Importing Librarires like numpy and python

PROTONU GHOSH

- IBM AICTE MAJOR PROJECT FINAL CODE
- student id- STU62a5e631eb60e1655039537

```
import numpy as np
import pandas as pd
```

```
dataset = pd.read_csv('/content/SampleSuperstore.csv')
```

```
dataset.head()
```

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	261.9600	2	0.00	41.9136
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.9400	3	0.00	219.5820
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	14.6200	2	0.00	6.8714
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	957.5775	5	0.45	-383.0310
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	22.3680	2	0.20	2.5164

```
dataset.shape
```

```
(9994, 13)
```

```
dataset.columns
```

```
Index(['Ship Mode', 'Segment', 'Country', 'City', 'State', 'Postal Code',
       'Region', 'Category', 'Sub-Category', 'Sales', 'Quantity', 'Discount',
       'Profit'],
      dtype='object')
```

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 13 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   Ship Mode     9994 non-null   object 
 1   Segment       9994 non-null   object 
 2   Country       9994 non-null   object 
 3   City          9994 non-null   object 
 4   State         9994 non-null   object 
 5   Postal Code   9994 non-null   int64  
 6   Region        9994 non-null   object 
 7   Category      9994 non-null   object 
 8   Sub-Category  9994 non-null   object 
 9   Sales         9994 non-null   float64
 10  Quantity      9994 non-null   int64  
 11  Discount      9994 non-null   float64
 12  Profit        9994 non-null   float64
dtypes: float64(3), int64(2), object(8)
memory usage: 1015.1+ KB
```

```
dataset.describe()
```

	Postal Code	Sales	Quantity	Discount	Profit
count	9994.000000	9994.000000	9994.000000	9994.000000	9994.000000
mean	55190.379428	229.858001	3.789574	0.156203	28.656896
std	32063.693350	623.245101	2.225110	0.206452	234.260108
min	1040.000000	0.444000	1.000000	0.000000	-6599.978000
25%	23223.000000	17.280000	2.000000	0.000000	1.728750
50%	56430.500000	54.490000	3.000000	0.200000	8.666500
75%	90008.000000	209.940000	5.000000	0.200000	29.364000
max	99301.000000	22638.480000	14.000000	0.800000	8399.976000

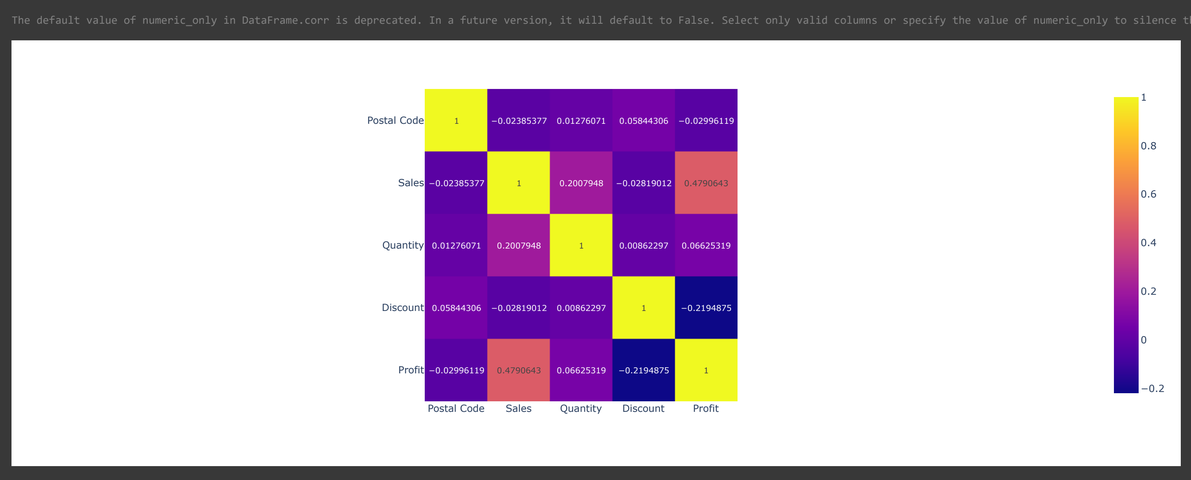
```
dataset.corr()
```

```
<ipython-input-40-c187c74d1e71>:1: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
```

	Postal Code	Sales	Quantity	Discount	Profit
Postal Code	1.000000	-0.023854	0.012761	0.058443	-0.029961
Sales	-0.023854	1.000000	0.200795	-0.028190	0.479064
Quantity	0.012761	0.200795	1.000000	0.008623	0.066253
Discount	0.058443	-0.028190	0.008623	1.000000	-0.219487
Profit	-0.029961	0.479064	0.066253	-0.219487	1.000000

```
import plotly.express as px
fig = px.imshow(dataset.corr(),text_auto=True)
fig.show()
```

```
<ipython-input-41-d3c505fa4982>:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
```




```
import seaborn as sns
ax=sns.heatmap(dataset.corr(),annot=True,cbar=True,square=True,linecolor="red",linewidths=1,xticklabels="auto",yticklabels="auto",fmt=".2f")
ax.set(xlabel="", ylabel="")
ax.xaxis.tick_top()
```

```
<ipython-input-42-eb4b826e69f5>:2: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns or specify the value of numeric_only to silence this warning.
```

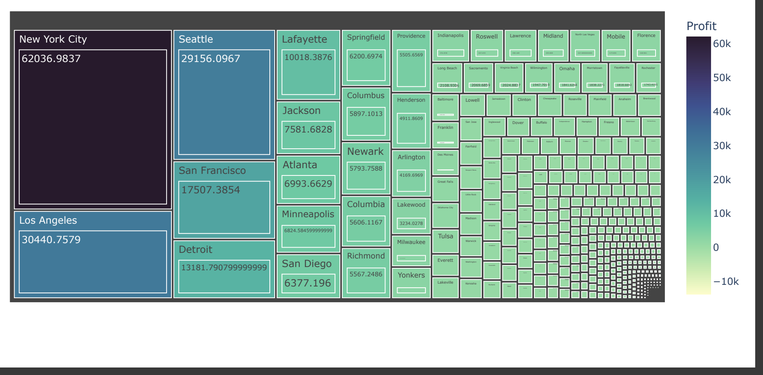

```
dataset["City"].unique()
```

```
'Spokane', 'Keller', 'Port Orange', 'Medford', 'Charlottesville',
'Missoula', 'Apopka', 'Reading', 'Broomfield', 'Paterson',
'Oklahoma City', 'Chesapeake', 'Lubbock', 'Johnson City',
'San Bernardino', 'Leominster', 'Bozeman', 'Perth Amboy',
'Ontario', 'Rancho Cucamonga', 'Moorhead', 'Mesquite', 'Stockton',
'Ormond Beach', 'Sunnyvale', 'York', 'College Station',
'Saint Louis', 'Manteca', 'San Angelo', 'Salt Lake City',
'Knoxville', 'Little Rock', 'Lincoln Park', 'Marion', 'Littleton',
'Bangor', 'Southaven', 'New Castle', 'Midland', 'Sioux Falls',
'Fort Collins', 'Clarksville', 'Sacramento', 'Thousand Oaks',
'Malden', 'Holyoke', 'Albuquerque', 'Sparks', 'Coachella',
'Elmhurst', 'Passaic', 'North Charleston', 'Newport News',
'Jamestown', 'Mishawaka', 'La Quinta', 'Tallahassee', 'Nashville',
'Bellingham', 'Woodstock', 'Haltom City', 'Wheeling',
'Summerville', 'Hot Springs', 'Englewood', 'Las Cruces', 'Hoover',
'Frisco', 'Vacaville', 'Waukesha', 'Bakersfield', 'Pompano Beach',
'Corpus Christi', 'Redondo Beach', 'Orlando', 'Orange',
'Lake Charles', 'Highland Park', 'Hempstead', 'Noblesville',
'Apple Valley', 'Mount Pleasant', 'Sterling Heights', 'Eau Claire',
'Pharr', 'Billings', 'Gresham', 'Chattanooga', 'Meridian',
'Bolingbrook', 'Maple Grove', 'Woodland', 'Missouri City',
'Pearland', 'San Mateo', 'Grand Rapids', 'Visalia',
'Overland Park', 'Temecula', 'Yucaipa', 'Revere', 'Conroe',
'Tinley Park', 'Dubuque', 'Dearborn Heights', 'Santa Fe',
'Hickory', 'Carol Stream', 'Saint Cloud', 'North Miami',
'Plantation', 'Port Saint Lucie', 'Rock Hill', 'Odessa',
'West Allis', 'Chula Vista', 'Manhattan', 'Altoona', 'Thornton',
'Champaign', 'Texarkana', 'Edinburg', 'Baytown', 'Greenwood',
'Woonsocket', 'Superior', 'Bedford', 'Covington', 'Broken Arrow',
'Miramar', 'Hollywood', 'Deer Park', 'Wichita', 'Mcallen',
'Iowa City', 'Boise', 'Cranston', 'Port Arthur', 'Citrus Heights',
'The Colony', 'Daytona Beach', 'Bullhead City', 'Portage', 'Fargo',
'Elkhart', 'San Gabriel', 'Margate', 'Sandy Springs', 'Mentor',
'Lawton', 'Hampton', 'Rome', 'La Crosse', 'Lewiston',
'Hattiesburg', 'Danville', 'Logan', 'Waterbury', 'Athens',
'Avondale', 'Marietta', 'Yuma', 'Wausau', 'Pasco', 'Oak Park',
'Pensacola', 'League City', 'Gaithersburg', 'Lehi', 'Tuscaloosa',
'Moreno Valley', 'Georgetown', 'Loveland', 'Chandler', 'Helena',
'Kirkwood', 'Waco', 'Frankfort', 'Bethlehem', 'Grand Island',
'Woodbury', 'Rogers', 'Clovis', 'Jupiter', 'Santa Barbara',
'Cedar Hill', 'Norfolk', 'Draper', 'Ann Arbor', 'La Mesa',
'Pocatello', 'Holland', 'Milford', 'Buffalo Grove', 'Lake Forest',
'Redding', 'Chico', 'Utica', 'Conway', 'Cheyenne', 'Owensboro',
'Caldwell', 'Kenner', 'Nashua', 'Bartlett', 'Redwood City',
'Lebanon', 'Santa Maria', 'Des Plaines', 'Longview',
'Hendersonville', 'Waterloo', 'Cambridge', 'Palatine', 'Beverly',
'Eugene', 'Oxnard', 'Renton', 'Glenview', 'Delray Beach',
'Commerce City', 'Texas City', 'Wilson', 'Rio Rancho', 'Goldsboro',
'Montebello', 'El Cajon', 'Beaumont', 'West Palm Beach', 'Abilene',
'Normal', 'Saint Charles', 'Camarillo', 'Hillsboro', 'Burbank',
'Modesto', 'Garden City', 'Atlantic City', 'Longmont', 'Davis',
'Morgan Hill', 'Clifton', 'Sheboygan', 'East Point', 'Rapid City',
'Andover', 'Kissimmee', 'Shelton', 'Danbury', 'Sanford',
'San Marcos', 'Greeley', 'Mansfield', 'Elyria', 'Twin Falls',
'Coral Gables', 'Romeoville', 'Marlborough', 'Laurel', 'Bryan',
'Pine Bluff', 'Aberdeen', 'Hagerstown', 'East Orange',
'Arlington Heights', 'Oswego', 'Coon Rapids', 'San Clemente',
'San Luis Obispo', 'Springdale', 'Lodi', 'Mason'], dtype=object)
```

```
cities_profit=pd.DataFrame(dataset.groupby("City")["Profit"].sum())
cities_profit.reset_index(inplace=True)
cities_profit=cities_profit.sort_values(by="Profit",ascending=False)
cities_profit.head(10)
```

	City	Profit
329	New York City	62036.9837
266	Los Angeles	30440.7579
452	Seattle	29156.0967
438	San Francisco	17507.3854
123	Detroit	13181.7908
233	Lafayette	10018.3876
215	Jackson	7581.6828
21	Atlanta	6993.6629
300	Minneapolis	6824.5846
437	San Diego	6377.1960

```
fig = px.treemap(cities_profit,
                 path=['City' ,'Profit'],
                 color_continuous_scale='deep',
                 values='Profit',color='Profit')

fig.update_layout(width=1000, height=500)
fig.show()
```



10 States by most Profit

```
dataset['State'].unique()
```

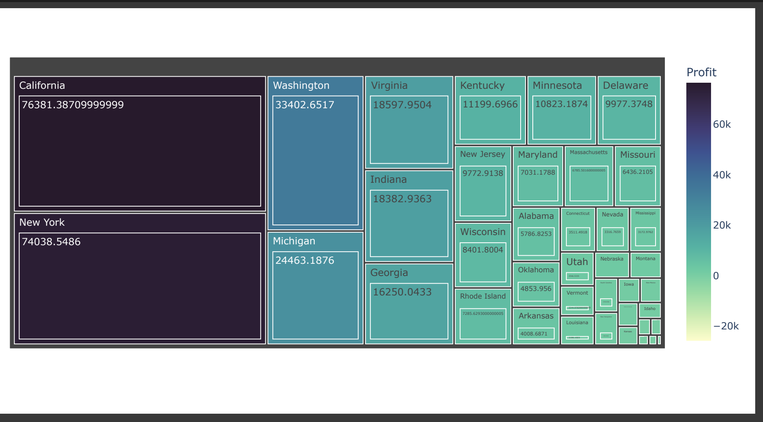
```
array(['Kentucky', 'California', 'Florida', 'North Carolina',
       'Washington', 'Texas', 'Wisconsin', 'Utah', 'Nebraska',
       'Pennsylvania', 'Illinois', 'Minnesota', 'Michigan', 'Delaware',
       'Indiana', 'New York', 'Arizona', 'Virginia', 'Tennessee',
       'Alabama', 'South Carolina', 'Oregon', 'Colorado', 'Iowa', 'Ohio',
       'Missouri', 'Oklahoma', 'New Mexico', 'Louisiana', 'Connecticut',
       'New Jersey', 'Massachusetts', 'Georgia', 'Nevada', 'Rhode Island',
       'Mississippi', 'Arkansas', 'Montana', 'New Hampshire', 'Maryland',
       'District of Columbia', 'Kansas', 'Vermont', 'Maine',
       'South Dakota', 'Idaho', 'North Dakota', 'Wyoming',
       'West Virginia'], dtype=object)
```

```
state_profit=pd.DataFrame(dataset.groupby("State")["Profit"].sum())
state_profit.reset_index(inplace=True)
```

```
state_profit=state_profit.sort_values(by="Profit",ascending=False)
state_profit.head(10)
```

	State	Profit
3	California	76381.3871
30	New York	74038.5486
45	Washington	33402.6517
20	Michigan	24463.1876
44	Virginia	18597.9504
12	Indiana	18382.9363
9	Georgia	16250.0433
15	Kentucky	11199.6966
21	Minnesota	10823.1874
6	Delaware	9977.3748

```
fig=px.treemap(state_profit,path=["State","Profit"],color_continuous_scale="deep",
               values="Profit",color="Profit")
fig.update_layout(width=1000,height=500)
fig.show()
```



Most Profitable Sub Category

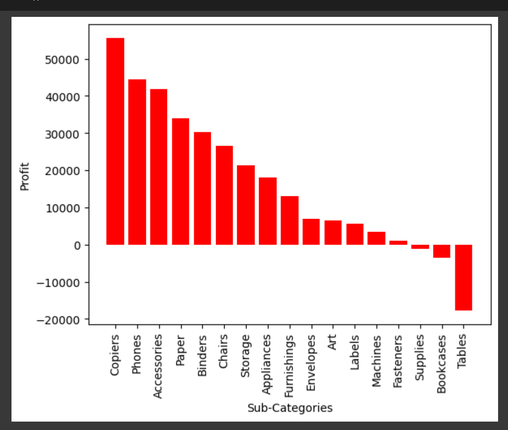
```
dataset["Sub-Category"].unique()
```

```
array(['Bookcases', 'Chairs', 'Labels', 'Tables', 'Storage',
       'Furnishings', 'Art', 'Phones', 'Binders', 'Appliances', 'Paper',
       'Accessories', 'Envelopes', 'Fasteners', 'Supplies', 'Machines',
       'Copiers'], dtype=object)
```

```
sub_categories_info=pd.DataFrame(dataset.groupby("Sub-Category")["Profit"].sum())
sub_categories_info.reset_index(inplace=True)
sub_categories_profit=sub_categories_info.sort_values(by="Profit",ascending=False)
sub_categories_profit.head(10)
```

	Sub-Category	Profit
6	Copiers	55617.8249
13	Phones	44515.7306
0	Accessories	41936.6357
12	Paper	34053.5693
3	Binders	30221.7633
5	Chairs	26590.1663
14	Storage	21278.8264
1	Appliances	18138.0054
9	Furnishings	13059.1436
7	Envelopes	6964.1767

```
import matplotlib.pyplot as plt
plt.bar(sub_categories_profit["Sub-Category"],sub_categories_profit["Profit"],color='red')
plt.xticks(sub_categories_profit['Sub-Category'] , rotation = "vertical")
plt.xlabel("Sub-Categories")
plt.ylabel("Profit")
plt.show()
```



Category thaty sales the most

```
dataset["Category"].value_counts()
```

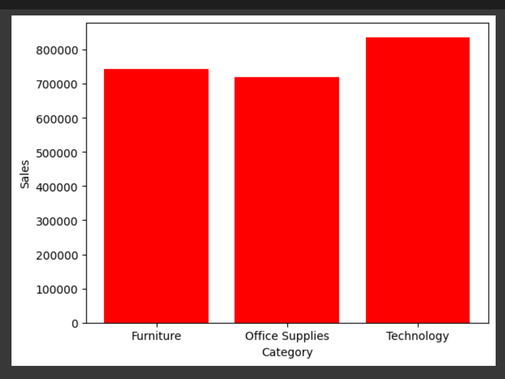
```
Office Supplies    6026
Furniture          2121
Technology         1847
Name: Category, dtype: int64
```

```
categories_sales=dataset.groupby("Category").sum()
categories_sales.reset_index(inplace=True)
categories_sales.head(10)
```

```
<ipython-input-54-07ea8d23304d>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
```

	Category	Postal Code	Sales	Quantity	Discount	Profit

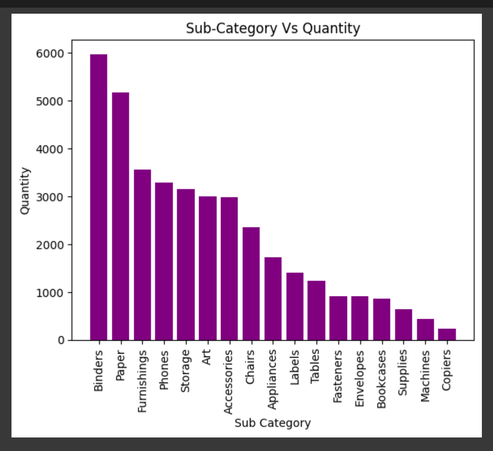
```
plt.bar(categories_sales["Category"], categories_sales['Sales'],color ='red')
plt.xticks(categories_sales.index)
plt.xlabel("Category")
plt.ylabel("Sales")
plt.show()
```



Most popular Subcategory

```
popular_sub_category=pd.DataFrame(dataset.groupby("Sub-Category")["Quantity"].sum())
popular_sub_category.reset_index(inplace=True)
popular_sub_category=popular_sub_category.sort_values(by="Quantity",ascending=False)
```

```
plt.bar(popular_sub_category["Sub-Category"],popular_sub_category["Quantity"],color="purple")
plt.xticks(popular_sub_category["Sub-Category"],rotation="vertical")
plt.title("Sub-Category Vs Quantity")
plt.xlabel("Sub Category")
plt.ylabel("Quantity")
plt.show()
```



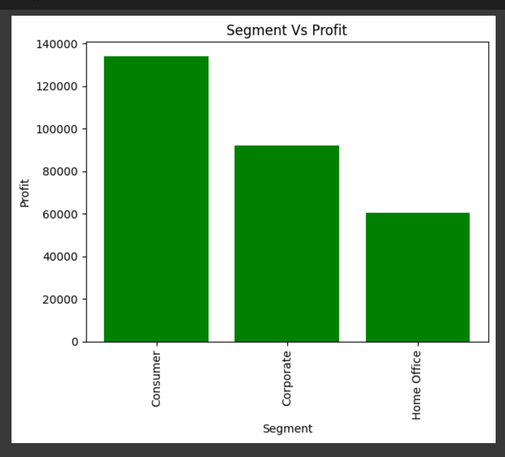
Most Profitable customer segment

```
customer_segment=dataset.groupby("Segment").sum()
customer_segment.reset_index(inplace=True)
customer_segment
```

```
<ipython-input-58-a0be90968528>:1: FutureWarning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric_only will default to False. Either specify numeric_only or select only columns which should be valid for the function.
```

	Segment	Postal Code	Sales	Quantity	Discount	Profit
0	Consumer	288878609	1.161401e+06	19521	820.91	134119.2092
1	Corporate	164536330	7.061464e+05	11608	477.85	91979.1340
2	Home Office	98157713	4.296531e+05	6744	262.33	60298.6785

```
plt.bar(customer_segment["Segment"],customer_segment["Profit"],color="green")
plt.xticks(customer_segment["Segment"],rotation="vertical")
plt.title("Segment Vs Profit")
plt.xlabel("Segment")
plt.ylabel("Profit")
plt.show()
```




```
dataset["Region"].value_counts()
```

```
West       3203
East       2848
Central    2323
South      1620
Name: Region, dtype: int64
```

```
profitable_region=pd.DataFrame(dataset.groupby("Region")["Profit"].sum())
profitable_region.reset_index(inplace=True)
profitable_region
```

	Region	Profit
0	Central	39706.3625
1	East	91522.7800
2	South	46749.4303
3	West	108418.4489

```
city_volume = pd.DataFrame(dataset.groupby('City')['Sales'].sum())
city_volume =city_volume.sort_values(by="Sales",ascending=False)
city_volume.head()
```

City	Sales
New York City	256368.161
Los Angeles	175851.341
Seattle	119540.742
San Francisco	112669.092
Philadelphia	109077.013

```
plt.bar(city_volume.index[0:20], city_volume['Sales'][0:20],color="gray")
plt.xticks(city_volume.index[0:20],rotation = 'vertical')
plt.xlabel("City" )
plt.ylabel("Sales")
plt.show()
```


MACHINE LEARNING MODEL

```
dataset.head()
```

	Ship Mode	Segment	Country	City	State	Postal Code	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Bookcases	261.9600	2	0.00	41.9136
1	Second Class	Consumer	United States	Henderson	Kentucky	42420	South	Furniture	Chairs	731.9400	3	0.00	219.5820
2	Second Class	Corporate	United States	Los Angeles	California	90036	West	Office Supplies	Labels	14.6200	2	0.00	6.8714
3	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Furniture	Tables	957.5775	5	0.45	-383.0310
4	Standard Class	Consumer	United States	Fort Lauderdale	Florida	33311	South	Office Supplies	Storage	22.3680	2	0.20	2.5164

```
dataset1=dataset.drop(columns=["Country","City","State","Sub-Category","Postal Code"])
```

```
dataset1.head()
```

	Ship Mode	Segment	Region	Category	Sales	Quantity	Discount	Profit
0	1	0	3	1	261.9600	2	0.00	41.9136
1	1	0	3	1	731.9400	3	0.00	219.5820
2	1	1	0	0	14.6200	2	0.00	6.8714
3	0	0	3	1	957.5775	5	0.45	-383.0310
4	0	0	3	0	22.3680	2	0.20	2.5164

```
dataset1.shape
```

```
(9994, 8)
```

```
dataset1["Ship Mode"].value_counts()
```

```
0    5968
1    1945
2    1538
3     543
Name: Ship Mode, dtype: int64
```

```
dataset["Category"].value_counts()
```

```
Office Supplies    6026
Furniture          2121
Technology         1847
Name: Category, dtype: int64
```

```
dataset1["Region"].value_counts()
```

```
0    3203
1    2848
2    2323
3    1620
Name: Region, dtype: int64
```

```
dataset1.replace({"Ship Mode":{"Standard Class":0,"Second Class":1,"First Class":2,"Same Day":3},"Segment":{"Consumer":0,"Corporate":1,"Home Office":2},
                  "Category":{"Office Supplies":0,"Furniture":1,"Technology":2},"Region":{"West":0,"East":1,"Central":2,"South":3}},inplace=True)
```

```
dataset1.head()
```

	Ship Mode	Segment	Region	Category	Sales	Quantity	Discount	Profit
0	1	0	3	1	261.9600	2	0.00	41.9136
1	1	0	3	1	731.9400	3	0.00	219.5820
2	1	1	0	0	14.6200	2	0.00	6.8714
3	0	0	3	1	957.5775	5	0.45	-383.0310
4	0	0	3	0	22.3680	2	0.20	2.5164

```
x=dataset1.iloc[:,:-1]
print(x)
```

```
      Ship Mode  Segment  Region  Category     Sales  Quantity  Discount
0             1        0       3         1  261.9600         2      0.00
1             1        0       3         1  731.9400         3      0.00
2             1        1       0         0   14.6200         2      0.00
3             0        0       3         1  957.5775         5      0.45
4             0        0       3         0   22.3680         2      0.20
...         ...      ...     ...       ...       ...       ...       ...
9989          1        0       3         1   25.2480         3      0.20
9990          0        0       0         1   91.9600         2      0.00
9991          0        0       0         2  258.5760         2      0.20
9992          0        0       0         0   29.6000         4      0.00
9993          1        0       0         0  243.1600         2      0.00

[9994 rows x 7 columns]
```

```
y=dataset1.iloc[:,-1]
print(y)
```

```
0        41.9136
1       219.5820
2         6.8714
3      -383.0310
4         2.5164
          ...
9989      4.1028
9990     15.6332
9991     19.3932
9992     13.3200
9993     72.9480
Name: Profit, Length: 9994, dtype: float64
```

```
from sklearn.preprocessing import MinMaxScaler
scaler=MinMaxScaler()
```

```
num_vars=["Sales","Quantity"]
x[num_vars]=scaler.fit_transform(x[num_vars])
print(x)
```

```
      Ship Mode  Segment  Region  Category     Sales  Quantity  Discount
0             1        0       3         1  0.011552  0.076923      0.00
1             1        0       3         1  0.032313  0.153846      0.00
2             1        1       0         0  0.000626  0.076923      0.00
3             0        0       3         1  0.042280  0.307692      0.45
4             0        0       3         0  0.000968  0.076923      0.20
...         ...      ...     ...       ...       ...       ...       ...
9989          1        0       3         1  0.001096  0.153846      0.20
9990          0        0       0         1  0.004043  0.076923      0.00
9991          0        0       0         2  0.011403  0.076923      0.20
9992          0        0       0         0  0.001288  0.230769      0.00
9993          1        0       0         0  0.010722  0.076923      0.00

[9994 rows x 7 columns]
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.25,random_state=100)
print(x_train.shape,x_test.shape,y_train.shape,y_test.shape)
from sklearn.ensemble import RandomForestRegressor
model=RandomForestRegressor(random_state=0,n_estimators=117)
```

```
(7495, 7) (2499, 7) (7495,) (2499,)
```

```
model=RandomForestRegressor(random_state=0,n_estimators=117)
model.fit(x_train,y_train)
```

```
RandomForestRegressor(n_estimators=117, random_state=0)
```

```
y_pred=model.predict(x_test)
```

```
from sklearn.metrics import r2_score
print("R2 Score=",r2_score(y_pred,y_test)*100)
```

```
R2 Score= 74.08625172960967
```

```
ig = plt.figure()
plt.scatter(y_test,y_pred)

plt.xlabel('y_test', fontsize=18)
plt.ylabel('Y_pred', fontsize=16)
```

```
Text(0, 0.5, 'Y_pred')
```