12.8
149	44.7	25.8	20.6
93	250.9	36.5	72.3
29	70.6	16.0	40.8
0	230.1	37.8	69.2
2	17.2	45.9	69.3
122	224.0	2.4	15.6
180	156.6	2.6	8.3
95	163.3	31.6	52.9
121	18.8	21.7	50.4
185	205.0	45.1	19.6
39	228.0	37.7	32.0
66	31.5	24.6	2.2
19	147.3	23.9	19.1
11	214.7	24.0	4.0
45	175.1	22.5	31.5
41	177.0	33.4	38.7
92	217.7	33.5	59.0
168	215.4	23.6	57.6
1	44.5	39.3	45.1
57	136.2	19.2	16.6
189	18.7	12.1	23.4
151	121.0	8.4	48.7
167	206.8	5.2	19.4

In [39]:
```
ytrain
```

Out[39]:
```
116    12.2
138     9.6
155     3.2
82     11.3
160    14.4
       ...
58     23.8
21     12.5
49      9.7
64     18.0
68     18.9
Name: Sales, Length: 160, dtype: float64
```

In [40]:
```
ytest
```

Out[40]:
```
56      5.5
37     14.7
67     13.4
79     11.0
80     11.8
188    15.9
183    26.2
10      8.6
128    24.7
62     15.7
65      9.3
17     24.4
133    19.6
195     7.6
146    13.2
38     10.1
173    11.7
149    10.1
93     22.2
29     10.5
0      22.1
2       9.3
122    11.6
180    10.5
95     16.9
121     7.0
185    22.6
39     21.5
66      9.5
19     14.6
11     17.4
45     14.9
41     17.1
92     19.4
168    17.1
1      10.4
57     13.2
189     6.7
151    11.6
167    12.2
Name: Sales, dtype: float64
```

In [41]:
```
xtrain=xtrain.astype(int)
ytrain=ytrain.astype(int)
xtest=xtest.astype(int)
ytest=ytest.astype(int)
```

As we've imported StandardScalar from sklearn

In [43]:
```
sc = StandardScaler()
```

In [44]:
```
xtrain_scaled = sc.fit_transform(xtrain)
xtest_scaled = sc.fit_transform(xtest)
```

In [46]:
```
lg = LinearRegression()
```

In [47]:
```
lg.fit(xtrain_scaled,ytrain)
```

Out[47]:
```
▾ LinearRegression
LinearRegression()
```

method used for predictive analysis in machine learning.

In [48]:
```
y_pred = lg.predict(xtest_scaled)
```

In [49]:
```
y_pred
```

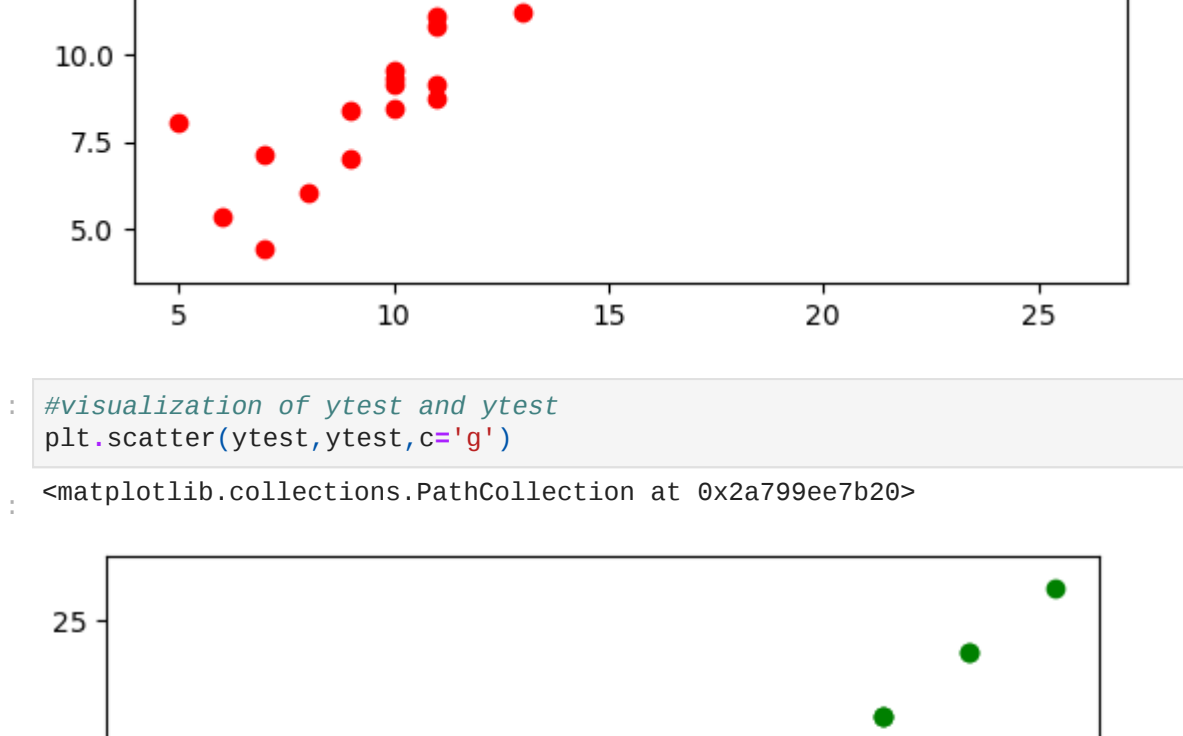
Out[49]:
```
array([ 8.07208561, 15.39694276, 11.20723017,  8.7231933 , 10.80138195,
       17.66036561, 23.785389  ,  6.05064595, 22.03934034, 15.92229543,
        7.02529552, 22.70917874, 18.68464752,  4.38385522, 14.34701329,
        9.2970207 , 11.08559493,  9.15049517, 20.67810902,  8.46853177,
       19.98026988, 11.97879363, 12.59404776,  9.52303143, 15.73547183,
        7.12996739, 20.52852873, 19.92139498,  8.37303394, 13.40290607,
       16.65207302, 14.45505255, 16.79013098, 18.58218129, 16.44742571,
       11.99347795, 12.08933433,  5.31167661,  9.13267146, 12.38991849])
```

In [50]:
```
ytest
```

Out[50]:
```
56      5
37     14
67     13
79     11
80     11
188    15
183    26
10      8
128    24
62     15
65      9
17     24
133    19
195     7
146    13
38     10
173    11
149    10
93     22
29     10
0      22
2       9
122    11
180    10
95     16
121     7
185    22
39     21
66      9
19     14
11     17
45     14
41     17
92     19
168    17
1      10
57     13
189     6
151    11
167    12
Name: Sales, dtype: int32
```

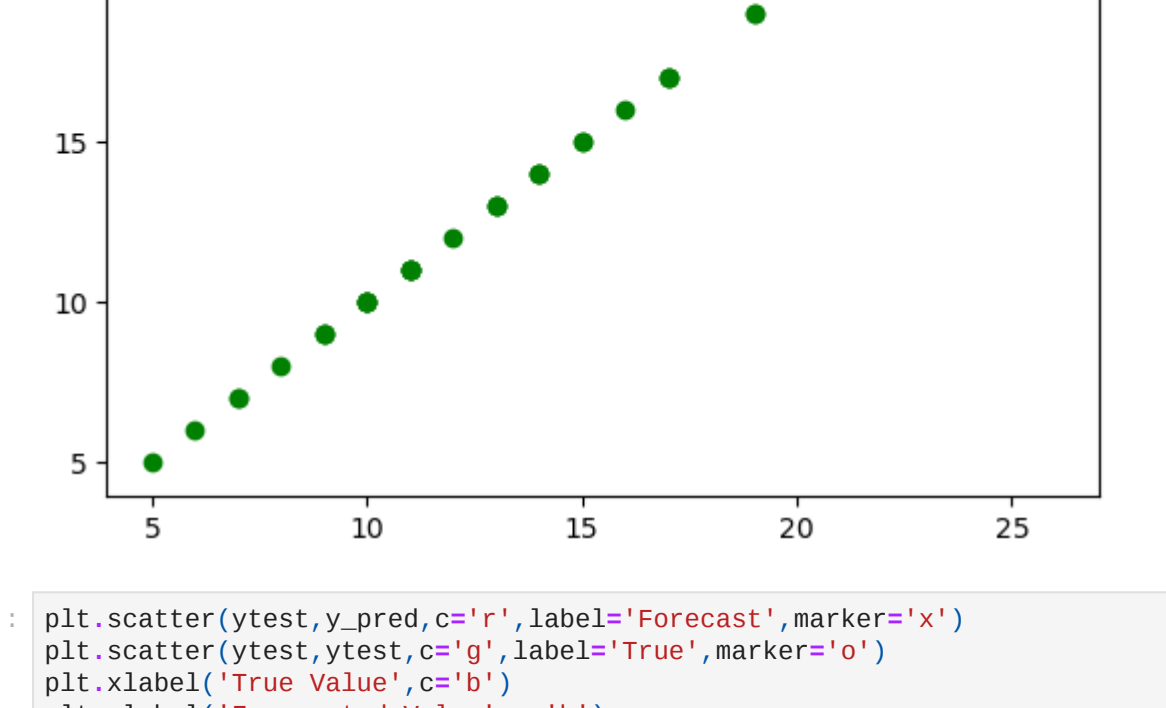
In [53]:
```
#visualization of ytest and y_pred
plt.scatter(ytest,y_pred,c='r')
```

Out[53]:
```
<matplotlib.collections.PathCollection at 0x2a78aee5870>
```

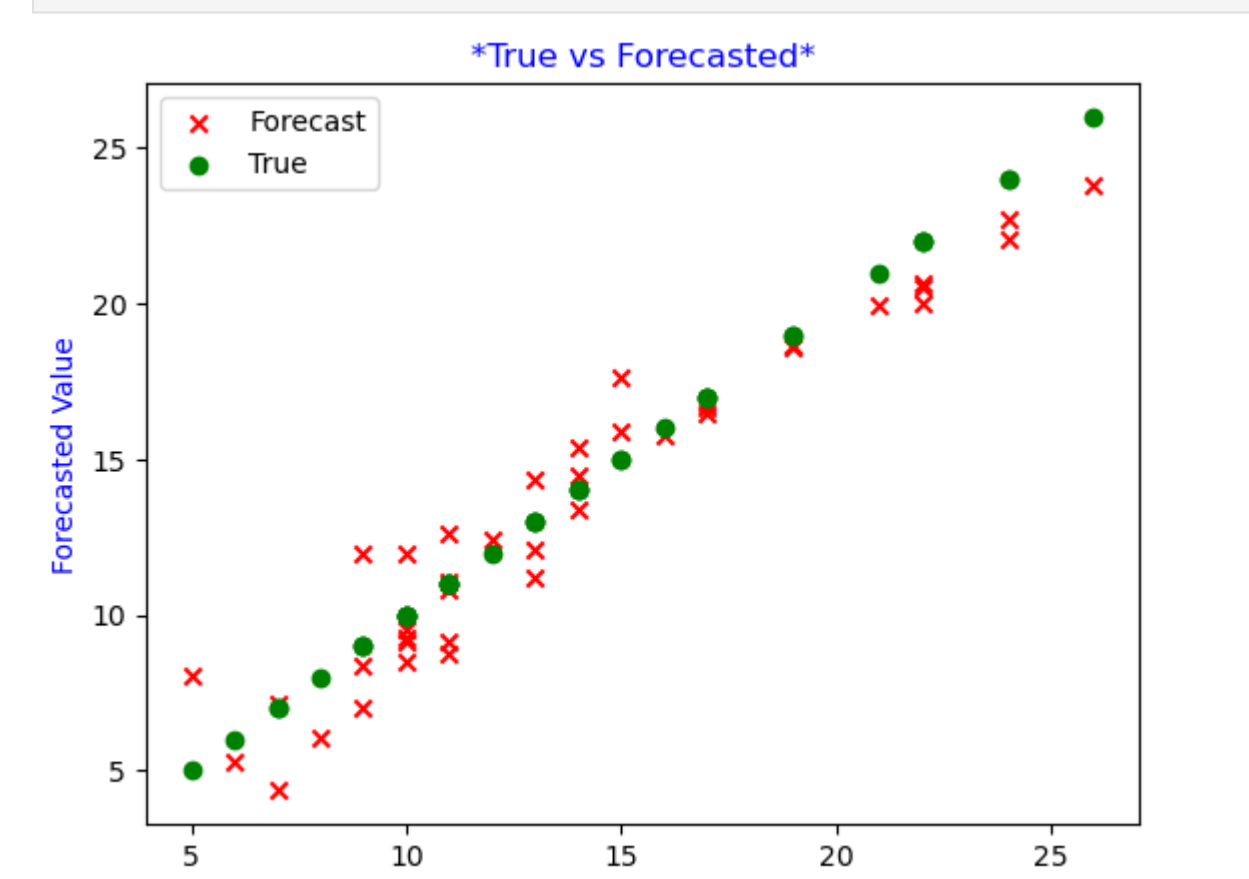


In [54]:
```
#visualization of ytest and ytest
plt.scatter(ytest,ytest,c='g')
```

Out[54]:
```
<matplotlib.collections.PathCollection at 0x2a799ee7b20>
```



In [56]:
```
plt.scatter(ytest,y_pred,c='r',label='Forecast',marker='x')
plt.scatter(ytest,ytest,c='g',label='True',marker='o')
plt.xlabel('True Value',c='b')
plt.ylabel('Forecasted Value',c='b')
plt.title('*True vs Forecasted*',c='b')
plt.legend()
plt.show()
```



Testing the effectiveness of the regression model

In [58]:
```
m_s_e = mean_squared_error(ytest,y_pred)
m_s_e
```

Out[58]:
```
2.2399312821569315
```

In [60]:
```
m_a_e = mean_absolute_error(ytest,y_pred)
m_a_e
```

Out[60]:
```
1.2387777705237972
```

In [63]:
```
r2score = r2_score(ytest,y_pred)
print(r2score)
```

```
0.9222988021105912
```

Diagnosis : Regression models with lower values are more accurate !!