**#import libraries**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import statsmodels.api as sm

from pandas import ExcelWriter

from pandas import ExcelFile

from datetime import timedelta

**#Read files**

orders=pd.read\_excel('C:/Users/gurdd/Desktop/python/global\_superstore\_2016.xlsx',sheet\_name='Orders')

orders2=pd.read\_excel('C:/Users/gurdd/Desktop/python/global\_superstore\_2016.xlsx',sheet\_name='Orders2')

returns=pd.read\_excel('C:/Users/gurdd/Desktop/python/global\_superstore\_2016.xlsx',sheet\_name='Returns')

People=pd.read\_excel('C:/Users/gurdd/Desktop/python/global\_superstore\_2016.xlsx',sheet\_name='People')

**#Remove Duplicates**

orders2=orders2.drop\_duplicates()

**#Concatenate datasets**

combined\_orders=pd.concat([orders,orders2])

**#joining orders with People and Return dataset**

combined\_orders = combined\_orders.merge(People,how='left')

combined\_orders = combined\_orders.merge(returns,how='left')

**#Data Cleaning and preparation**

#SunString - Extracting the first two letters of column CustomerID

combined\_orders['St']=combined\_orders['Customer ID'].str.slice(start=0,stop=2,step=None)

**#Split the Customer Name into First, Middle and Last name**

combined\_orders[['FirstName', 'LastName', 'MiddleName']] = combined\_orders['Customer Name'].str.split(' ', expand=True)

**#Extract the year from the Date column**

combined\_orders['Year']=combined\_orders['Order Date'].dt.year

**#Adding the days to OrderDate Column**

combined\_orders["OrderDate\_Plus\_2"] = combined\_orders["Order Date"] + timedelta(days=2)

**#Getting the difference in dates between two date columns**

combined\_orders["DateDiff"]= combined\_orders['Order Date']-combined\_orders['Ship Date']

**#Concatenate two fields**

combined\_orders["Cat\_Sub\_Cat"]=combined\_orders['Category']+' - '+combined\_orders['Sub-Category']

**#Define a function and create a flag**

def func(row):

if row['Profit'] <= 0 and row['Profit'] == 'Critical':

return 'A'

elif row['Profit'] <= 0 and row['Profit'] == 'High':

return 'B'

else:

return 'C'

combined\_orders['Flag'] = combined\_orders.apply(func, axis=1)

**#Create a dataset after grouping by year**

combined\_orders\_grouped = combined\_orders.groupby('Year')

**#Aggregating the sum and mean of profit**

combined\_orders\_grouped["Profit"].agg([np.sum,np.mean])

**#Create a dataset after grouping by Product Name**

combined\_orders\_grouped2 = combined\_orders.groupby('Product Name')

**#Aggregating the sum of profit**

combined\_orders\_grouped3 = combined\_orders\_grouped2["Profit"].agg([np.sum])

combined\_orders\_grouped3

**#Order by sum in the decending order and select only the first 10 records**

top\_products = combined\_orders\_grouped3.sort\_values(by='sum', ascending=False).head(10)

**#Use the lerad and lag function to move the value of sum**

top\_products['SumLag'] = top\_products['sum'].shift(1)

top\_products['SumLead'] = top\_products['sum'].shift(-1)

**#Calculate the percentage change in value of the one product with other**

top\_products['Change']= ((top\_products['sum']- top\_products['SumLead'])/top\_products['sum'])\*100

**#VISUALIZATION**

**#Pie-Chart**

colors = ['yellowgreen', 'gold', 'lightskyblue', 'lightcoral','red','green','blue','orange','white','brown']

combined\_orders['Segment'].value\_counts().plot(kind='pie',title='Superstore by Segment',colors=colors)

plt.show()

**#scatter plot 1**

combined\_orders.plot(kind='scatter', x='Sales' ,y='Profit')

plt.show()

**#scatter plot 2**

plt.scatter(combined\_orders['Sales'],combined\_orders['Profit'],color='red',label='ScatterPlot')

plt.xlabel("Sales")

plt.ylabel("Profit")

plt.title("Profit vs Sales")

plt.xlim(0,15000)

plt.ylim(-4000,4000)

plt.legend()

plt.show()

**#line plot 1**

combined\_orders.plot.line(x='Order Date', y='Sales', figsize=(12,3), lw=1)

plt.show()

**#line plot 2**

plt.plot(combined\_orders['Sales'],color='blue',label='line')

plt.xlabel("Sales")

plt.ylabel("Profit")

plt.title("Profit vs Sales")

plt.xlim(0,15000)

plt.ylim(0,20000)

plt.legend()

plt.show()

**#line plot 3**

data=combined\_orders.head(50)

plt.plot(data['Sales'],data['Profit'],color='red',label='Line-graph')

plt.scatter(data['Sales'],data['Profit'],color='blue',label='Scatter')

plt.legend()

plt.xlabel('Sales', color='green')

plt.ylabel('Profit', color='brown')

plt.show()

**#horizontal bar-graph**

data=combined\_orders.head(10000)

plt.barh(data['Sales'],data['Profit'],color=['blue','green','yellow'],label='Bar')

plt.legend()

plt.xlabel('Sales', color='green')

plt.ylabel('Profit', color='brown')

plt.show()

**#bar-graph**

data=combined\_orders.head(10000)

plt.bar(data['Sales'],data['Profit'],color='blue',label='Bar')

plt.legend()

plt.xlabel('Sales', color='green')

plt.ylabel('Profit', color='brown')

plt.show()

**#Figure and sub-plot**

data1=combined\_orders.head(25)

**#figure**

fig=plt.figure()

fig.patch.set\_facecolor('gray')

**#(row, column, position)**

graph1=fig.add\_subplot(2,2,1, facecolor='yellow')

graph1.plot(data['Sales'],data['Profit'])

graph1.set\_title('Plot: Sales vs Profit')

graph2=fig.add\_subplot(2,2,2, facecolor='lightgreen')

graph2.scatter(data['Sales'],data['Profit'])

graph2.set\_title('Scatter : Sales vs Profit')

graph3=fig.add\_subplot(2,1,2, facecolor='orange')

graph3.scatter(data['Sales'],data['Profit'])

graph3.set\_title('Scatter : Sales vs Profit')

plt.show()

**#Figure and sub-plot**

data1=combined\_orders.head(25)

**#figure**

fig=plt.figure()

fig.patch.set\_facecolor('gray')

#(row, column, position)

graph1=fig.add\_subplot(2,2,1, facecolor='yellow')

graph1.plot(data['Sales'],data['Profit'])

graph1.set\_title('Plot: Sales vs Profit')

graph2=fig.add\_subplot(2,2,2, facecolor='lightgreen')

graph2.scatter(data['Sales'],data['Profit'])

graph2.set\_title('Scatter : Sales vs Profit')

graph3=fig.add\_subplot(2,1,2, facecolor='orange')

d1=len(data[data.Year == 2013])

d2=len(data[data.Year == 2014])

d3=len(data[data.Year == 2015])

graph3.pie([d1,d2,d3],colors=["yellow","blue","green"],labels=['First','Second','Same Day'])

graph3.legend(title='Description')

plt.show()

**#Regression**

**#Import libraries**

from sklearn.model\_selection import train\_test\_split

import numpy as np

import matplotlib.pyplot as pt

from sklearn import linear\_model

import pandas as pd

import math

**#Divide the dataset into training and test dataset**

train, test = train\_test\_split(combined\_orders, test\_size=0.2)

**#Creating the predictor and response variables**

X=train[['Sales']].as\_matrix()

Y=train[['Profit']].as\_matrix()

Xtest=test[['Sales']].as\_matrix()

Ytest=test[['Profit']].as\_matrix()

np.mean(X)

np.median(X)

np.mean(Y)

np.median(Y)

**#Plot a scatter plot**

pt.scatter(X,Y,s=5,c='black',marker='\*')

pt.show()

**#Linear regression**

#plot a graph between training values to get the idea of the linear relationship between X and Y

lm=linear\_model.LinearRegression()

lm.fit(X,Y)

**#calculate the coefficient of determination and correlation coefficient to get the measures the association between**

**#2 var(correlation coeff.) and the extent to which dependent variables can be predicted from the indpndnt var(coeff**

**#of determination.)**

print('Coeff of determination:',lm.score(X,Y))

print('correlation is:',math.sqrt(lm.score(X,Y)))

p=lm.predict(X)

pt.title('Scatter between predicted values and actual values in training set')

pt.scatter(Y,p,s=5)

pt.xlabel('actual value')

pt.ylabel('predicted value')

pt.show()

pr=lm.predict(Xtest)

pt.title('plot between actual values and predicted values in the test set')

pt.scatter(Ytest,pr,s=9,c='cyan')

pt.xlabel('test values')

pt.ylabel('predicted values')

pt.show()