CAPSTONE PROJECT REPORT

SALES PREDICTION

PGPDSE – FT Bangalore April 22

Submitted by:- Group 2

1. Krishan Kumar
2. Vaibhav Mehta
3. Ankit Jhajharia
4. Sheikh Aqeeb
5. Ravi Ranjan

Mentored By:-

Mr. Animesh Tiwari

CONTENTS

ABSTRACT

Superstore Systems give your brand the platform to launch into the fast-emerging marketplace arena. The B2C marketplace accounts for more than 53% of online sales in 2018. Your own marketplace gives you ownership of the complete customer journey together with all the data. The $7 trillion B2B market is ripe for disruption as Amazon have shown with their rise in the B2B sector from $1B sales in 2016 to $10B in 2018.

The Covid-19 pandemic has changed the buying behaviour of the consumers, there has been significant decline in the number of customers visiting to supermarkets, hypermarkets. Almost a year since the nationwide lockdown was announced due to Covid -19 the panic buying of daily essentials was at surge. The online players struggled to meet the demands of the customers and local kiranas appeared to be the saviours. With shift due to pandemic the store owners have adopted the new and latest technologies to meet the expectation of customers. Customers got accustomed to the new way of shopping for daily needs and store owner have fast-tracked the push towards digital transformation which impacted the sales of supermarkets immensely.

The main business objective or the challenge faced by the hypermarkets in such precedented times is how to minimize the operational expenses and caters the need of the business stability. What should be done to face the challenges aroused due to declining sales because of the pandemic. What should be the steps to be taken to increase the sales.

Key Words- Back Order, Machine Learning, Prediction, Supply Chain Management

1.1 Industry Review - Current practices, Background Research

The impact of the COVID-19 outbreak on small businesses has been staggering. All across the county, many businesses have had to reduce their hours and their staff. Some businesses have shut down their operations altogether in an attempt to help stop the spread, seeing huge drop-offs in profits and creating overwhelming uncertainty for their owners and employees.

Shopping online is currently the need of the hour. Because of this COVID, it's not easy to walk in a store randomly and buy anything you want. I this I am trying to understand a few things like

- **Customers Analysis**
  1. Profile the customers based on their frequency of purchase - calculate frequency of purchase for each customer
  2. Do the high frequent customers are contributing more revenue
  3. Are they also profitable - what is the profit margin across the buckets
  4. Which customer segment is most profitable in each year.
  5. How the customers are distributed across the countries- -

- **Product Analysis**
  1. Which country has top sales?
  2. Which are the top 5 profit-making product types on a yearly basis
  3. How is the product price varying with sales - Is there any increase in sales with the decrease in price at a day level?
  4. What is the average delivery time across the counties - bar plot

1.2 Literature Survey - Publications, Application, past and undergoing research

Machine Learning (ML) techniques enable us to forecast accurately multiple aspects related to supply chain management such as demand, sale, revenue, production, and backorder. ML approaches have been used to predict manufacturers' garbled demands where some researchers applied a representative set of ML-based and traditional forecasting methods to the data to compare the precision of those used methods (Carbonneau R, Vahidov R & Laframboise K, 2007).

An analysis of the supply chain's demand prediction was carried out by applying the SupportVector Regression (SVR) method in the paper of (Guanghui, 2012).

To minimize the supply chain and inventory control costs, a risk-based dynamic backorder replenishment planning framework was proposed by applying the Bayesian Belief Network (ShinK, Shin Y, Kwon JH, Kang SH, 2012).

The prediction of uncertainty in an inventory model, and proposed a framework to estimate thedemand to obtain more accurate inventory decisions (Prak D & Teunter R, 2019). Machine learning to predict and optimize product backorders using a stack-ensemble machine learning approach. The author also discussed the cost–benefit of early prediction of the backorder(Dancho M, 2017).

A Machine Learning framework for customer purchase evaluated the performance of ML modelssuch as Lasso, Extreme learning machine, and Gradient tree boosting to forecast future purchase trends (Martínez A, Schmuck C, Pereverzyev S Jr, Pirker C & Haltmeier M, 2020).

The efficiency and impact of different types of forecasting methods were measured for promotional products in business in the research of De Baets and Harvey, 2019.
Based on the literature review, there are some research gaps. To our knowledge, just a few researchers have explored the negative values in the supply chain data. Besides, very few papers have considered flexible inventory control and cost minimization techniques separately, but noneof them provided the probable backorder scenarios in inventory management.

2 Data set and Domain

The dataset selected by us is a ‘Global Super Store Dataset’ and we are trying to understand how is the product price varying with sales - Is there any increase in sales with the decrease in price at a day level. The domain for the project is e-commerce supply chain.

```
df=pd.read_excel("C:\\Users\\krishan\\Downloads\\archive (2)\\Global_Superstore2.xlsx")
df.head()
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	City	State	Country	Postal Code	Market	Region	Product ID	Category	
0	32298	CA-2012-124891	31-07-2012	31-07-2012	Same Day	RH-19495	Rick Hansen	Consumer	New York City	New York	United States	10024.0	US	East	TEC-AC-10003033	Technology	A
1	26341	IN-2013-77878	05-02-2013	07-02-2013	Second Class	JR-16210	Justin Ritter	Corporate	Wollongong	New South Wales	Australia	NaN	APAC	Oceania	FUR-CH-10003950	Furniture	
2	25330	IN-2013-71249	17-10-2013	18-10-2013	First Class	CR-12730	Craig Reiter	Consumer	Brisbane	Queensland	Australia	NaN	APAC	Oceania	TEC-PH-10004664	Technology	
3	13524	ES-2013-1579342	28-01-2013	30-01-2013	First Class	KM-16375	Katherine Murray	Home Office	Berlin	Berlin	Germany	NaN	EU	Central	TEC-PH-10004583	Technology	
4	47221	SG-2013-4320	05-11-2013	06-11-2013	Same Day	RH-9495	Rick Hansen	Consumer	Dakar	Dakar	Senegal	NaN	Africa	Africa	TEC-SHA-10000501	Technology	

2.1 Basis Statistics of Data.

- The dataset contains 51290 rows and 24 columns.

```
# Checking the Shape of the Data.
df.shape
```

```
(51290, 24)
```

2.2 Data Dictionary

```
columns=df.columns
print("The Features of the Data-Set are:-")
for i,j in enumerate(columns):
    print(i+1,".",j)
```

```
The Features of the Data-Set are:-
1 . Row ID
2 . Order ID
3 . Order Date
4 . Ship Date
5 . Ship Mode
6 . Customer ID
7 . Customer Name
8 . Segment
9 . City
10 . State
11 . Country
12 . Postal Code
13 . Market
14 . Region
15 . Product ID
16 . Category
17 . Sub-Category
18 . Product Name
19 . Sales
20 . Quantity
21 . Discount
22 . Profit
23 . Shipping Cost
24 . Order Priority
```

2.3 Feature description

S. No	Feature	Description
1	Row ID	Id of the row
2	Order ID	Id of the order
3	Order Date	Date of the order
4	Ship Date	Date of the shipping
5	Ship Mode	Mode of the shipping
6	Customer ID	Id of the customer
7	Customer Name	Name of the customer
8	Segment	Type of the customer
9	City	City the customer belongs to
10	State	State the customer belongs to
11	Country	Country of the customer
12	Postal Code	Postal code of the customer
13	Market	Market to which the customer belongs
14	Region	Region of the customer
15	Product ID	Id of the product
16	Category	Product category
17	Sub-Category	Sub-category of the product
18	Product Name	Name of the product
19	Sales	Total sales of the product
20	Quantity	Total quantity sold
21	Discount	Discount offered on the product
22	Profit	Total profit generated from the sales
23	Shipping Cost	Total shipping cost of the product
24	Order Priority	Priority level of the shipment

3.Variable Categorization

Classification of variables into Categorical variables and Numerical variables.

```
df_cat=df.select_dtypes(include="object")
df_cat.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Order ID        51290 non-null  object
 1   Order Date      51290 non-null  object
 2   Ship Date       51290 non-null  object
 3   Ship Mode       51290 non-null  object
 4   Customer ID     51290 non-null  object
 5   Customer Name   51290 non-null  object
 6   Segment         51290 non-null  object
 7   City            51290 non-null  object
 8   State           51290 non-null  object
 9   Country         51290 non-null  object
 10  Market          51290 non-null  object
 11  Region          51290 non-null  object
 12  Product ID      51290 non-null  object
 13  Category        51290 non-null  object
 14  Sub-Category    51290 non-null  object
 15  Product Name    51290 non-null  object
 16  Order Priority  51290 non-null  object
dtypes: object(17)
memory usage: 6.7+ MB
```

Description: -

1. There are 16 categorical variables.
2. There are 6 numerical variables in the dataset.
3. There are no null Values present in the Categorical Features.
4. There are 9994 non-null values present in Postal Code in Numerical features.

```
df_num=df.select_dtypes(exclude="object")
df_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 51290 entries, 0 to 51289
Data columns (total 7 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Row ID         51290 non-null  int64
 1   Postal Code    9994 non-null   float64
 2   Sales          51290 non-null  float64
 3   Quantity       51290 non-null  int64
 4   Discount       51290 non-null  float64
 5   Profit         51290 non-null  float64
 6   Shipping Cost  51290 non-null  float64
dtypes: float64(5), int64(2)
memory usage: 2.7 MB
```

4.Pre-Processing Data Analysis (count of missing/ null values, redundantcolumns, etc.)

As earlier we have seen there were missing values present in the numerical feature postal code. Now We will see how to check for missing values and what can be done in our current scenario:-

```
# Checking for the missing values in the data.
```

```
df.isnull().sum()
```

```
Row ID            0
Order ID          0
Order Date        0
Ship Date         0
Ship Mode         0
Customer ID       0
Customer Name     0
Segment           0
City              0
State             0
Country           0
Postal Code       41296
Market            0
Region            0
Product ID        0
Category          0
Sub-Category      0
Product Name      0
Sales             0
Quantity          0
Discount          0
Profit            0
Shipping Cost     0
Order Priority    0
dtype: int64
```

Description: -

1. There are 41296 missing values present in the feature postal code.
2. As per the requirement of current data set we do not require postal code for our studies so we will drop the same.

```
# Dropping the feature called postal code.
```

```
df1=df.drop("Postal Code",axis=1)
```

```
# We have dropped the feature and made a new data frame with the remaining features i.e.df1.
```

```
df1.isnull().sum()
```

```
Row ID            0
Order ID          0
Order Date        0
Ship Date         0
Ship Mode         0
Customer ID       0
Customer Name     0
Segment           0
City              0
State             0
Country           0
Market            0
Region            0
Product ID        0
Category          0
Sub-Category      0
Product Name      0
Sales             0
Quantity          0
Discount          0
Profit            0
Shipping Cost     0
Order Priority    0
dtype: int64
```

4.1 Statistics about the numerical features in the dataset

```
1 # Statistical Summary of the numerical features of the data.
```

```
1 df1.select_dtypes(exclude="object").describe().T
```

	count	mean	std	min	25%	50%	75%	max
Sales	25035.0	378.521238	615.461671	0.556	52.8435	159.732	447.94400	17499.950
Quantity	25035.0	3.810945	2.414887	1.000	2.0000	3.000	5.00000	14.000
Discount	25035.0	0.141982	0.207388	0.000	0.0000	0.000	0.20000	0.850
Profit	25035.0	43.535601	235.581331	-6599.978	0.0000	14.976	64.68084	8399.976
Shipping Cost	25035.0	42.202011	75.751269	0.000	4.9200	15.580	45.45000	933.570

Insights: -

1. There are Outliers in the features.
2. Through five-point summary we get to know the count, mean, std, min, max values .

3. Sales: - The Average sale amount of an order is 378 whereas min amount for an order is 0.55 and max is 17500.
4. Quantity: - The Average quantity ordered single time is 3.8 whereas min is 1 and max is 14 for a order.
5. Discount: - The Average Discount offered is 0.14 whereas the min is 0 and max discount is 0.85
6. Profit: - The Average profit for a order is 43.5 whereas min is -6559 indicating that the order is being cancelled whereas max is 8399.
7. Shipping Cost: - The Average shipping cost is 42 dollars whereas min is 0 and max is 933.

4.1 Statistics about the categorical features in the dataset.

```
# Statistical Summary of the categorical features of the data.
```

```
df1.select_dtypes(include="object").describe().T
```

	count	unique	top	freq
Order ID	25035	25035	CA-2012-124891	1
Order Date	25035	1428	18-06-2014	59
Ship Date	25035	1464	22-11-2014	54
Ship Mode	25035	4	Standard Class	14953
Customer ID	25035	1589	PO-18850	40
Customer Name	25035	795	Laura Armstrong	46
Segment	25035	3	Consumer	12916
City	25035	3601	New York City	450
State	25035	1094	California	1019
Country	25035	147	United States	4991
Market	25035	7	APAC	5437
Region	25035	13	Central	5237
Product ID	25035	8815	FUR-CH-10002647	16
Category	25035	3	Office Supplies	12627
Sub-Category	25035	17	Storage	2649
Product Name	25035	3601	Staples	91
Order Priority	25035	4	Medium	14281

5.Redundancy Reduction: -

Sometimes there exits duplicate records in the data. So, we will check if any such records exist in the data. This will help to reduce the redundancy and in better analysis of data.

```
# Checking Data for duplicate records.
```

```
df1.nunique()
```

```
Row ID            51290
Order ID          25035
Order Date         1430
Ship Date          1464
Ship Mode             4
Customer ID        1590
Customer Name       795
Segment               3
City               3636
State              1094
Country             147
Market                7
Region               13
Product ID         10292
Category               3
Sub-Category          17
Product Name        3788
Sales              22995
Quantity              14
Discount              27
Profit             24575
Shipping Cost      10037
Order Priority         4
dtype: int64
```

Description: -

1.By analyzing the data we can see there are only 25035 unique orders implying there is redundancy in data.

2.This method also help us with better understanding on categorical columns about the classes or segments they are further segregated into.

3.We will treat the data reduce the duplicate values present in the data.

4.After removal of duplicate values.

```
# Checking for the duplicate order ids.
```

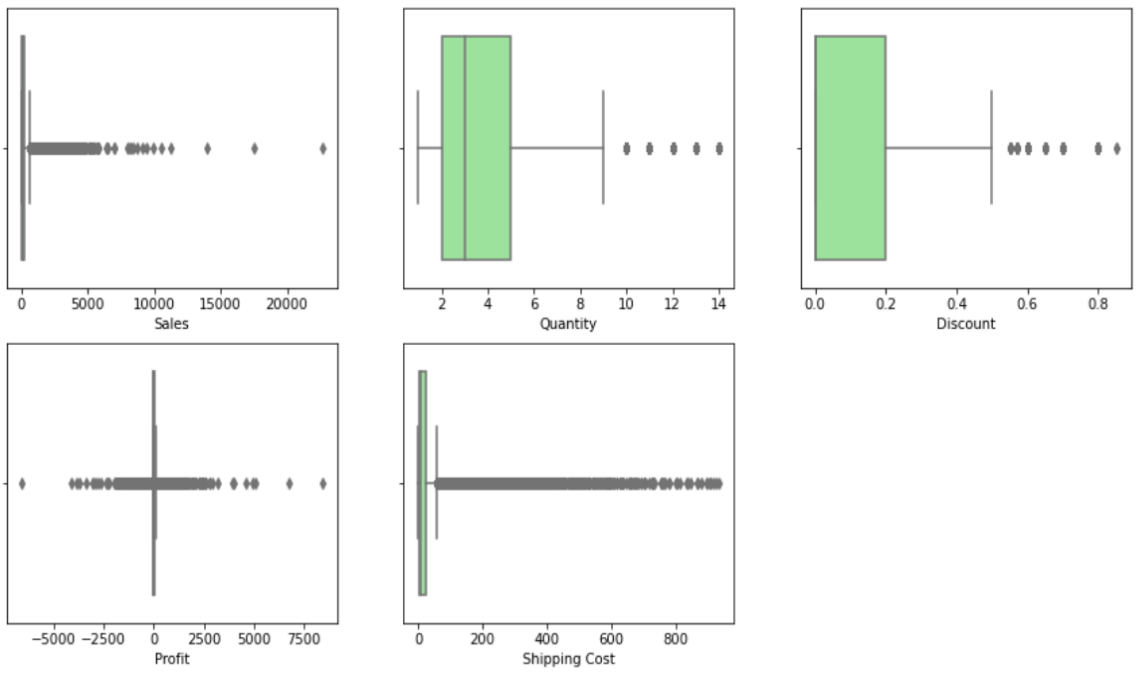
```
df1.drop_duplicates(subset="Order ID",inplace=True)
df1.shape
```

```
(25035, 23)
```

```
In [ ]: 1 # Removed the duplicate orders from the data.

In [ ]: 1 df1.nunique()

Out[ ]: Row ID           25035
        Order ID         25035
        Order Date        1428
        Ship Date         1464
        Ship Mode            4
        Customer ID       1589
        Customer Name      795
        Segment              3
        City              3601
        State             1094
        Country            147
        Market               7
        Region              13
        Product ID        8815
        Category             3
        Sub-Category        17
        Product Name      3601
        Sales            15950
        Quantity            14
        Discount            27
        Profit           15782
        Shipping Cost     9141
        Order Priority       4
        dtype: int64
```

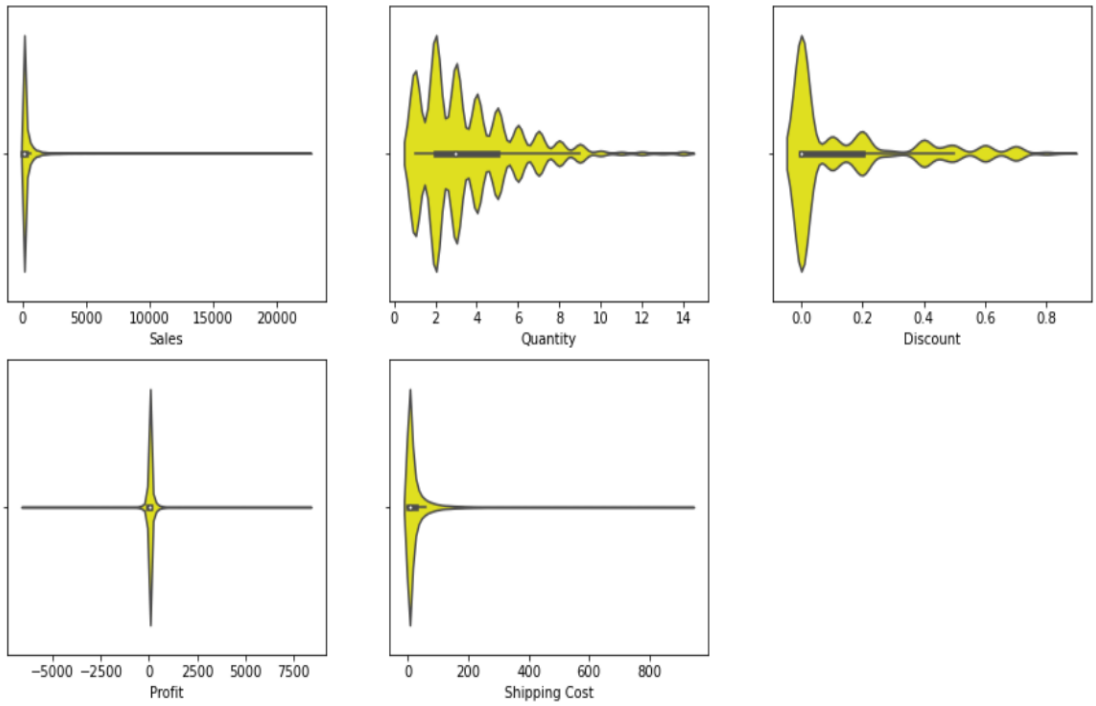
6. Data Exploration (EDA)

Visualisation of Numerical Features as whole: -



Insights:-

- Shows us there are outliers present in the data.
- Help us understand about the min and max range of the data for the high frequencies
- Help us with the values of outliers.

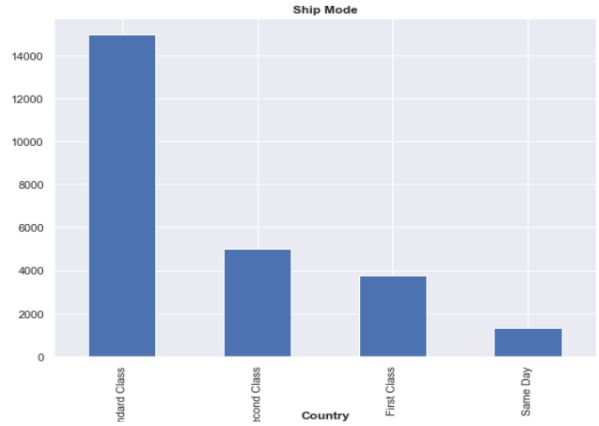


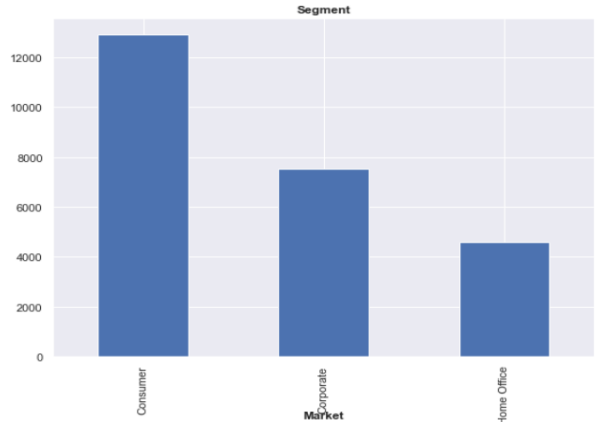
Insights: -

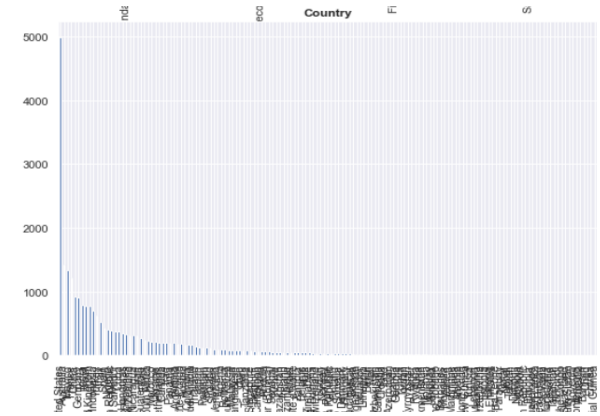
- Help us with the easy visualization of the frequency distribution of data along features w.r.t values.

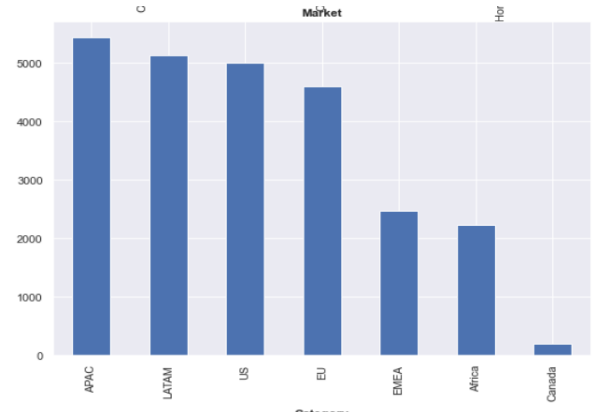
Visualization of Categorical features as whole: -

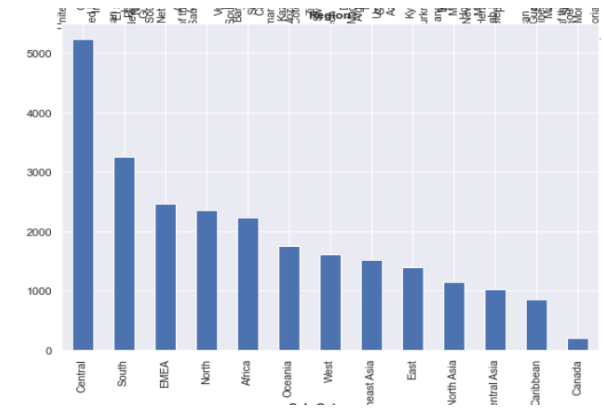
```
n=1
plt.figure(figsize=(20,30))
for i in df_cat1.columns:
    sns.set(style='darkgrid')
    plt.subplot(4,2,n)
    df_cat1[i].value_counts().sort_values(ascending=False).plot(kind="bar")
    plt.title(i,weight="bold")
    n+=1
    plt.xticks(rotation=90)
    plt.tight_layout

```

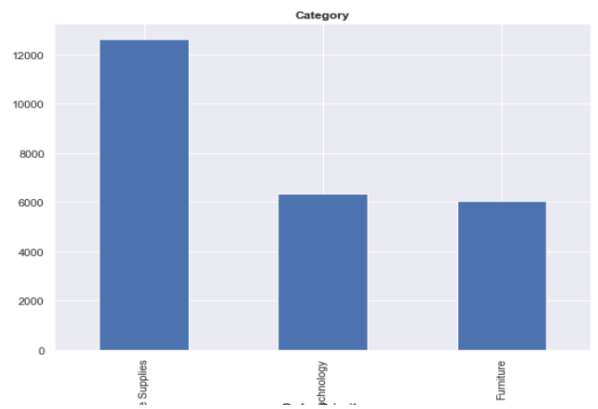

Ship Mode
14000
12000
10000
8000
6000
4000
2000
0
First Class
Same Day
Country

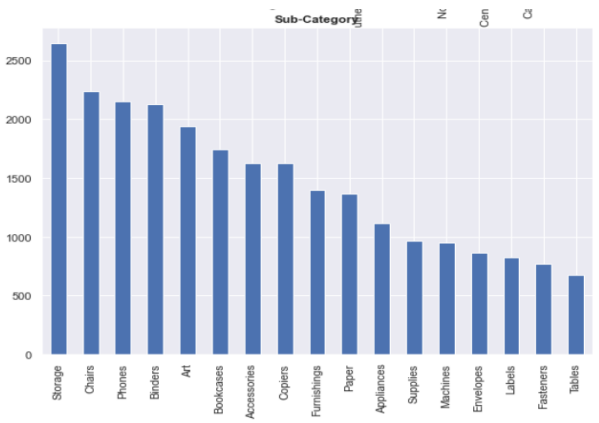

Segment
12000
10000
8000
6000
4000
2000
0
Consumer
Market

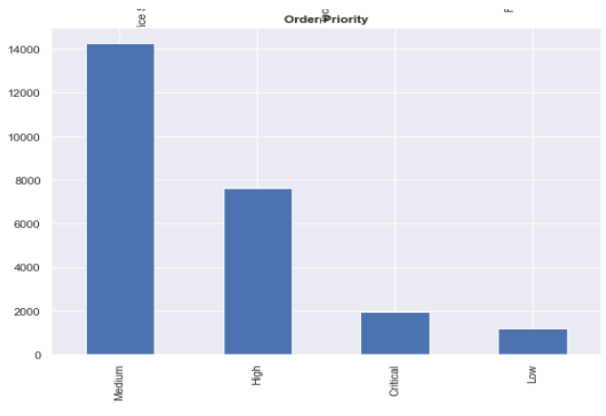

Country
5000
4000
3000
2000
1000
0


Market
5000
4000
3000
2000
1000
0
APAC
LATAM
US
EU
EMEA
Africa
Canada


5000
4000
3000
2000
1000
0
Central
South
EMEA
North
Africa
Oceania
West
East
Canada

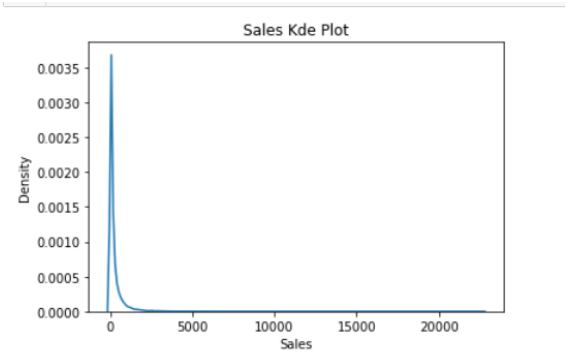

Category
12000
10000
8000
6000
4000
2000
0
Furniture

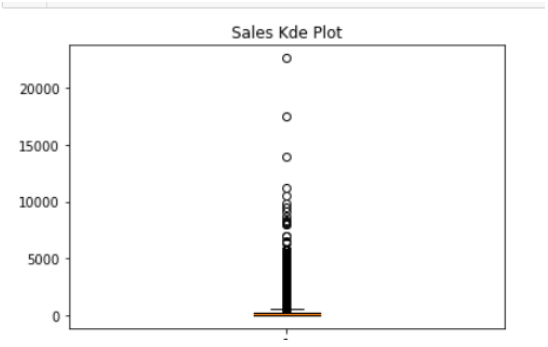

Sub-Category
2500
2000
1500
1000
500
0
Storage
Chairs
Phones
Binders
Art
Bookcases
Accessories
Copiers
Furnishings
Paper
Appliances
Supplies
Machines
Envelopes
Labels
Fasteners
Tables


Order Priority
14000
12000
10000
8000
6000
4000
2000
0
Medium
High
Critical
Low

6.1 Relation between variables - Univariate Analysis for numerical features

Numeric Feature: ‘Sales’

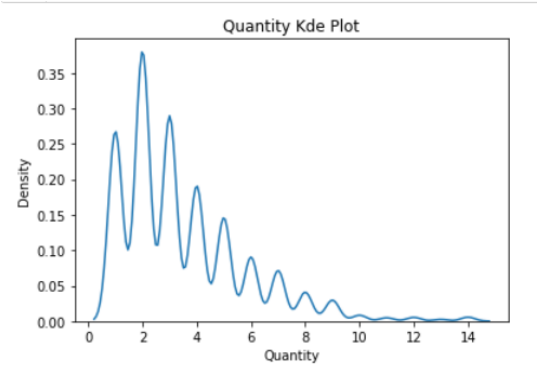


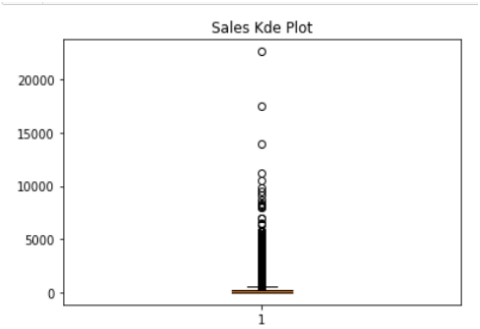



```
Skewness 8.138080021220542
Kurtosis 176.7311999336456
```

- IQR of the 'Sales ' is small as compared to the range of its values.
- Feature heavily right skewed as it has high positive skewedness value.
- Kurtosis value is very high implying that there are a lot of values located in the tail part of thedistribution.

Numeric Feature: ‘Quantity’

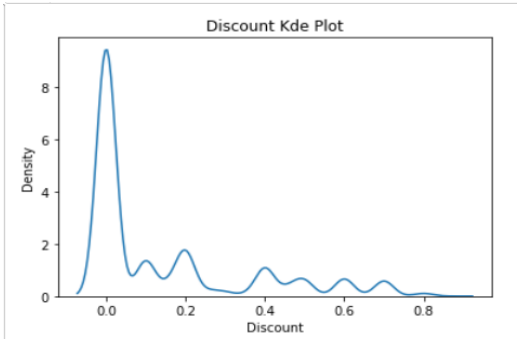


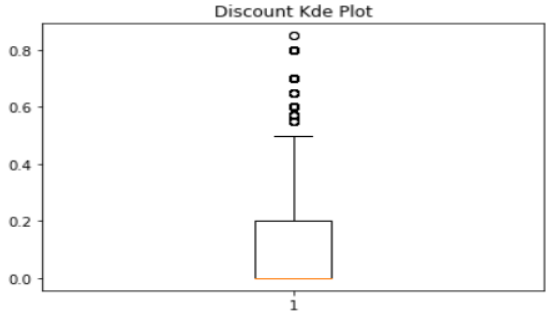



```
Skewness 1.36036773070242
Kurtosis 2.2758887299856587
```

- IQR of the 'Quan tity' is small as compared to the range of its values.
- Feature is right skewed as it has positive skewedness value.
- Kurtosis is medium.

 Numeric Feature: ‘Quantity’

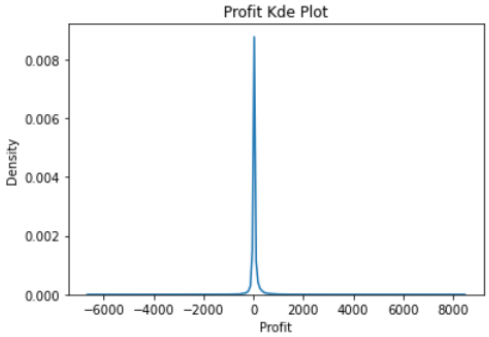


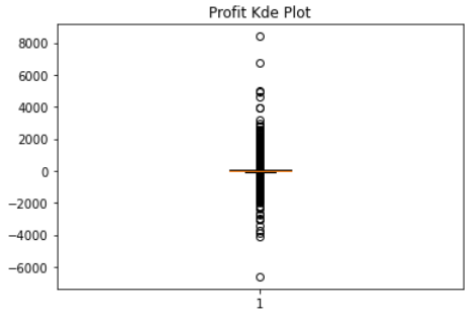



```
Skewness 1.3877745521338385
Kurtosis 0.7166824084745773
```

- IQR of the Discount is medium as compared to the range of its values and q1 is zero as discount cannot be less than 0.
- Feature is right skewed as it has positive skewedness value.
- Kurtosis is medium.

 Numeric Feature: ‘Profit

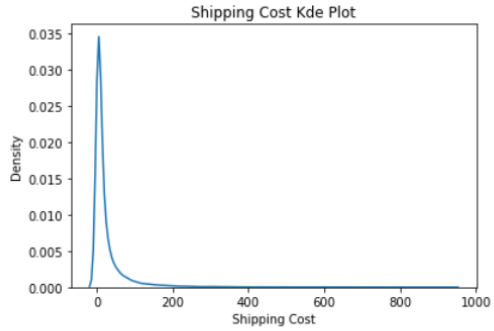


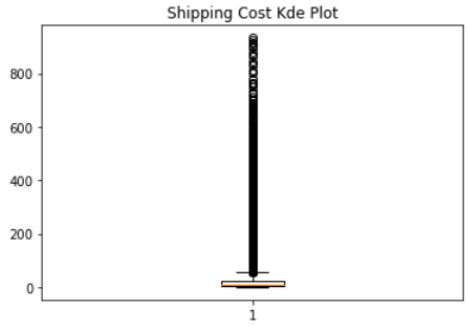



```
Skewness 4.157188532777366
Kurtosis 291.41108956980673
```

- IQR of the Profit is small as compared to the range of its values. Q1 is negative indicating the return orders.
- Feature is right skewed as it has positive skewedness value.
- Kurtosis value is very high implying that there are a lot of values located in the tail part of thedistribution.

 Numeric Feature: ‘Shipping Cost’



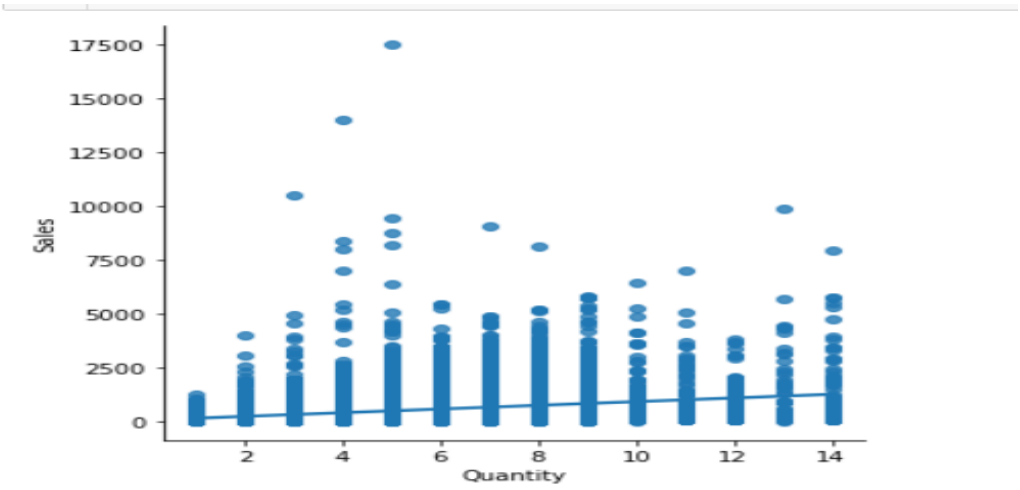



```
Skewness 5.8632264257106765
Kurtosis 50.020157738724116
```

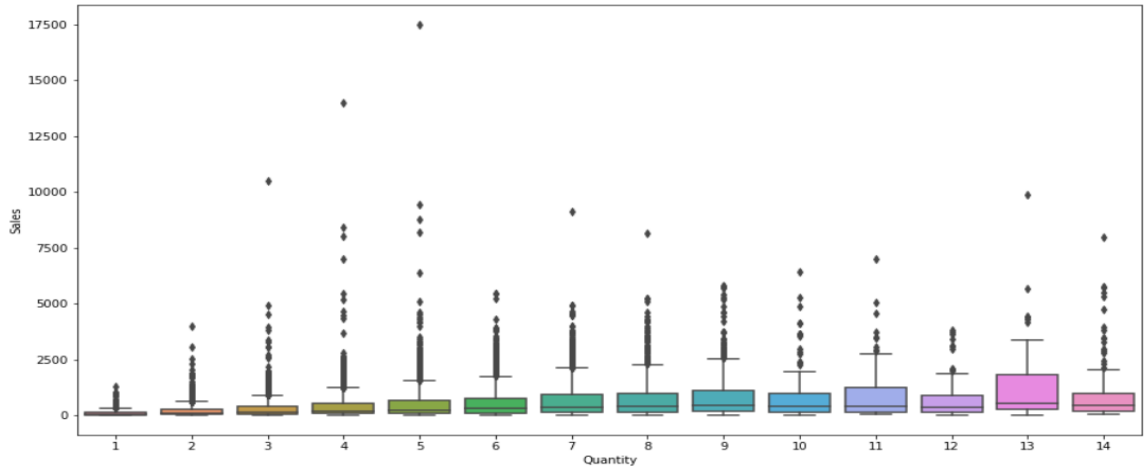
- IQR of the Shipping Cost ' is small as compared to the range of its values.
- Feature heavily right skewed as it has high positive skewedness value.
- Kurtosis value is very high implying that there are a lot of values located in the tail part of thedistribution.

6.2 Relation between variables - Bi-variate Analysis for numerical features.

- Numeric Feature: ‘Quantity w.r.t Sales’

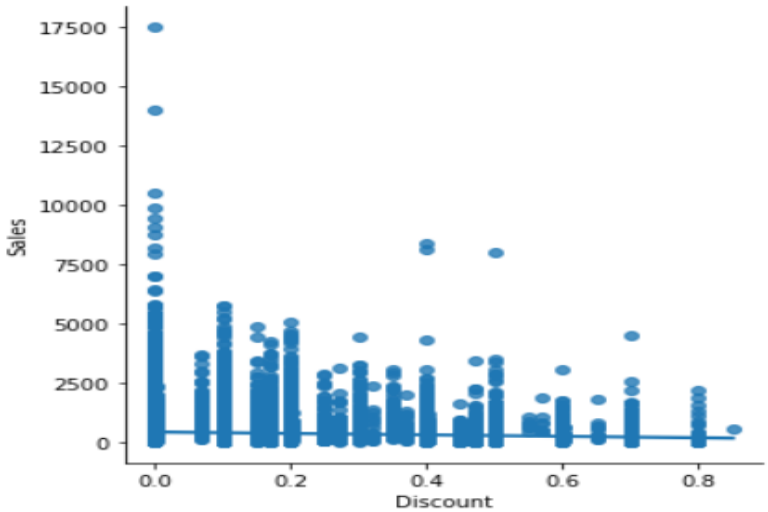


- There is a slight co relation between Sales and quantity indicated by the slope of the line.
- There are few outliers present in the data which indicates that although quantity was less but the product price is high which increased the overall sales amount.



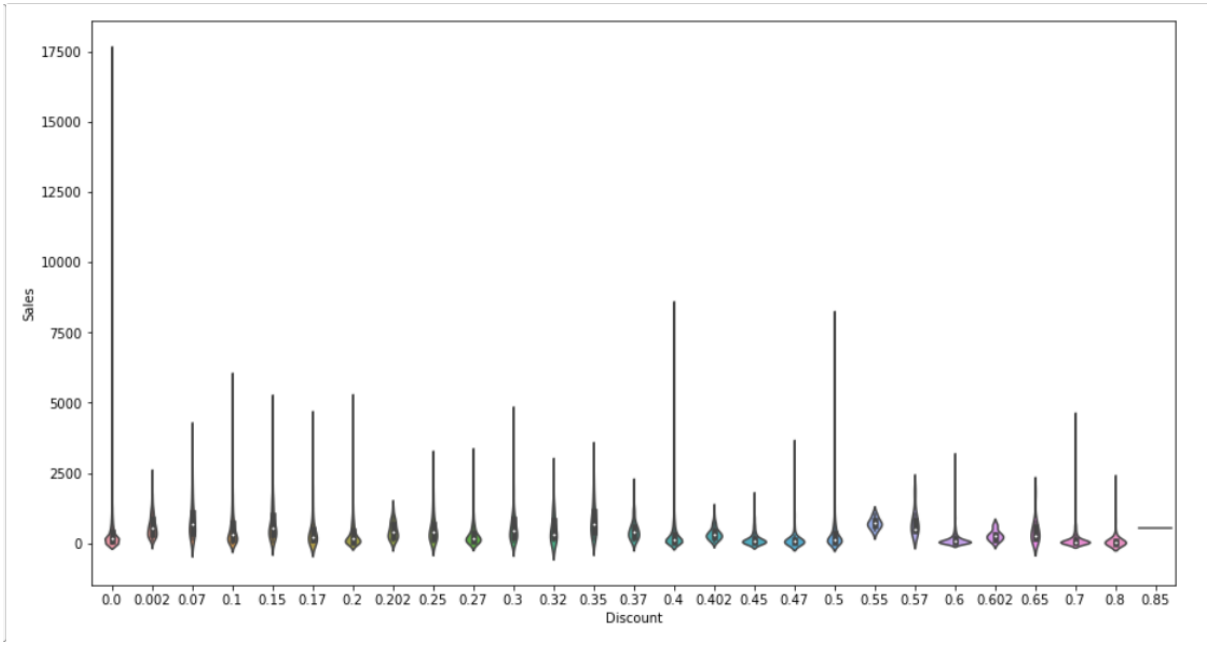
- As we can see for every order quantity ordered of products have few outliers which aroused due to variation in the price of the products.

- Numeric Feature: 'Discount w.r.t Sales'

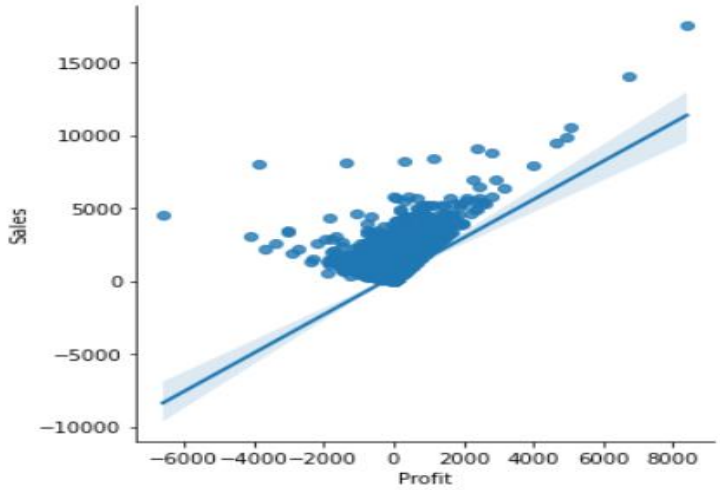


- There is weak correlation between Discount and Sales.
- Indicating Discount does not have effect have overall Sales amount.
- Max Sales happens when there was no discount offered.

Distribution of Sales over Discount: -

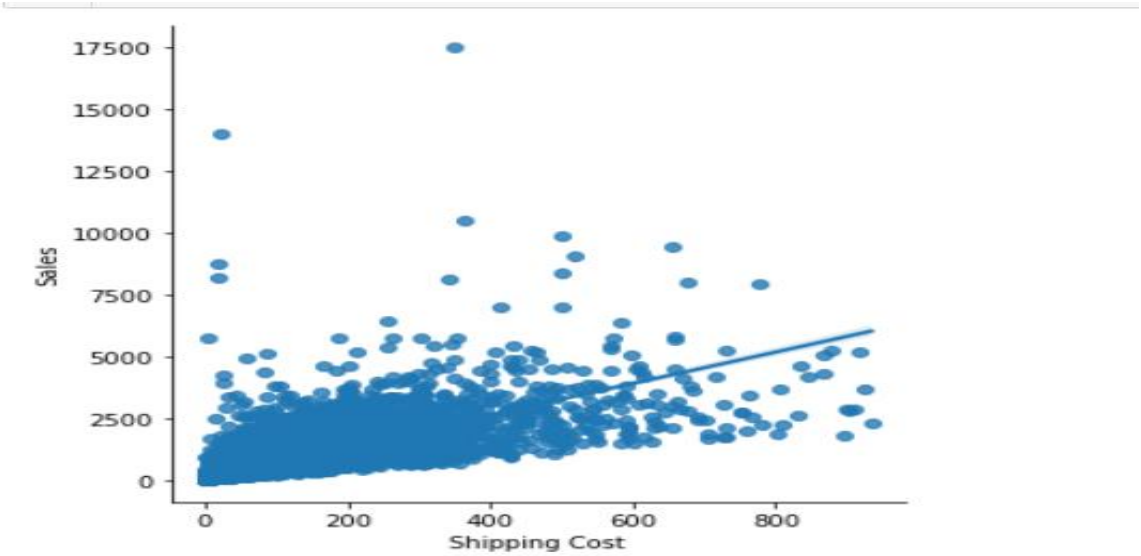


- Numeric Feature: 'Profit w.r.t Sales'



- There is strong correlation between profit and sales.
- This is because in one or the way both are derived from each other or we can say that profit is a part of Sales that's why showing strong correlation.

- Numeric Feature: ‘Shipping Cost w.r.t Sales’



- There is a strong correlation between Shipping Cost and sales.

7. Analysis of Data from Customer Point of View

7.1 Profile the customers based on their frequency of purchase - calculate frequency of purchase for each customer.

```
def new_features(x):
    d = []
    d.append(x['Order ID'].nunique())
    d.append(x['Sales'].sum())
    d.append(x['Profit'].sum())
    d.append(x['Shipping Cost'].sum())
    d.append(pd.to_datetime(x['Order Date']).min())
    d.append(pd.to_datetime(x['Order Date']).max())
    d.append(x['City'].nunique())
    return pd.Series(d, index=['Purchases','Total_Sales',"Total_Profit",'Total_Cost',
                               'First_Purchase_Date','Latest_Purchase_Date','Location_Count'])

df_customer_new = df.groupby('Customer ID').apply(new_features)
```

- We have made a list of features and then made a new group series that we want to further classify the customers according to the frequency of purchase.

Customer ID	Purchases	Total_Sales	Total_Profit	Total_Cost	First_Purchase_Date	Latest_Purchase_Date	Location_Count
AA-10315	19	13747.41300	447.69050	1236.15	2011-03-31	2014-12-23	18
AA-10375	23	5884.19500	677.47740	903.92	2011-04-21	2014-12-25	23
AA-10480	20	17695.58978	1516.47518	1633.67	2011-01-11	2014-09-05	20
AA-10645	36	15343.89070	3051.43900	1752.27	2011-01-12	2014-12-05	35
AA-315	7	2243.25600	535.56600	215.80	2011-08-06	2014-12-29	7

- By subtracting the latest purchase and first chase and dividing it by no of purchases we got the frequency of ordering by customers.
- We further classify the customers into three classes Low, Med and High.

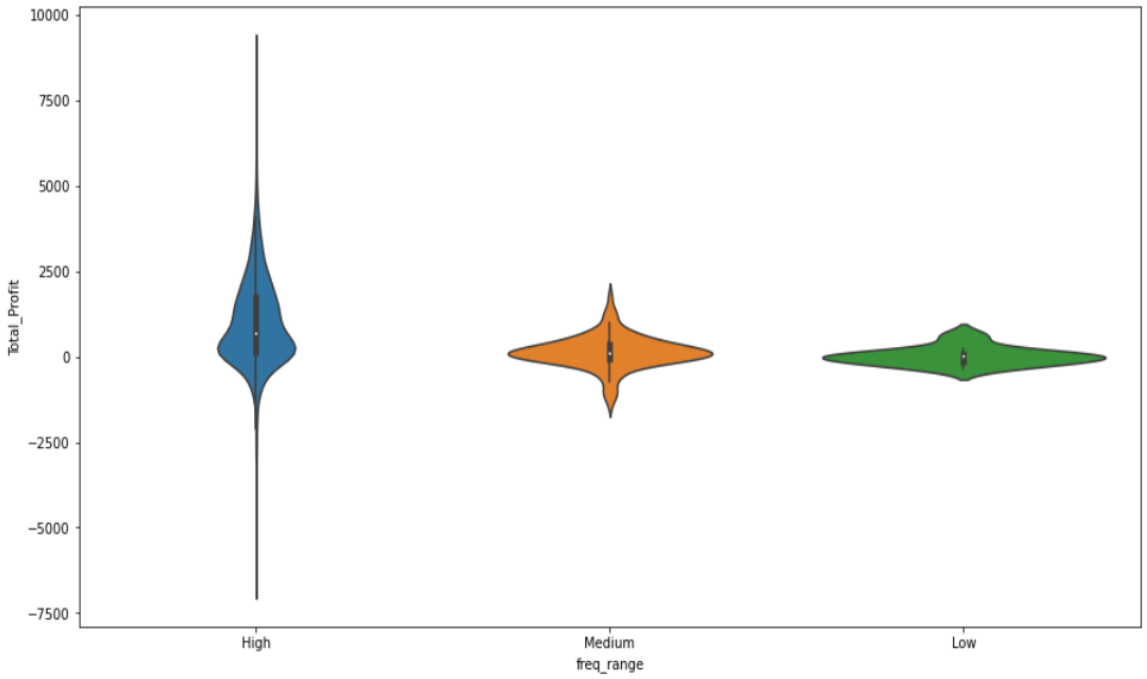
```
# bucketing continuous data
def freq(x):
    if x < 219:
        return 'High'
    elif x < 436:
        return 'Medium'
    else:
        return 'Low'

df_customer_new['freq_range'] = df_customer_new.Frequency.apply(freq)

df_customer_new['freq_range'].value_counts()
```

- Lower the Value of frequency means highly frequent he/she is.

```
High      1442
Medium     138
Low         10
Name: freq_range, dtype: int64
```

7.2. Are the high frequent customers contributing more revenue

```
sns.violinplot(data=df_customer_new,x="freq_range",y="Total_Profit")
plt.show()
```

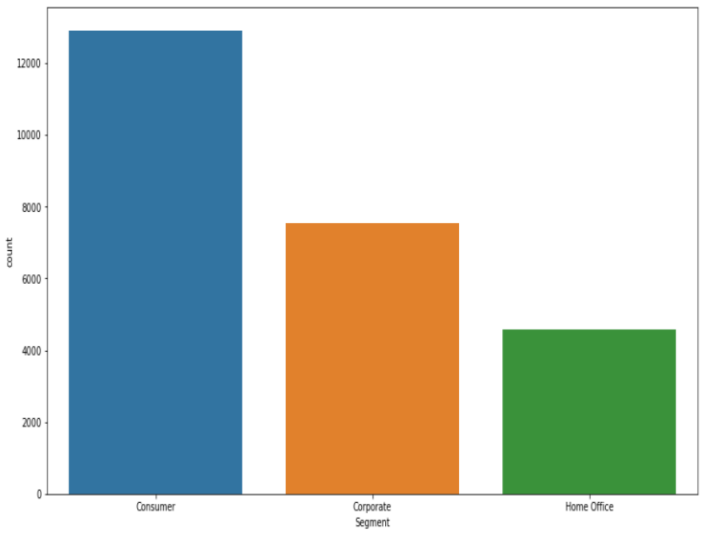


- We can observe that the total profit is maximum for highly frequent customers.
- We should try to increase the measures that leads to high frequency of customers.
- There is no such variation among medium and low frequent customers.

7.3 Categorisation of These Customers into Segments.

```
sns.countplot(df1["Segment"])
```

```
<AxesSubplot:xlabel='Segment', ylabel='count'>
```

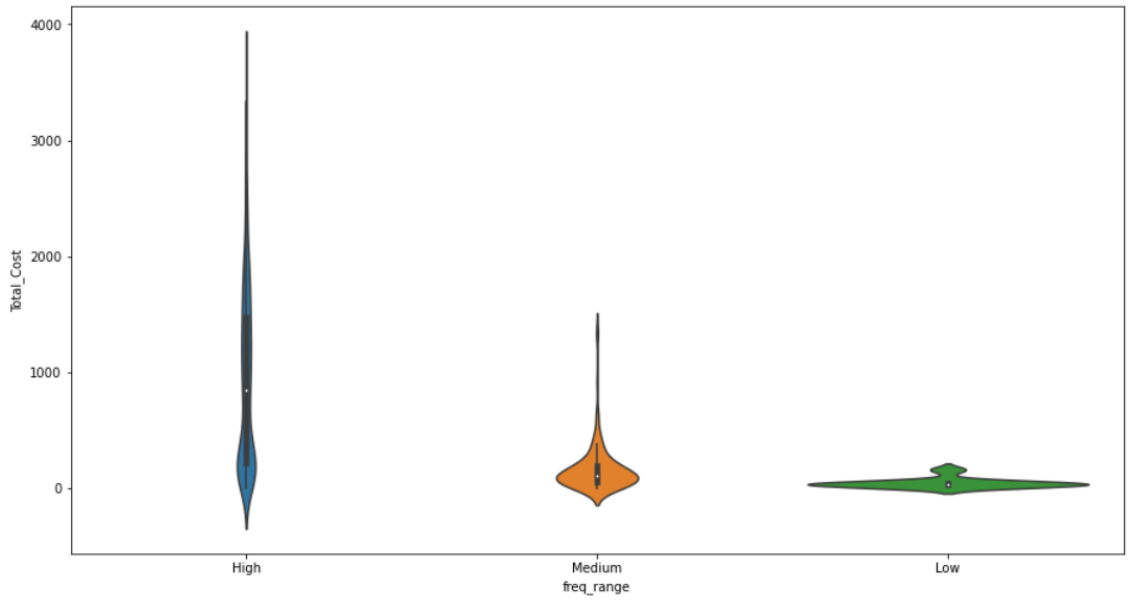


Description: -

- There are 3 segments to which the customer belongs Consumer, Corporate Segment, Home office.
- Maximum customer belongs to retail consumer.

7.4 Categorisation of Customers on basis of Shipping Cost.

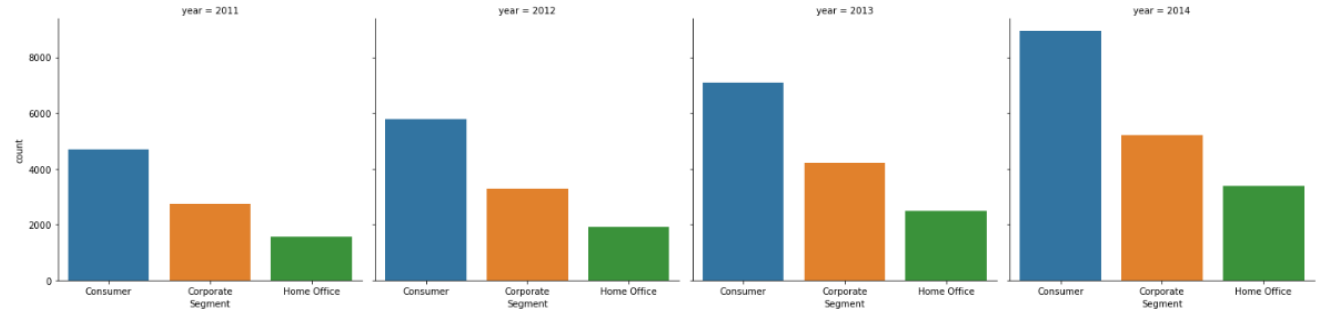
```
sns.violinplot(data=df_customer_new,x="freq_range",y="Total_Cost")
plt.show()
```



- Customers with High Frequency category contributed to maximum shipping cost.
- Sometimes shipping cost also might be a parameter there might be some discounts that can be offered to med and low frequent customers.

7.5 Year on Year Analysis of Customer Segment.

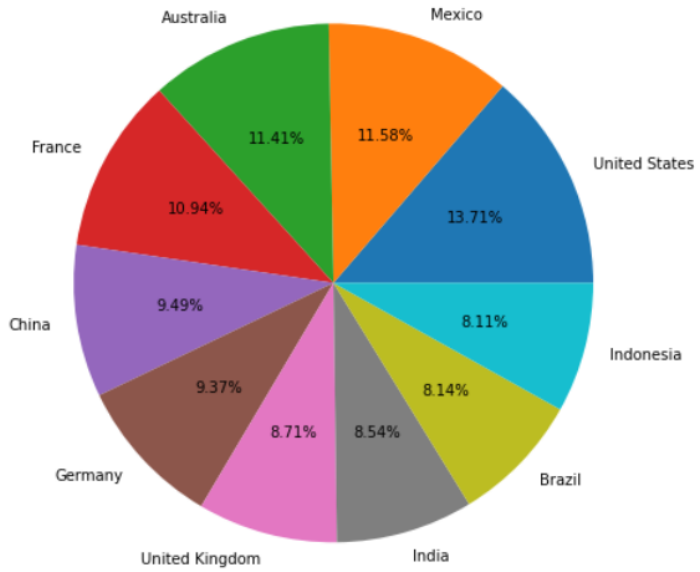
```
sns.catplot(x="Segment", col="year", data=df_x, kind="count")
plt.show()
```



- Segregation of customer in different segments on yearly basis.
- We can observe that we have increased the number of customers across all the segments and the contribution along retail customer is very high .

7.6 Top 10 Countries Customers Wise.

```
plt.pie(x=customer_country.Count.values,labels=customer_country.Country.values,autopct="%0.2f%%")
plt.show()
```



8. Product Analysis

8.1. Top Selling Product

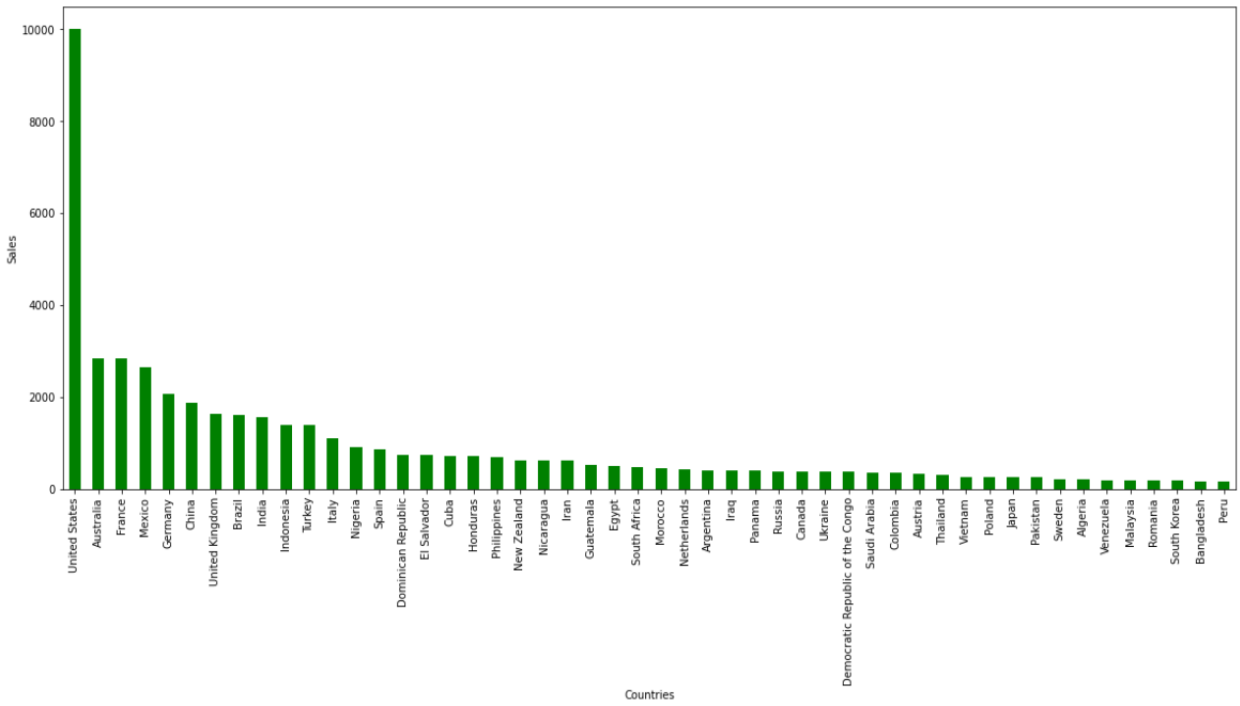
```
best_product = df.groupby('Product Name')['Sales'].count().sort_values(ascending=False)
best_product.head()
```

```
Product Name
Staples                             227
Cardinal Index Tab, Clear            92
Eldon File Cart, Single Width        90
Rogers File Cart, Single Width       84
Ibico Index Tab, Clear               83
```

- We Calculated the top products which have highest number of sales till now
- We counted the sales of each product as number of times was sold
- These were top 5 products which were most sold

8.2. Top Countries with most orders

```
plt.figure(figsize=(20,8))
top_countires = df.groupby('Country')['Sales'].count().sort_values(ascending = False)
#Though the countries are very much we will only take for 50 countires
top_countires = top_countires[:50]
top_countires.plot(kind = 'bar', color = 'green')
plt.ylabel('Sales')
plt.xlabel('Countries')
plt.show()
```



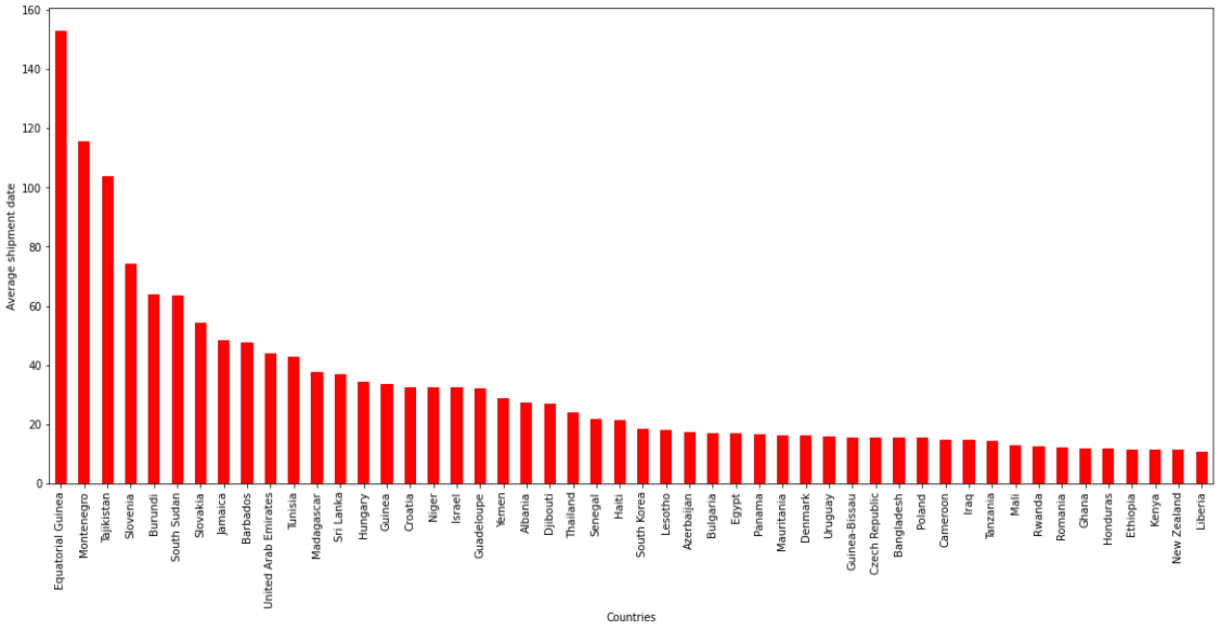
- Observed that United States has the most number of orders world wide
- Observations also displays top 50 countries with hightest number of orders

8.3. Average days to Ship the order of each Country

```
df_new['Days_to_ship_orders'] = (pd.to_datetime(df_new['Ship Date']) - pd.to_datetime(df_new['Order Date'])).dt.days
df_new.head()
```

- Here we subtracted the date of ‘Ship Date’ With ‘Order Date’ To calculate the number of days for each order placed so that we can calculate the average days to deliver an order of whole dataset.
- This way we can observe for each country the average time of an order to be delivered

```
plt.figure(figsize=(20,8))
avg_dates = df_new.groupby('Country')['Days_to_ship_orders'].mean().sort_values(ascending=False)
#WIll will take only 50 countries
avg_dates = avg_dates [:50]
avg_dates.plot(kind='bar', color='red')
plt.ylabel('Average shipment date')
plt.xlabel('Countries')
```



- Here we Observed via plotting a graph and grouping by countries to get the avg time taken for an order to get delivered country wise.

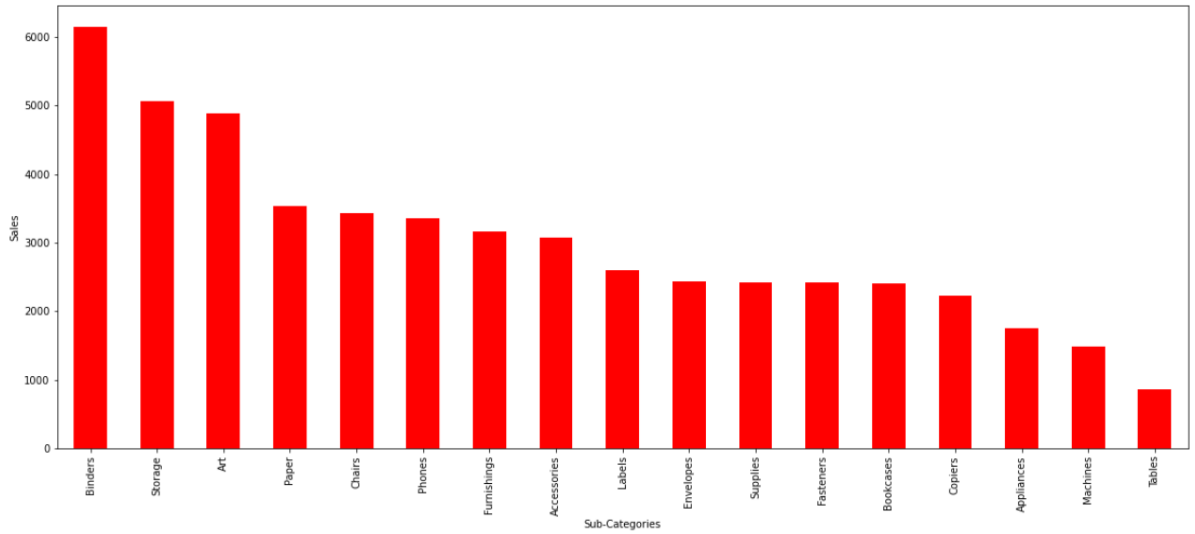
8.4. Best Performing Sub-Category

```
best_sub_category = df_x.groupby('Sub-Category')['Sales'].count().sort_values(ascending=False)
best_sub_category.head()
```

```
Sub-Category
Binders    6152
Storage    5059
Art        4883
Paper      3538
Chairs     3434
```

- These are top 5 best Sub-Category which have performed really well during these years
- We counted the sales all sub-category and given the Top 5

```
plt.figure(figsize=(20,8))
best_sub_category.plot(kind='bar', color='red')
plt.ylabel('Sales')
plt.xlabel('Sub-Categories')
```



- Here is an Bar Plot observation of sales of all sub-categories.
- We can observe all Top and least performing Category

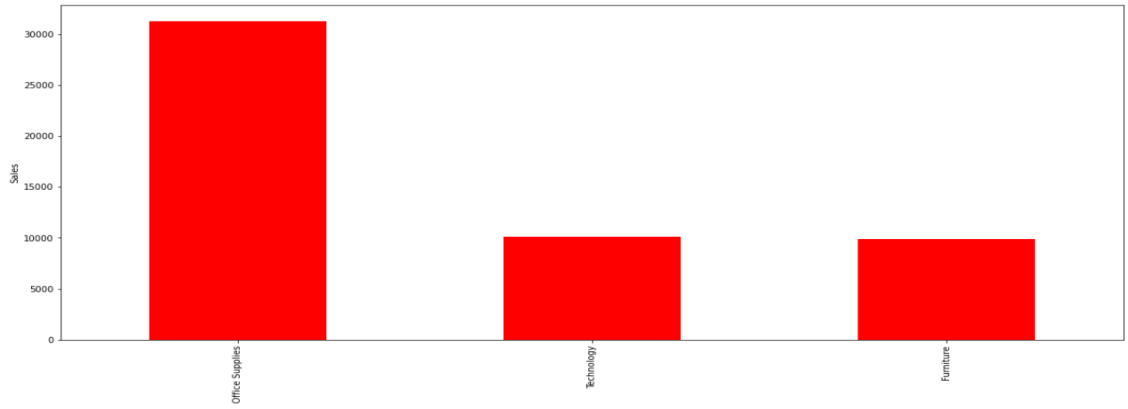
8.5. Best Performing Category

```
best_category = df_new.groupby('Category')['Sales'].count().sort_values(ascending=False)
best_category.head()
```

```
Category
Office Supplies    31273
Technology         10141
Furniture           9876
```

- We only have 3 Main Categories here but with observation we can see that Office supplies is the Category which have highest number of Sales
- Technology stands at second position and Furnitue at Third
-

```
plt.figure(figsize=(20,8))
best_category.plot(kind='bar', color='red')
plt.ylabel('Sales')
plt.xlabel('Categories')
```



8.6. Top Profit-Making Products per Year

```
products_best = df_x.groupby(['year','Product Name'])['Profit'].max().sort_values(ascending=False)
products_best.head(4)
```

```
year  Product Name
2013  Canon imageCLASS 2200 Advanced Copier                8399.9760
2014  Canon imageCLASS 2200 Advanced Copier                6719.9808
2013  GBC Ibimaster 500 Manual ProClick Binding System     4946.3700
2011  Ibico EPK-21 Electric Binding System                 4630.4755
```

- We can observe the top performing Products for year 2011, 2012, 2013 and 2014 means of all all 4 years top product names.
- Most Profit came out from these products every year.

9. Business Problem Review: -

So far, we have conducted a thorough research on our data regarding the sales of different products, under different Sub- Categories according to the Region, Country, State, City of the customers to understand their consumption pattern and behaviour and tried to figure out their impact on sales with in consideration of Discounts and Shipping Cost etc. Now we will look at the intensity of these features and try to analyse if we can find any Linear relationship between sales and the features affecting the sale and trying to figure out the value of Beta coefficients to capture the ratio of change and try to predict the possible solutions to increase the sales.

10. Feature Engineering: -

- Transformation techniques like square root, cube root, log1p, power transformation will beapplied and results verified before finalising.
- Standardization and Normalization techniques using Standard Scaler, Min Max Scaler, One Hot Encoder, Ordinal Encoder, Label encoder.
- Imputing Missing values using Simple Imputer ,KNN Imputer.
- Feature selection – using K-Best, Recursive Feature Elimination or Sequential FeatureSelection.
- Dimensionality reduction using PCA can be considered or usage of Variance Inflation Factor to drop down insignificant variables for the tuning of model.
- SMOTE or oversampling of train data to deal with imbalance.
-

Transformation of Categorical features using ordinal encoder.

```
df_cat.drop(['Order ID', 'Order Date', 'Ship Date','Customer ID','Product ID'],axis=1,inplace=True)
```

```
df_cat1=pd.DataFrame(oe.fit_transform(df_cat),columns=df_cat.columns)
```

```
df_cat1.head()
```

	Ship Mode	Customer Name	Segment	City	State	Country	Market	Region	Category	Sub-Category	Product Name	Order Priority
0	1.000000	632.000000	0.000000	2262.000000	703.000000	139.000000	6.000000	6.000000	2.000000	0.000000	2625.000000	0.000000
1	2.000000	413.000000	1.000000	3483.000000	702.000000	6.000000	0.000000	9.000000	0.000000	5.000000	2407.000000	0.000000
2	0.000000	181.000000	0.000000	492.000000	820.000000	6.000000	0.000000	9.000000	2.000000	13.000000	2384.000000	3.000000
3	0.000000	424.000000	2.000000	373.000000	145.000000	47.000000	4.000000	3.000000	2.000000	13.000000	2306.000000	3.000000
4	1.000000	632.000000	0.000000	848.000000	270.000000	110.000000	1.000000	0.000000	2.000000	6.000000	3020.000000	0.000000

Transformation of Numerical features using MinMaxScaler

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

```
b=mm.fit_transform(df_num)
```

```
df_num1=pd.DataFrame(b,columns=df_num.columns)
```

```
df_num1.head()
```

	Row ID	Sales	Quantity	Discount	Profit	Shipping Cost	Order_Year
0	0.629706	0.131953	0.461538	0.000000	0.490812	1.000000	0.333333
1	0.513560	0.211941	0.615385	0.117647	0.420749	0.989353	0.666667
2	0.493849	0.295703	0.615385	0.117647	0.501331	0.980633	0.666667
3	0.263663	0.165260	0.307692	0.117647	0.433564	0.974924	0.666667
4	0.920665	0.161857	0.538462	0.000000	0.460768	0.967298	0.666667

11. What is Regression Analysis?

- Dependent variable (y): It is the variable that we predict. It is also known as 'Target' or 'Response' variable. For regression, the dependent variable should be numeric.
- Independent variable (X): It is the variable used in predicting the values of a target variable. There can be one or more independent variables in a dataset. It is also known as 'predictors' or 'features'. These variables can be numerical as well as categorical.
- Regression analysis is used to understand the effect of a set of independent variables on the dependent variable. The variables that affect the dependent variable the most can also be identified. Linear regression is one of the methods in regression analysis, which fits the line to predict the values of the dependent variable

12.BASE MODEL WITH THE DATA.

M-1: - Ordinary Least Square Method (OLS)

- Splitting of Data using Train_Test_Split.

```
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=1)
print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
x_train (17524, 18)
x_test (7511, 18)
y_train (17524,)
y_test (7511,)
```

- Fitting a model and summary of the model.

```
OLS_Model_1=sm.OLS(y_train,x_train).fit()
print(OLS_Model_1.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.725
Model:                            OLS   Adj. R-squared:                  0.725
Method:                 Least Squares   F-statistic:                     2718.
Date:                Wed, 10 Aug 2022   Prob (F-statistic):               0.00
Time:                        23:41:35   Log-Likelihood:                 45307.
No. Observations:               17524   AIC:                        -9.058e+04
Df Residuals:                   17506   BIC:                        -9.044e+04
Df Model:                          17
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            -0.3237      0.005    -70.069      0.000      -0.333      -0.315
Quantity          0.0206      0.001     26.108      0.000       0.019       0.022
Discount          0.0103      0.001     16.672      0.000       0.009       0.012
Profit            0.7055      0.010     68.261      0.000       0.685       0.726
Shipping Cost     0.2952      0.002    145.233      0.000       0.291       0.299
Order_Year    -4.472e-05      0.000     -0.119      0.905      -0.001       0.001
Ship Mode         0.0024      0.000     17.484      0.000       0.002       0.003
Customer Name -1.314e-08   6.03e-07     -0.022      0.983   -1.19e-06    1.17e-06
Segment          -0.0002      0.000     -0.865      0.387      -0.001       0.000
City          -1.002e-07   1.43e-07     -0.701      0.483    -3.8e-07     1.8e-07
State            4.3e-07   4.68e-07      0.919      0.358   -4.87e-07    1.35e-06

Country       -7.463e-07   3.38e-06     -0.221      0.825   -7.38e-06    5.89e-06
Market           -0.0003   6.96e-05     -3.998      0.000      -0.000      -0.000
Region         6.371e-05   3.94e-05      1.619      0.105   -1.34e-05       0.000
Category         -0.0003      0.000     -1.663      0.096      -0.001    5.84e-05
Sub-Category      0.0001   2.83e-05      4.696      0.000    7.74e-05       0.000
Product Name  -2.999e-07   1.39e-07     -2.160      0.031   -5.72e-07   -2.77e-08
Order Priority    0.0039      0.000     27.831      0.000       0.004       0.004
==============================================================================
Omnibus:                    19589.139   Durbin-Watson:                   1.997
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          7448987.277
Skew:                           5.225   Prob(JB):                         0.00
Kurtosis:                     103.462   Cond. No.                     2.33e+05
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.33e+05. This might indicate that there are
strong multicollinearity or other numerical problems.
```

Interpretation: The R-squared value obtained from this model is .725 which means that the above model explains 72.5% of the variation in the Sales.

What is Auto-Correlation??

Autocorrelation means the relationship between each value of errors in the equation. Or in the other hand, autocorrelation means the self-relationship of errors.

The **`Durbin-Watson`** test is used to check the autocorrelation between the residuals.
If the Durbin-Watson test statistic is near to 2: no autocorrelation
If the Durbin-Watson test statistic is between 0 and 2: positive autocorrelation
If the Durbin-Watson test statistic is between 2 and 4: negative autocorrelation
The summary output shows that the value of the test statistic is close to 2 (= 1.997) which means there is no autocorrelation.

The `Jarque-Bera` test is used to check the normality of the residuals. Here, the p-value of the test is less than 0.05; that implies the **residuals are not normally distributed**.

What is Multi collinearity??
Multicollinearity occurs when two or more independent variables are highly correlated with one another in a regression model. This means that an independent variable can be predicted from another independent variable in a regression model.
The 'Cond. No' (= 1) represents the **`Condition Number (CN)`** which is used to check the multicollinearity.
If CN < 100: no multicollinearity
If CN is between 100 and 1000: moderate multicollinearity
If CN > 1000: severe multicollinearity

Here the Condition Number is very high indicating the High multicollinearity among features.

Measure of Variation

Residual: It is calculated as the difference between the actual and predicted value of the dependent variable.

Sum of Squared Residuals (SSR):
It is defined as the sum of the squared difference between the predicted value and the mean of the dependent variable.

Sum of Squared Error (SSE)
It is defined as the sum of the squared difference between the actual value and the predicted value.

Sum of Squared Total (SST)
It is the sum of the squared difference between the actual value and the mean of the dependent variable.

13. Different Models and Techniques

M-2 Multiple Linear Regression (MLR)

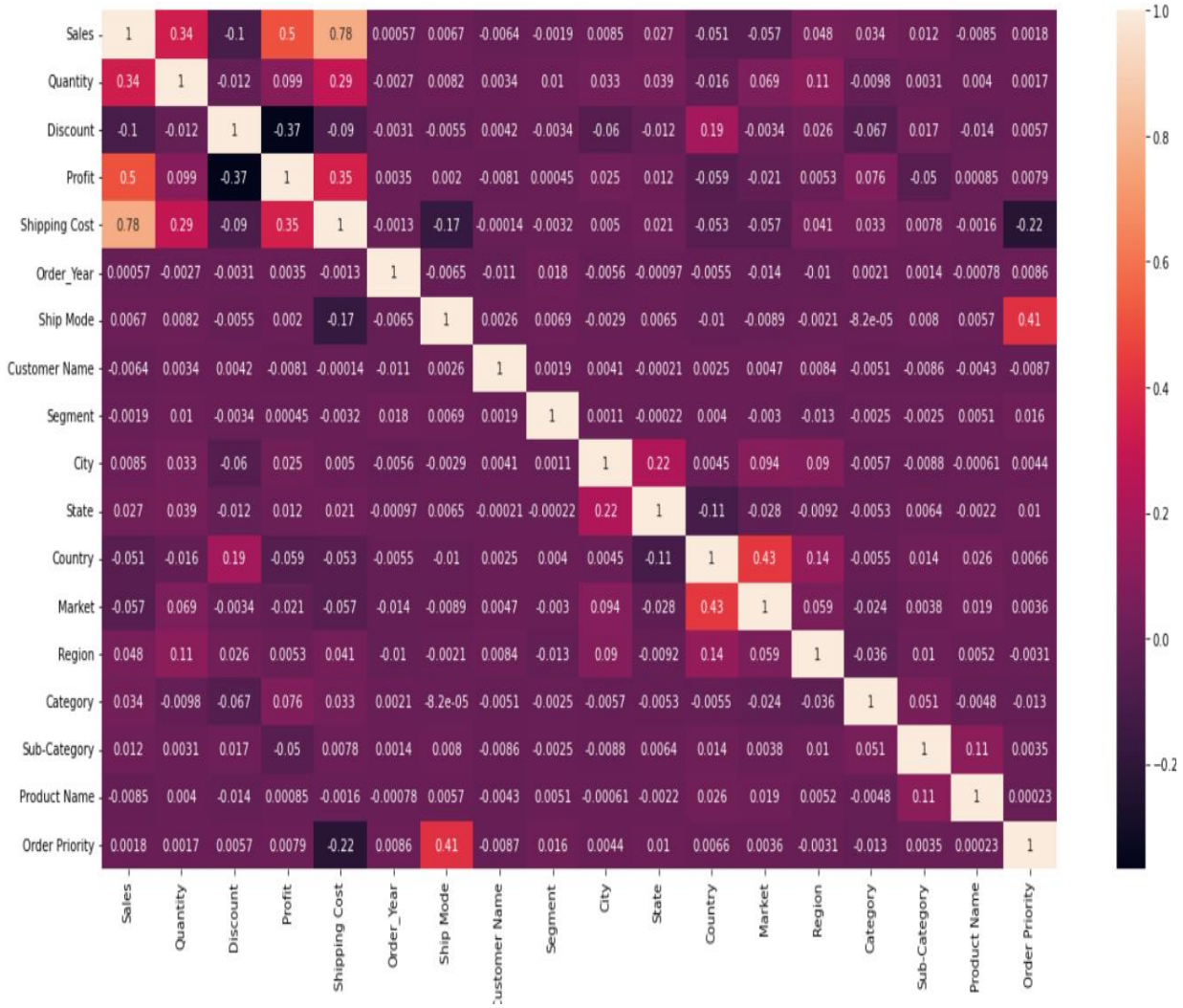
Assumptions of MLR Model.

There are several assumptions of linear regression. We can check two of them before building the regression model. Let us check the following assumptions in this section:

- Assumption on Dependent Variable
- No or little multicollinearity

1. The assumption for linear regression is that the dependent (target) variable should be numeric. In our dataset, the variable Sales is the target/dependent variable.
2. Whenever there are a large number of independent variables present in the dataset, there is a possibility that such independent variables will be strongly correlated. The presence of a strong correlation between the independent variables is called multicollinearity.
3. The presence of multicollinearity can destabilize the model. Thus, the existence of multicollinearity must be detected and corrected actions should be taken.

It Can be checked by removing the highly correlated features from the data which can be visualised using heatmap of correlation of features.



Interpretations: -

By analysing the heat map it can be inferred that the hight multicollinearity is because of our categorical features.

We will try another model with all the numerical features and with few categorical features.

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                  Sales   R-squared:                       0.698
Model:                            OLS   Adj. R-squared:                  0.697
Method:                 Least Squares   F-statistic:                     6734.
Date:                Thu, 11 Aug 2022   Prob (F-statistic):               0.00
Time:                        16:48:41   Log-Likelihood:                 44466.
No. Observations:               17524   AIC:                         -8.892e+04
Df Residuals:                   17517   BIC:                         -8.886e+04
Df Model:                           6
Covariance Type:            nonrobust
================================================================================
                   coef    std err          t      P>|t|      [0.025      0.975]
--------------------------------------------------------------------------------
const            -0.3260      0.005    -68.206      0.000      -0.335      -0.317
Row ID           -0.0002      0.001     -0.467      0.640      -0.001       0.001
Quantity          0.0230      0.001     27.830      0.000       0.021       0.025
Discount          0.0107      0.001     16.842      0.000       0.009       0.012
Profit            0.7417      0.011     68.967      0.000       0.721       0.763
Shipping Cost     0.2740      0.002    133.801      0.000       0.270       0.278
Order_Year     2.557e-05      0.000      0.065      0.948      -0.001       0.001
==============================================================================
Omnibus:                    18438.949   Durbin-Watson:                   2.009
Prob(Omnibus):                  0.000   Jarque-Bera (JB):          4893608.450
Skew:                           4.785   Prob(JB):                         0.00
Kurtosis:                      84.305   Cond. No.                         113.
==============================================================================

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

When we removed the features and made the model again there is significant drop in multicollinearity and it comes out approx. 100.

Model Performance Evaluation.

To capture the performance of model we have checked for R-Square, Mean Squared Error, Root Mean Squared Error, Mean Absolute Error, Mean Absolute Percentage Error for both Train and test.

```
Score_Card=pd.DataFrame(columns=["Algorithm_Name","Model_Name","R_Square_train","R_Square_test","Train_MSE","Test_MSE",
                                 "Train_MAE","Test_MAE","Train_RMSE","Test_RMSE","Train_MAPE","Test_MAPE"])
```

```
def Update_Score_Card(Algorithm_Name, model_name,model, x_train,x_test,y_train,y_test):
    global Score_Card
    train_pred=get_train_pred(model,x_train)
    test_pred = get_test_pred(model,x_test)

    Score_Card=Score_Card.append({
        "Algorithm_Name":Algorithm_Name,
        "Model_Name":model_name,
        "R_Square_train":R_Square(y_train,train_pred),
        "R_Square_test":R_Square(y_test,test_pred),
        "Train_MSE":get_Train_MSE(y_train,train_pred),
        "Test_MSE":get_Test_MSE(y_test,test_pred),
        "Train_MAE":get_Train_MAE(y_train,train_pred),
        "Test_MAE":get_Test_MAE(y_test,test_pred),
        "Train_RMSE":get_Train_RMSE(y_train,train_pred),
        "Test_RMSE":get_Test_RMSE(y_test,test_pred),
        "Train_MAPE":get_Train_MAPE(y_train,train_pred),
        "Test_MAPE":get_Test_MAPE(y_test,test_pred)
        },ignore_index=True)
```

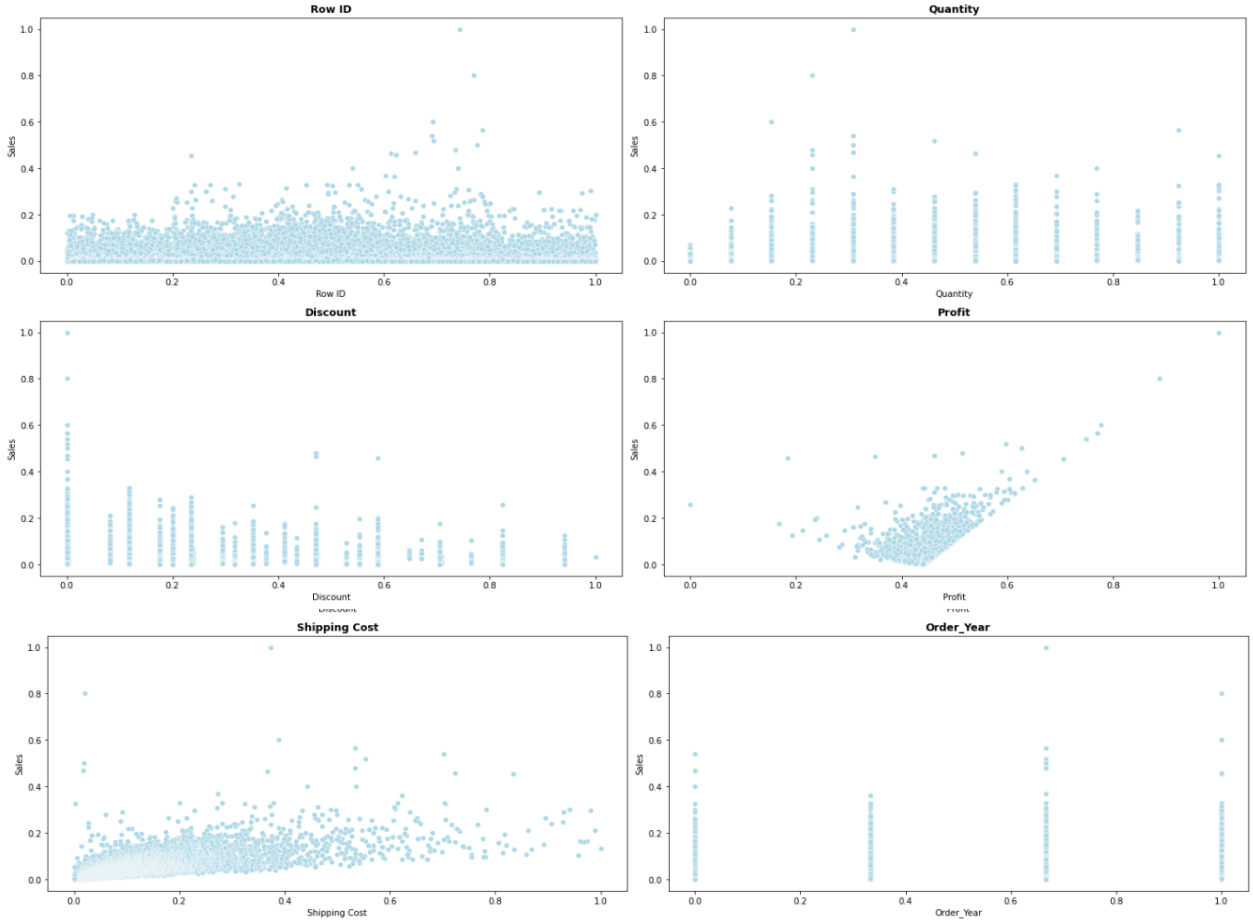
```
1 Score_Card
```

Algorithm_Name	Model_Name	R_Square_train	R_Square_test	Train_MSE	Test_MSE	Train_MAE	Test_MAE	Train_RMSE	Test_RMSE	Train_MAPE	Test_MAPE
OLS	OLS_Model_1	0.725234	0.688087	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	251.352064
OLS	OLS_Model_2	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	204.863606
MLR	MLR_Model_3	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	204.863606

Assumptions After MLR Model

Now, use the model with significant variables to check some of the assumptions based on the residuals of linear regression:

1. Linear Relationship Between Dependent and Independent Variable
2. Autocorrelation
3. Heteroscedasticity
4. Tests of Normality



2.Autocorrelation

The Value of Durbin Watson Test is approx. 2 Which indicates there is no autocorrelation among features.

3.Heteroscedasticity

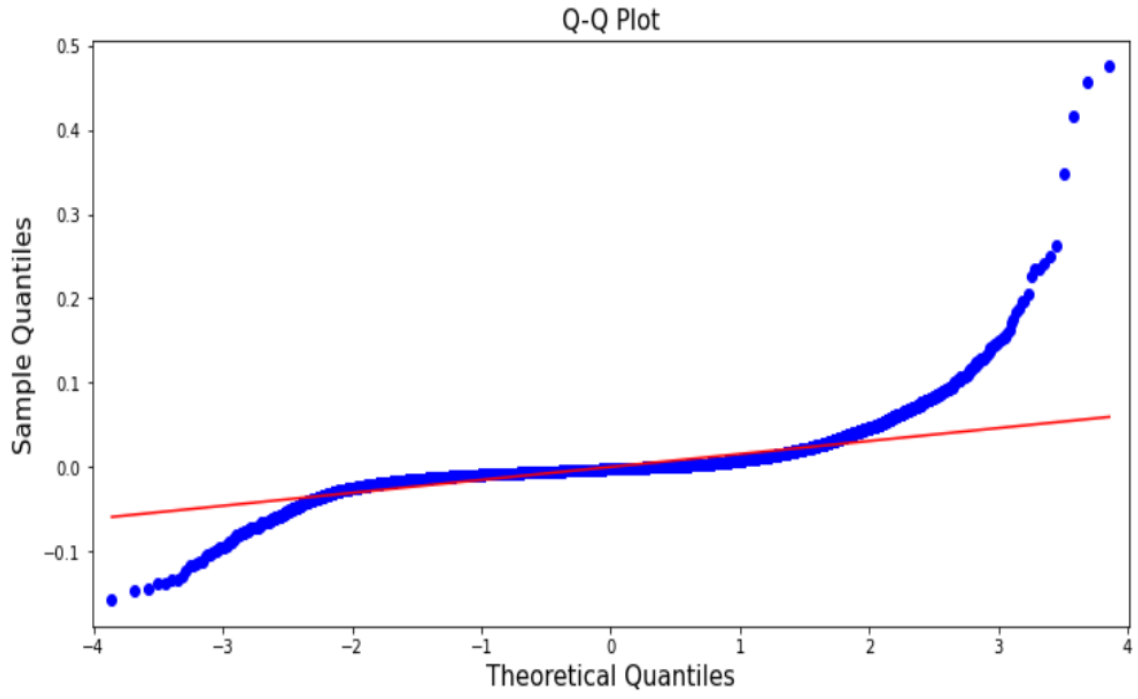
If the residuals have constant variance across different values of the predicted values, then it is known as `Homoskedasticity`. The absence of homoskedasticity is known as, heteroskedasticity. One of the assumptions of linear regression is that heteroskedasticity should not be present. Let us study two different tests to check the presence of heteroskedasticity.
Breusch-Pagan is one of the tests for detecting heteroskedasticity in the residuals.
The test hypothesis for the Breusch-Pagan test is given as:

H_0: There is homoscedasticity present in the data
H_1: There is a heteroscedasticity present in the data

```
[('f-value', 289.55234894099533), ('p-value', 0.0)]
```

The results obtained showed that the residuals are not normally distributed.

QQ Plot of the residuals.



14. Sequential Feature Selection/Elimination.

M-4 Sequential Feature selection taking Hyper Parameter K as 7.

```
linreg=LinearRegression()
linreg_forward=sfs(estimator=linreg,k_features=7,forward=True,verbose=0,scoring='r2')
linreg_forward.fit(x_train,y_train)
```

SequentialFeatureSelector
estimator: LinearRegression
LinearRegression

```
print(linreg_forward.k_feature_names_)
```

```
('Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Ship Mode', 'Sub-Category', 'Order Priority')
```

Got the Name of the features selected by SFS Forward model and making a model with these features and will check on their performance.

```
Update_Score_Card("SFS_FORWARD","sfs_forward_features",sfs_forward_features,x_train,x_test,y_train,y_test)
```

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	Train_MSE	Test_MSE	Train_MAE	Test_MAE	Train_RMSE	Test_RMSE	Train_MAPE	Te
0	OLS	OLS_Model_1	0.725234	0.688087	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
1	OLS	OLS_Model_2	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2(
2	MLR	MLR_Model_3	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2(
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	2

The Overall performance of the model improved from other models. We will try to further tune the model and let us the performance of model .

M-5 Sequential Feature selection taking Hyper Parameter K as BEST.

Making the model using SFS forward technique and k-features as Best. SFS uses a unique criteria to select the features, it comes up with a specific order of features for the optimum performance.

```
linreg=LinearRegression()
Linreg_Forward_Best_4=sfs(estimator=linreg,k_features="best",verbose=0,forward=True,scoring="r2")
Linreg_Forward_Best_4.fit(x_train,y_train)
```

SequentialFeatureSelector
estimator: LinearRegression
LinearRegression

```
print(Linreg_Forward_Best_4.k_feature_names_)
```

```
('Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Ship Mode', 'Market', 'Region', 'Sub-Category', 'Product Name', 'Order Pri
ority')
```

```
print(Linreg_Forward_Best_4.k_score_)
```

```
0.7160730898820484
```

Making a model in the same order as we got from model SFS forward selection k-features=Best. And checking the performance of the model.

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	Train_MSE	Test_MSE	Train_MAE	Test_MAE	Train_RMSE	Test_RMSE	Train_MAPE	Te
0	OLS	OLS_Model_1	0.725234	0.688087	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
1	OLS	OLS_Model_2	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2
2	MLR	MLR_Model_3	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	2
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2

There is little improvement in Best Features then taking K=7.

M-6 SFS Backward Elimination k features=7

```
linreg=LinearRegression()
linreg_backward=sfs(estimator=linreg,forward=False, k_features=7,verbose=0,scoring="r2")
linreg_backward.fit(x_train,y_train)
```

SequentialFeatureSelector
estimator: LinearRegression
LinearRegression

```
print(linreg_backward.k_score_)
```

```
0.7157125382670905
```

```
print(linreg_backward.k_feature_names_)
```

```
('Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Ship Mode', 'Sub-Category', 'Order Priority')
```

On the basis of features names from SFS Backward elimination we are again making a linear model and checking it on all the required parameters.

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	Train_MSE	Test_MSE	Train_MAE	Test_MAE	Train_RMSE	Test_RMSE	Train_MAPE	Te
0	OLS	OLS_Model_1	0.725234	0.688087	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	25
1	OLS	OLS_Model_2	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2(
2	MLR	MLR_Model_3	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2(
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	25
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	25
5	SFS Backward	sfs_backward	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	25

There is not much improvement in the performance model but there is one observation here The order and the features selected by the models SFS Forward and SFS Backward are same And so does their performance.

M-7 SFS Backward Elimination k features=Best

```
linreg=LinearRegression()
linreg_backward_best=sfs(estimator=linreg,k_features="best",scoring="r2",verbose=0,forward=False)
linreg_backward_best.fit(x_train,y_train)
```

SequentialFeatureSelector
estimator: LinearRegression
LinearRegression

```
print(linreg_backward_best.k_score_)
```

```
0.7160730898820484
```

```
print(linreg_backward_best.k_feature_names_)
```

```
('Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Ship Mode', 'Market', 'Region', 'Sub-Category', 'Product Name', 'Order Pri
ority')
```

Using the SFS Backward elimination k features = Best we got the above order of the features and consecutively proceeding with the same to check the performance.

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	Train_MSE	Test_MSE	Train_MAE	Test_MAE	Train_RMSE	Test_RMSE	Train_MAPE	Te
0	OLS	OLS_Model_1	0.725234	0.688087	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
1	OLS	OLS_Model_2	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2
2	MLR	MLR_Model_3	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	2
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
5	SFS Backward	sfs_backward	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	2
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2

The overall performance is quite similar to that one of the previous models.

M-8 Recursive feature Elimination K=5

```
linreg=LinearRegression()
linreg_rfe=RFE(estimator=linreg,verbose=0,n_features_to_select=5)
linreg_rfe.fit(x_train,y_train)
```

RFE
estimator: LinearRegression
LinearRegression

```
test=pd.Series(data=linreg_rfe.ranking_,index=x.columns)
```

```
test[test==1].index
```

```
Index(['Quantity', 'Discount', 'Profit', 'Shipping Cost', 'Order Priority'], dtype='object')
```

Using Rfe model to select the features from the given list and given hyper parameter as 5 and got the above five features for model making.

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	Train_MSE	Test_MSE	Train_MAE	Test_MAE	Train_RMSE	Test_RMSE	Train_MAPE	Te
0	OLS	OLS_Model_1	0.725234	0.688087	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
1	OLS	OLS_Model_2	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2
2	MLR	MLR_Model_3	0.697576	0.659667	0.000400	0.000400	0.009800	0.010200	0.019100	0.021000	inf	2
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	2
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
5	SFS Backward	sfs_backward	0.724715	0.687700	0.000300	0.000400	0.009700	0.010100	0.018300	0.020100	inf	2
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126	0.000300	0.000400	0.009700	0.010000	0.018200	0.020100	inf	2
7	RFE	linreg_rfe	0.719553	0.682139	0.000300	0.000400	0.009800	0.010100	0.018400	0.020300	inf	2

The performance of the model is somewhat similar Rfe does not helped a lot in improving the performance.

M-9 K-fold Cross validation using cross Val score

Making the model and getting the values for the subsequent splits for the training data set.

```
linreg=LinearRegression()
linreg.fit(x_train,y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
scores_train=cross_val_score(estimator=linreg,X=x_train,y=y_train, scoring='r2',cv=10)
```

```
scores_train
```

```
array([0.72264261, 0.70558972, 0.72777492, 0.69869562, 0.67936119,
       0.78966719, 0.71882464, 0.73349746, 0.69545608, 0.7240309 ])
```

Printing the scores of different sub groups obtained from cross validation and analysing the maximum, minimum and average score.

```
print("Maximum Score",scores_train.max())
```

```
Maximum Score 0.78966718616814
```

```
print("Minimum Score",scores_train.min())
```

```
Minimum Score 0.6793611882404262
```

```
print("Average Score",scores_train.mean())
```

```
Average Score 0.719554032318938
```

M-10 Using Stochastic Gradient Descent

Making of model

```
y=df_tranformed["Sales"]
x=df_tranformed.drop("Sales",axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=1)
print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
x_train (17524, 17)
x_test (7511, 17)
y_train (17524,)
y_test (7511,)
```

```
sgd=SGDRegressor(random_state=1)
sgd.fit(x_train,y_train)
```

```
SGDRegressor(random_state=1)
```

```
Update_Score_Card("SGD Regressor","sgd",sgd,x_train,x_test,y_train,y_test)
```

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test
0	OLS	OLS_Model_1	0.725234	0.688087
1	OLS	OLS_Model_2	0.697576	0.659667
2	MLR	MLR_Model_3	0.697576	0.659667
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126
5	SFS Backward	sfs_backward	0.724715	0.687700
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126
7	RFE	linreg_rfe	0.719553	0.682139
8	SGD Regressor	sgd	-2436940625661056214272794312048640.000000	-225153488265582573805076888105779 2.000000	2949518675639666214328 31

- The values obtained by SGD regressor not at good.
- We use either OLS or Linear regression models for our Dataset.
- Sgd Regressor gave very high value for errors. So it is not recommended.
-

15. Regularization

One way to deal with the overfitting problem is by adding the `Regularization` to the model. It is observed that inflation of the coefficients cause overfitting. To prevent overfitting, it is important to regulate the coefficients by penalizing possible coefficient inflations. Regularization imposes penalties on parameters if they inflate to large values to prevent them from being weighted too heavily. In this section, we will learn about the three regularization techniques:

1. Ridge Regression
2. Lasso Regression
3. Elastic Net Regression

M-11 Using Ridge Regularization

Making of model using Ridge, according to ridge the alpha hyper parameter is set in such a way that it minimizes the effect of insignificant variables.

```
y=df_tranformed["Sales"]
x=df_tranformed.drop("Sales",axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=1)
print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
x_train (17524, 17)
x_test (7511, 17)
y_train (17524,)
y_test (7511,)
```

```
ridge=Ridge(alpha=1,max_iter=500,random_state=1)
ridge.fit(x_train,y_train)
```

```
Ridge(alpha=1, max_iter=500, random_state=1)
```

Performance Analysis

```
Update_Score_Card("Ridge Reg.","ridge",ridge,x_train,x_test,y_train,y_test)
```

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test
0	OLS	OLS_Model_1	0.725234	0.688087
1	OLS	OLS_Model_2	0.697576	0.659667
2	MLR	MLR_Model_3	0.697576	0.659667
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126
5	SFS Backward	sfs_backward	0.724715	0.687700
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126
7	RFE	linreg_rfe	0.719553	0.682139
8	SGD Regressor	sgd	-2436940625661056214272794312048640.000000	-22515348826558257380507688810577792.000000	2949518675639666214328312
9	Ridge Reg.	ridge	0.721129	0.699392

We got decent performance using Ridge and it is so far the best suited model for train and test.

M-12 Using Lasso Regularization

Making Of Model using Ridge. In Ridge the cost function is set in such a manner that the hyper parameter alpha given to the function completely ignores the effect of insignificant variables.

```
y=df_tranformed["Sales"]
x=df_tranformed.drop("Sales",axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=1)
print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
x_train (17524, 17)
x_test (7511, 17)
y_train (17524,)
y_test (7511,)
```

```
lasso=Lasso(random_state=1,alpha = 0.01, max_iter = 500)
lasso.fit(x_train,y_train)
```

```
Lasso(alpha=0.01, max_iter=500, random_state=1)
```

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	
0	OLS	OLS_Model_1	0.725234	0.688087	
1	OLS	OLS_Model_2	0.697576	0.659667	
2	MLR	MLR_Model_3	0.697576	0.659667	
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	
5	SFS Backward	sfs_backward	0.724715	0.687700	
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126	
7	RFE	linreg_rfe	0.719553	0.682139	
8	SGD Regressor	sgd	-2436940625661056214272794312048640.000000	-225153488265582573805076888105779 2.000000	2949518675639666214328 3
9	Ridge Reg.	ridge	0.721129	0.699392	
10	Lasso Reg	lasso	0.003090	0.002174	

- The performance of Lasso is not at all to be considered.
- It seems like lasso has almost dropped all the features.

M-13 Using Elastic Net Regularization

Making of Model Elastic Net act as combination of Ridge and Lasso . The results are calculated on the basis of combined effect of both.

```
y=df_tranformed["Sales"]
x=df_tranformed.drop("Sales",axis=1)
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=.3,random_state=1)
print("x_train",x_train.shape)
print("x_test",x_test.shape)
print("y_train",y_train.shape)
print("y_test",y_test.shape)
```

```
x_train (17524, 17)
x_test (7511, 17)
y_train (17524,)
y_test (7511,)
```

```
elasticnet=ElasticNet(alpha=.1,l1_ratio=.01,max_iter=500,random_state=1)
elasticnet.fit(x_train,y_train)
```

```
ElasticNet(alpha=0.1, l1_ratio=0.01, max_iter=500, random_state=1)
```

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	
0	OLS	OLS_Model_1	0.725234	0.688087	
1	OLS	OLS_Model_2	0.697576	0.659667	
2	MLR	MLR_Model_3	0.697576	0.659667	
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	
5	SFS Backward	sfs_backward	0.724715	0.687700	
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126	
7	RFE	linreg_rfe	0.719553	0.682139	
8	SGD Regressor	sgd	-24369406256610562142727943120486 40.000000	-22515348826558257380507688810577 92.000000	2949518675639666214328 3
9	Ridge Reg.	ridge	0.721129	0.699392	
10	Lasso Reg	lasso	0.003090	0.002174	
11	Elastic Net Reg	elasticnet	0.070041	0.070247	

As it was very clear that Elastic Net act as a combination of Both Ridge and Lasso, So does the performance of concludes. The results lies in between the Performance of Ridge and Lasso.

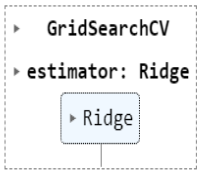
16. Tunning of Hyper Parameters

M-14 Using Grid Search CV

Grid Search CV is used to tune the hyper parameters given by us. It computes the performance of model according to the possible permutations of tunning parameters supplied to the model and gives us the optimum results.

```
tuned_paramaters = [{'alpha':[1e-15, 1e-10, 1e-8, 1e-4,1e-3, 1e-2, 0.1, 1, 5, 10, 20, 40, 60, 80, 100]}]
ridge = Ridge()
ridge_grid = GridSearchCV(estimator = ridge,
                          param_grid = tuned_paramaters,
                          cv = 10)
```

```
ridge_grid.fit(x_train, y_train)
```




```
print("Best Parameters",ridge_grid.best_params_)
```

```
Best Parameters {'alpha': 0.1}
```

Here we are using Ridge regularization and hyper parameter for same is alpha and after Grid Search CV the Value of Alpha Comes out 0.1

Again Making the model with the optimised value and checking for performance.

```
Score_Card
```

	Algorithm_Name	Model_Name	R_Square_train	R_Square_test	
0	OLS	OLS_Model_1	0.725234	0.688087	
1	OLS	OLS_Model_2	0.697576	0.659667	
2	MLR	MLR_Model_3	0.697576	0.659667	
3	SFS_FORWARD	sfs_forward_features	0.724715	0.687700	
4	SFS Forward Best	sfs_forward_best	0.725160	0.688126	
5	SFS Backward	sfs_backward	0.724715	0.687700	
6	SFS Backward Best	sfs_backward_best	0.725160	0.688126	
7	RFE	linreg_rfe	0.719553	0.682139	
8	SGD Regressor	sgd	-2436940625661056214272794312048640.000000	-225153488265582573805076888105779 2.000000	2949518675639666214328 3
9	Ridge Reg.	ridge	0.721129	0.699392	
10	Lasso Reg	lasso	0.003090	0.002174	
11	Elastic Net Reg	elasticnet	0.070041	0.070247	
12	Ridge Tuned	ridge_cv	0.725167	0.690023	

So far the best value of R-Square is obtained by the Ridge model using tuned parameter.

17.Implications: -

We have tried to study the data of the Global Store which is Hyper Market and dealing with thousands of products. The sales volume, the number of products, countries, states and regions where the store is operating is quite high. We have tried to analyse the sales pattern of the store, we analysed the data from customer point of view, from product point of view come up with best top selling products, we categorised the customer on the basis of frequency of their orders so that suitable offers can be made to low frequent customers. And using all the information come up with a model.

18.Limitations: -

We have tried a lot of models and tried different methods to make a best model although despite all the methods and hyper tunning the best we can explain is about 72 % of the patters in the data for our independent variable Sales which indicates there is scope of some more relevant features that should be considered for the further computations.

19.References: -

https://www.researchgate.net/post/What_is_the_major_meaning_of_PCs_in_Principal_Component_Analysis
https://onlinelibrary.wiley.com/doi/10.1111/poms.13717
https://www.researchgate.net/publication/353923375_The_Impact_of_Covid-19_Lockdown_on_General_Trade_Grocery_Stores_Kirana_Stores_and_Independent_Self_Service_Stores_in_India

https://www.census.gov/library/stories/2022/04/ecommerce-sales-surged-during-pandemic.html
https://www.analyticsvidhya.com/blog/2022/02/exploratory-data-analysis-in-python/#h2_10
https://towardsdatascience.com/top-machine-learning-algorithms-for-classification-2197870ff501
https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/
https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm