Machine Learning Project: Superstore Data Analysis

Submitted by

ATHIRA VALSAN

ABSTRACT

Most organizations store the data they generate during their daily operations. Analyzing this data to extract crucial information is becoming an important part of the decision-making process. The sample store dataset includes data for the Sales of multiple products sold by a company along with subsequent information related to geography, Product categories, and subcategories, sales, and profits, segmentation among the consumers, etc. This is a sample superstore dataset, a kind of simulation where you perform extensive data analysis to deliver insights on how the company can increase its profits while minimizing the losses. Our task is to analyze the sales data and identify weak areas and opportunities for Super Store to boost business growth. Throughout this project, we will employ various machine learning techniques, statistical analysis, and data visualization to explore and analyze the superstore data. Moreover, this project aims to provide a clear and structured workflow, allowing fellow data enthusiasts to learn from our methodology, replicate our analysis, and further contribute to the field.

CONTENT

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

In today's highly competitive retail landscape, businesses strive to gain a competitive edge by leveraging data-driven insights to optimize their operations and enhance customer experiences. One such business, a fictional superstore, aims to harness the power of machine learning and data analysis techniques to unlock valuable patterns and trends within their sales and customer data. This Kaggle Machine Learning project delves into the vast dataset provided by the superstore, with the objective of extracting meaningful insights and building predictive models to drive decision-making and operational efficiency.

By conducting a comprehensive analysis on this rich dataset, we aim to uncover hidden patterns, understand customer behavior, identify influential factors affecting sales performance, and develop accurate predictive models. These insights will enable the superstore to make better decisions, optimize its inventory management, enhance customer targeting, and ultimately maximize its sales revenue and profitability.

PROBLEM STATEMENT

The sample Dataset includes data for the Sales of multiple products sold by a company along with subsequent information related to geography, Product categories, and subcategories, sales, and profits, segmentation among the consumers, etc. Here we are trying to perform extensive data analysis to deliver insights on how the company can increase its profits while minimizing the losses and also understand which products, regions, categories and customer segments they should target or avoid. Our task is

to analyse the sales data and identify weak areas and opportunities for Super Store to boost business growth. Analyzing this data to extract crucial information and to become an important part of the decision-making process. Our task is to analyse the sales data and identify weak areas and opportunities for Super Store to boost business growth.

1. Which state yields the highest profit and highest sales?
2. Which Sub Category yields the highest profit and highest sales?
3. Name of the customers providing highest sales and profit?
4. Most preferred Shipping Mode, and its relationship with sales and profit?
5. Identifying the segment which contributes to sales and yields higher profit?
6. Which region generates the most sales?
7. Identify the city contributing the to sales and profit?
8. Which category of products generates the highest revenue and profit?
9. What are the top-selling products in the superstore?
10. Most commonly given discount %?
11. Most commonly given discount %?
12. What is the profit trend over time (monthly, yearly)?
13. Does delivery time affect sales and profit?

STUDY OF EXISTING SYSTEMS

EDA: Superstore Dataset Complete Analysis | Plotly:
Author: OLIEKSII ZHUKOV - In this project he did the data analysis part and applied machine learning algorithms to predict the distribution of total sales, total profit, Profit margins, shipping mode,various segments by order, Sub Categories

IDENTIFICATION OF GAPS IN EXISTING SYSTEMS

EDA: Superstore Dataset Complete Analysis | Plotly:
Author: OLIEKSII ZHUKOV-Only EDA is being applied, we need to apply the machine learning algorithm to make predictions. Also find the best fit model.

TOOLS AND TECHNOLOGIES USED TO IMPLEMENT THE PROPOSED SOLUTION

Python

Pandas

Numpy

Mathplotlib

Seaborn

Sklearn

Juypter Notebook

CHAPTER 2

DATA FEATURES AND PREDICTION:

- Row ID - Unique ID for each row.
- Order ID - Unique Order ID for each Customer.
- Order Date - Order Date of the product.
- Ship Date - Shipping Date of the Product.
- Ship Mode - Shipping Mode specified by the Customer.
- Customer ID - Unique ID to identify each Customer.
- Customer Name - Name of the Customer.
- Segment - The segment where the Customer belongs.
- Country - Country of residence of the Customer.
- City - City of residence of of the Customer.
- State - State of residence of the Customer.
- Postal Code - Postal Code of every Customer.
- Region - Region where the Customer belong.
- Product ID - Unique ID of the Product.
- Category - Category of the product ordered.
- Sub-Category - Sub-Category of the product ordered.
- Product Name - Name of the Product
- Sales - Sales of the Product
- Quantity - Quantity of the Product.
- Discount - Discount provided.
- Profit - Profit/Loss incurred.

Note:-

As we can see there are 3 types of datatypes.object: data is in categorical format like Order ID,Product Name, State, etc.

- int64: data is in numerical format like Postal code, Quantity, row ID.
- float64: data is in decimal format like Sales, Discount, Profit.

CHAPTER 3

METHODOLOGY

DATA CLEANING AND PRE-PROCESSING:

Data cleaning and preprocessing play a crucial role in the success of any machine learning project. In order to build accurate and reliable models, it is essential to ensure that the data used for training is of high quality, consistent, and free from errors and inconsistencies. This process, often referred to as data cleaning or data preprocessing, involves several steps aimed at preparing the data for analysis and modeling.

The first step in data cleaning is to address missing values. Missing data can significantly impact the performance of machine learning models, as they can introduce bias and affect the accuracy of predictions. available data.

Another crucial aspect of data cleaning is handling outliers. Outliers are data points that deviate significantly from the majority of the data and can distort the model's training process.

Data preprocessing also involves dealing with categorical variables. Machine learning models typically work with numerical data, so categorical variables need to be encoded into a numerical format.

Furthermore, data cleaning may involve removing duplicate entries, handling inconsistent or erroneous data, and ensuring data consistency and integrity throughout the dataset. This step helps to improve the reliability and accuracy of the models built on the cleaned data.

MACHINE LEARNING ALGORITHMS:

Logistic Regression:

Logistic regression is a widely used statistical modeling technique for binary classification problems, where the target variable has two possible outcomes. It is a linear model that applies the logistic function (also known as the sigmoid function) to transform the linear combination of input features into a probability score. The logistic function maps the output to a range between 0 and 1, allowing the interpretation of the output as the probability of belonging to a particular class. The model is trained using maximum likelihood estimation, optimizing the parameters to maximize the likelihood of the observed outcomes.

Decision Tree:

Decision trees are a popular supervised machine learning algorithm used for both classification and regression tasks. They are constructed by recursively splitting the data based on feature thresholds to create a hierarchical structure of decision nodes. Each internal node represents a feature and its corresponding splitting criteria, while leaf nodes represent the predicted outcome or value. Decision trees are interpretable and provide insights into the decision-making process by identifying the most important features and their relative importance. They can handle both categorical and numerical features, making them versatile for a wide range of applications.

Random Forest:

Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It leverages the concept of bagging by training each decision tree on a different subset of the training data. Random Forest reduces overfitting by averaging the predictions of multiple trees, resulting in a more robust and accurate model. It can handle both classification and regression tasks, making it a versatile

algorithm for various applications. Random Forest also provides insights into feature importance, allowing for feature selection and understanding the impact of variables on the predictions.

KNN:

K-Nearest Neighbors (KNN) is a simple yet powerful machine learning algorithm used for classification and regression tasks. It makes predictions by identifying the K nearest data points in the feature space and assigning the majority class (for classification) or averaging the target values (for regression). KNN is a non-parametric algorithm, meaning it does not make assumptions about the underlying data distribution. The choice of K, the number of neighbors, affects the model's bias-variance tradeoff, with smaller K values leading to more flexible models. KNN is easy to understand and implement, but it can be computationally expensive for large datasets and requires appropriate scaling of features.

Naive Bayes:

Naive Bayes is a popular machine learning algorithm based on Bayes' theorem and the assumption of feature independence. It is primarily used for classification tasks and performs well with high-dimensional data. Naive Bayes calculates the posterior probability of each class based on the prior probabilities and the likelihood of the features. Despite its simplicity, Naive Bayes can be surprisingly effective and computationally efficient, making it suitable for large-scale applications. However, the assumption of feature independence may limit its performance in cases where dependencies exist among the features.

IMPLEMENTATION STEPS:

As we already discussed in the methodology section about some of the implementation details.The language used in this project is Python Programming.We are running python code in anaconda navigators, Jupyter notebook. Jupyter Notebook is much faster than other available IDE platforms for implementing ML algorithms.The main advantage of Juypter notebook is that while writing code, its really helpful for Data visualization and plotting some visualization graphs like histogram , heatmap for correlation matrics etc.

The implementation steps of a machine learning model generally involve the following:

Data Preparation: Collect and preprocess the data by cleaning, transforming, and formatting it to ensure it is suitable for analysis. This includes handling missing values, outliers, and categorical variables, as well as splitting the data into training and testing sets.

Model Selection: Choose the appropriate machine learning algorithm based on the problem type (classification, regression, clustering, etc.) and the characteristics of the data. Consider factors such as interpretability, scalability, and performance.

Model Training: Train the selected model using the training dataset. This involves feeding the algorithm with the input features and their corresponding target variables to learn the underlying patterns and relationships. Adjust hyperparameters to optimize model performance if necessary.

Model Evaluation: Assess the performance of the trained model using evaluation metrics such as accuracy, precision, recall, F1-score, or mean squared error, depending on the problem type. Evaluate the model's generalization ability by testing it on the unseen testing dataset.

Model Optimization: Fine-tune the model to improve its performance. This can involve adjusting hyperparameters, feature selection, or applying regularization techniques to prevent overfitting. Cross-validation can be used to estimate the model's performance on unseen data.

Model Deployment: Once the model has been trained and optimized, it can be deployed for real-world use. This may involve integrating the model into a larger software system or creating an API for others to access and utilize the model's predictions.

LIBRARIES USED:

- import pandas as pd
- import numpy as np
- import matplotlib.pyplot as plt
- import seaborn as sns
- import warnings
- warnings.filterwarnings("ignore")
- from sklearn.preprocessing import LabelEncoder
- from sklearn.model_selection import train_test_split
- from sklearn.tree import DecisionTreeClassifier
- from sklearn.ensemble import RandomForestClassifier
- from sklearn.naive_bayes import GaussianNB
- from sklearn.neighbors import KNeighborsClassifier
- from sklearn.metrics import accuracy_score
- from sklearn.metrics import confusion_matrix
- from sklearn.metrics import plot_confusion_matrix
- from sklearn.metrics import classification_report

Reading the dataset

In [2]:
```
df=pd.read_csv("SampleSuperstore.csv")
```

In [3]:
```
df
```

Out[3]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country
0	1	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States
1	2	CA-2016-152156	11/8/2016	11/11/2016	Second Class	CG-12520	Claire Gute	Consumer	United States

Checking the Data Attributes:

In [4]:
```
df.shape
```

Out[4]: (9994, 21)

In [5]:
```
df.columns
```

Out[5]:
```
Index(['Row ID', 'Order ID', 'Order Date', 'Ship Date', 'Ship Mode',
       'Customer ID', 'Customer Name', 'Segment', 'Country', 'City', 'State',
       'Postal Code', 'Region', 'Product ID', 'Category', 'Sub-Category',
       'Product Name', 'Sales', 'Quantity', 'Discount', 'Profit'],
      dtype='object')
```

In [6]:
```
df.dtypes
```

Out[6]:
```
Row ID           int64
Order ID        object
Order Date      object
Ship Date       object
Ship Mode       object
Customer ID     object
Customer Name   object
Segment         object
Country         object
City            object
State           object
Postal Code      int64
Region          object
Product ID      object
Category        object
Sub-Category    object
Product Name    object
Sales          float64
Quantity         int64
Discount       float64
Profit         float64
dtype: object
```

As we can see there are 3 datatypes:

- object: data is in categorical format like Order ID,Product Name, State, etc.
- int64: data is in numerical format like Postal code, Quantity, row ID.
- float64: data is in decimal format like Sales, Discount, Profit.

Preparing data:

- We can see that Ship Date and Order Date are in object format so we have to convert it into datetime format.

In [7]:
```
df['Ship Date']=pd.to_datetime(df['Ship Date'],format='%m/%d/%Y')
df['Ship Date'].head()
```

Out[7]:
```
0   2016-11-11
1   2016-11-11
2   2016-06-16
3   2015-10-18
4   2015-10-18
Name: Ship Date, dtype: datetime64[ns]
```

In [8]:
```
df['Order Date']=pd.to_datetime(df['Order Date'],format='%m/%d/%Y')
df['Order Date'].head()
```

Out[8]:
```
0   2016-11-08
1   2016-11-08
2   2016-06-12
3   2015-10-11
4   2015-10-11
Name: Order Date, dtype: datetime64[ns]
```

In [9]:
```
df['OrderY']=df['Order Date'].dt.year
df['OrderM']=df['Order Date'].dt.month
df['OrderD']=df['Order Date'].dt.day
```

Handling Missing Values:

In [10]:
```
df.isnull().sum()
```

Out[10]:
```
Row ID          0
Order ID        0
Order Date      0
Ship Date       0
Ship Mode       0
Customer ID     0
Customer Name   0
Segment         0
Country         0
City            0
State           0
Postal Code     0
Region          0
Product ID      0
Category        0
```

```
OrderM      0
OrderD      0
dtype: int64
```

As we can see there is no null entry which is good for us.

Exploratory Data Analysis:

Which country do they have sales in ?

In [11]:
```
df['Country'].value_counts()
```

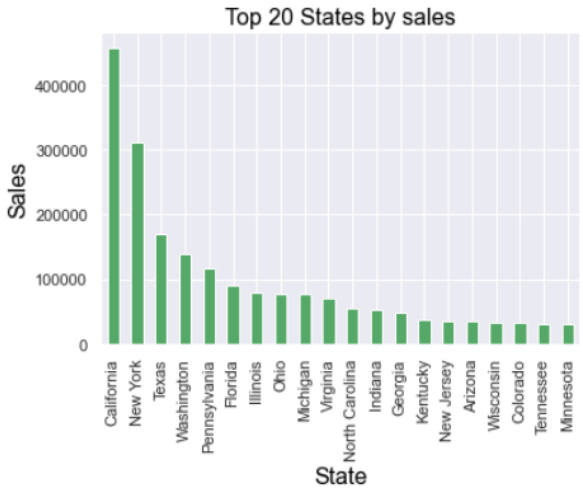
Out[11]:
```
United States    9994
Name: Country, dtype: int64
```

In [12]:
```
f1={"Family":"Arial","color":"black","size":16}
sns.set_theme()
```

Observation: All the sales are happening in the United States Of America.

Top 20 states having highest sales

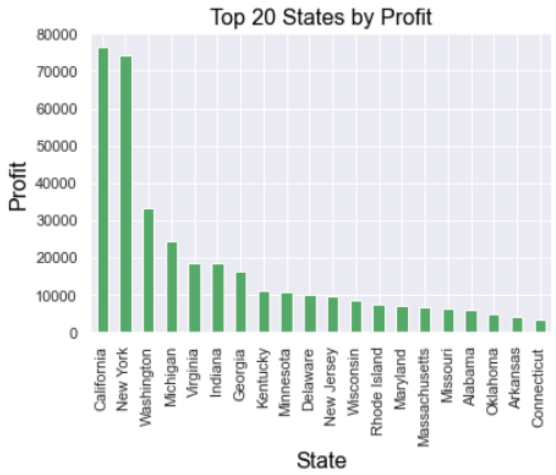
In [13]:
```
df.groupby('State')['Sales'].sum().sort_values(ascending=False).head(20).plot.bar(color="g")
plt.xlabel("State",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Top 20 States by sales",fontdict=f1)
plt.show()
```



Observation: The above bar graph shows the list of top 20 states having the highest sales.First is "California",followed by "New York" and then"Texas" holding the first three positions.

Top 20 states by Profit

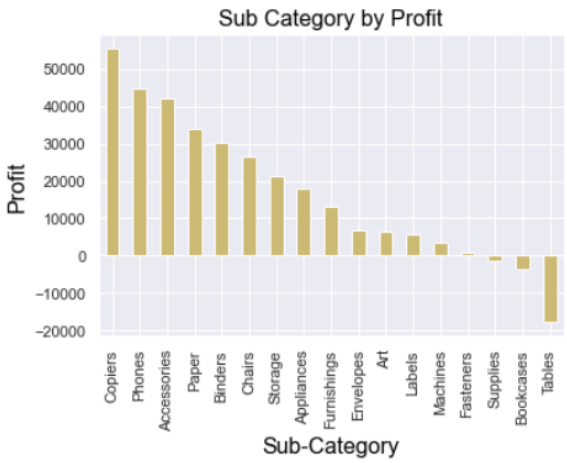
In [14]:
```
df.groupby('State')['Profit'].sum().sort_values(ascending=False).head(20).plot.bar(color="g")
plt.xlabel("State",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Top 20 States by Profit",fontdict=f1)
plt.show()
plt.show()
```



Observation: The above bar graph shows the list of top 20 states giving the highest profit. New York and California provide the highest profit amoung all the states.

Let us analyse the Sub Categories by Profit

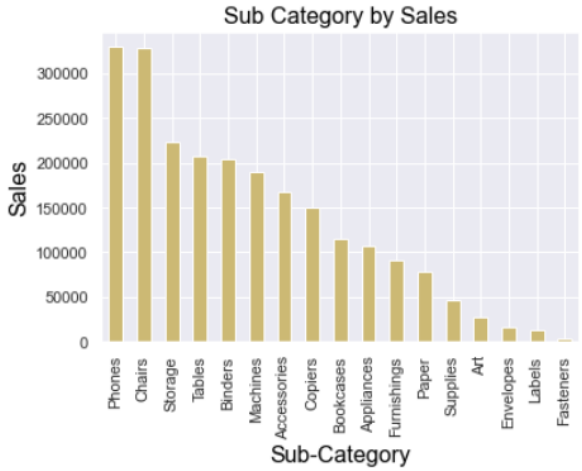
In [16]:
```
df.groupby('Sub-Category')['Profit'].sum().sort_values(ascending=False).plot.bar(color="y" )
plt.xlabel("Sub-Category",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Sub Category by Profit",fontdict=f1)
plt.show()
```



Observation:From the above bar graph we can conclude that, the products having the highest profits are Copies,Phones and Accessories.Where as Supplies, Bookcases and Tables are being sold at a loss.

Let us analyse the Sub Categories by Sales

In [15]:
```
df.groupby('Sub-Category')['Sales'].sum().sort_values(ascending=False).plot.bar(color="y" )
plt.xlabel("Sub-Category",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Sub Category by Sales",fontdict=f1)
plt.show()
```



Observation:The above bar graph shows the list of top 20 sub-categories by sales. Phones and chairs are the highest selling product.

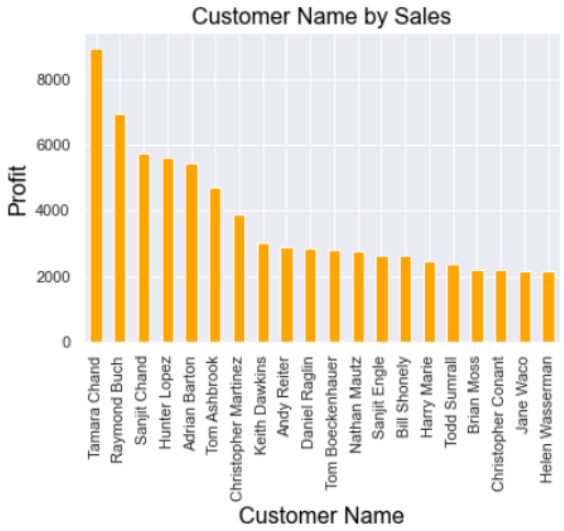
Let us analyse the top 20 customers by sales

In [17]:
```
df.groupby('Customer Name')['Sales'].sum().sort_values(ascending=False).head(20).plot.bar(color="orange")
plt.xlabel("Customer Name",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Customer Name by Sales",fontdict=f1)
plt.show()
```

Observation:From the above bar graph we can find the top 20 customers against sales. Mr.Sean Miller provides the highest sales.

Let us analyse the top 20 customers by profit

In [18]:
```
df.groupby('Customer Name')['Profit'].sum().sort_values(ascending=False).head(20).plot.bar(color="orange")
plt.xlabel("Customer Name",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Customer Name by Sales",fontdict=f1)
plt.show()
```



Observation:From the above bar graph we can find the top 20 customers w.r.t profit. Deals made by Ms.Tamara Chand has yield the higest profit

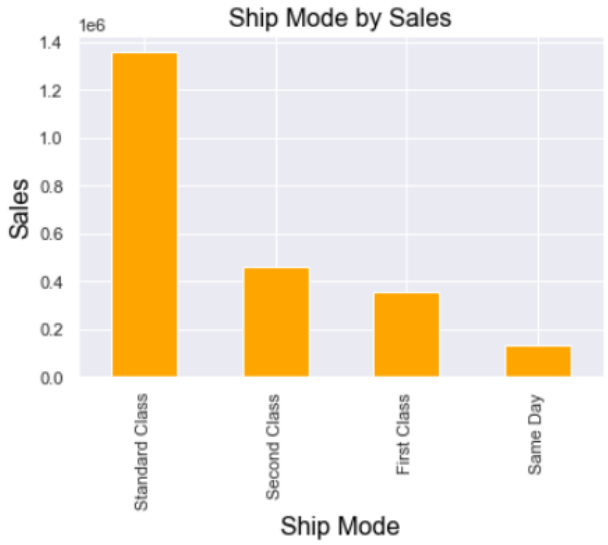
Let us analyse which is the most preferred shipping mode

In [19]:
```
df['Ship Mode'].value_counts().plot.bar(color="green")
plt.xlabel("Shipping Mode",fontdict=f1)
plt.ylabel("Rate",fontdict=f1)
plt.title("Most preferred Shipping mode",fontdict=f1)
plt.show()
```

Observation:From the above bar graph we can conclude that customers prefer "standard class ship mode" the most.

Analysing the relationship between shipping mode and sales

In [20]:
```
df.groupby('Ship Mode')['Sales'].sum().sort_values(ascending=False).plot.bar(color="orange")
plt.xlabel("Ship Mode",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Ship Mode by Sales",fontdict=f1)
plt.show()
```



Observation:From the above bar graph we can conclude that most sales prefer standard shipping mode.

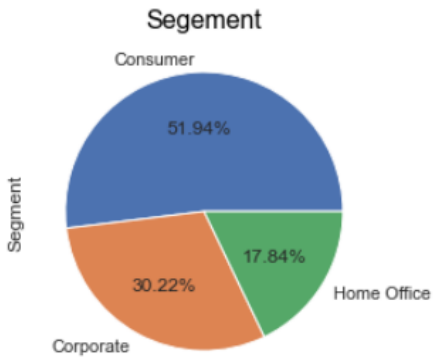
Analysing the relationship between shipping mode and profit

In [21]:
```
df.groupby('Ship Mode')['Profit'].sum().sort_values(ascending=False).plot.bar(color="orange")
plt.xlabel("Ship Mode",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Ship Mode by Profit",fontdict=f1)
plt.show()
```

Observation:From the above bar graph we can conclude that sales providing the highest profit prefer "standard class ship mode".

Let us analyse the contribution of the various segments

In [22]:
```
df['Segment'].value_counts().plot.pie(autopct='%1.2f%%')
plt.title("Segement",fontdict=f1)
plt.show()
```



Observation:From the above Pie chart we can conclude that more than half customer belong to the Consumer Segment.

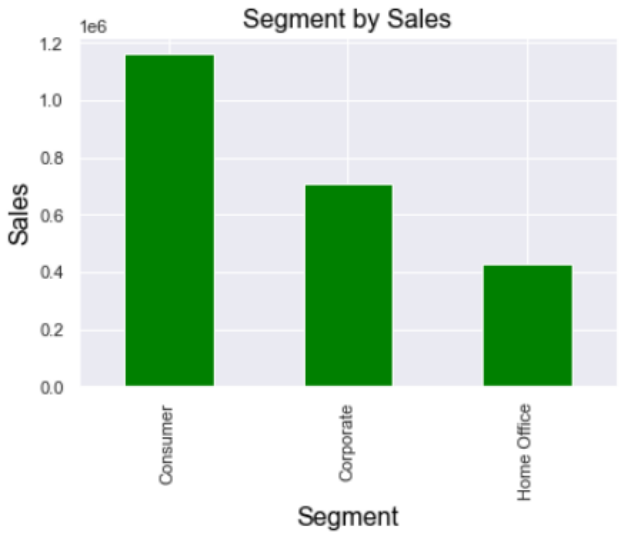
Let us analyse the segment vise profit

In [23]:
```
df.groupby('Segment')['Profit'].sum().sort_values(ascending=False).plot.bar(color="green")
plt.xlabel("Segment",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Segment by Profit",fontdict=f1)
plt.show()
```

Observation:From the above bar graph we can conclude that sales in the consumer segment provide highest profit.

Let us analyse the segment vise sales

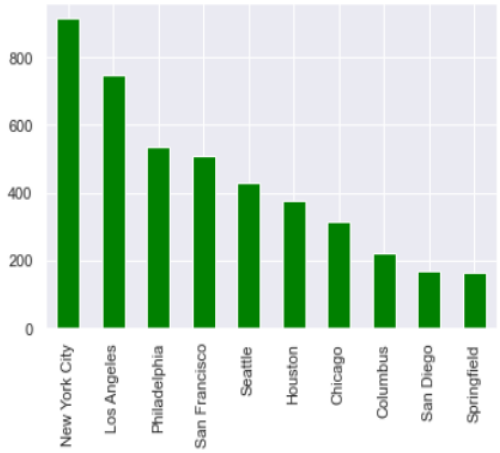
In [24]:
```
df.groupby('Segment')['Sales'].sum().sort_values(ascending=False).plot.bar(color="green")
plt.xlabel("Segment",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Segment by Sales",fontdict=f1)
plt.show()
```



Observation:From the above bar graph we can conclude that highest sales happen in the Consumer segment.

Let us analyse the top 10 cities

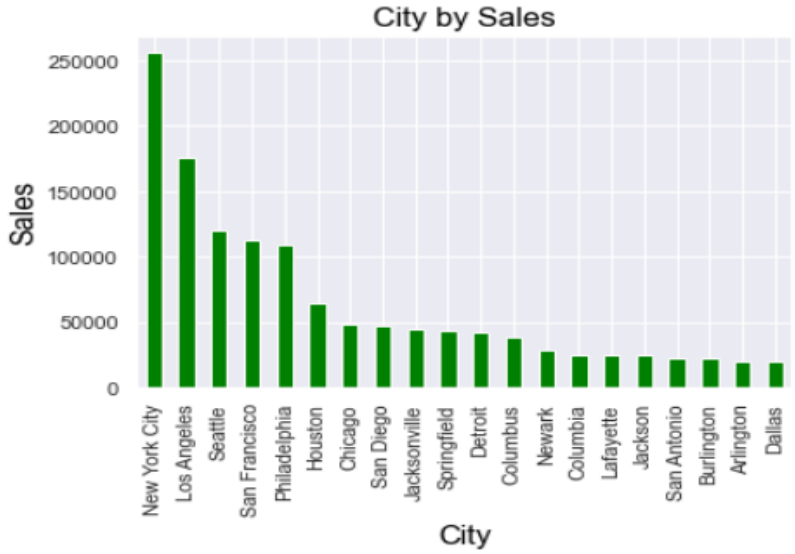
In [25]:
```
df['City'].value_counts().head(10).plot.bar(color="green")
plt.show()
```



Observation: Above bar graph represent top 10 cities present in the data and as we can see New York City has most orders placed followed by Los Angles, Philadelphia, etc.

Let us analyse the cities with respect to sales.

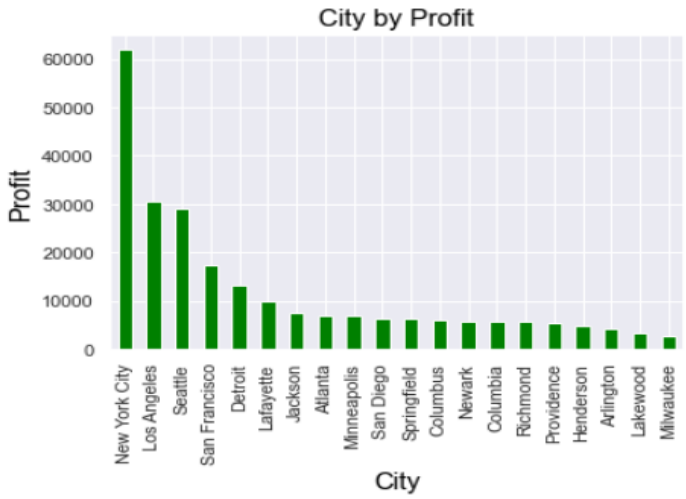
In [26]:
```
df.groupby('City')['Sales'].sum().sort_values(ascending=False).head(20).plot.bar(color="green")
plt.xlabel("City",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("City by Sales",fontdict=f1)
plt.show()
```



Observation: Above bar graph represent top 20 cities by sales .New York City leading the list.

Lets analyse the cities with respect to profit.

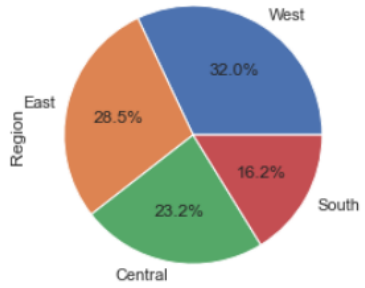
In [27]:
```
df.groupby('City')['Profit'].sum().sort_values(ascending=False).head(20).plot.bar(color="green")
plt.xlabel("City",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("City by Profit",fontdict=f1)
plt.show()
```



Observation: Above bar graph represent top 20 cities by profit .New York City providing the highest profit.

Let us analyse the contribution of various regions

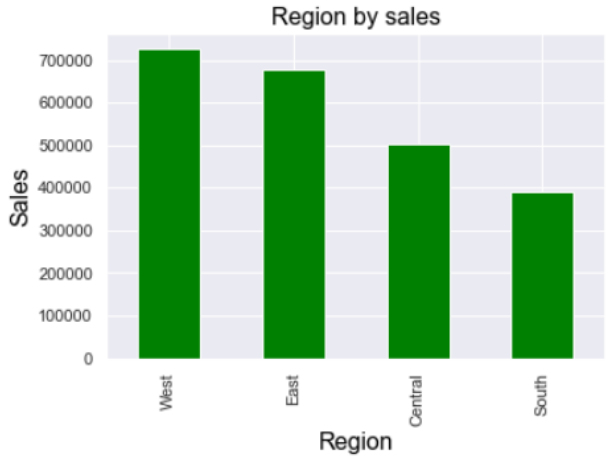
In [28]:
```
df['Region'].value_counts(normalize=True).plot.pie(autopct='%1.1f%%')
plt.show()
```



Observation: From the above pie chart we can conclude that most of the sales happen in West side of country followed by East, Central and South.

Let us analyse the region wise sales

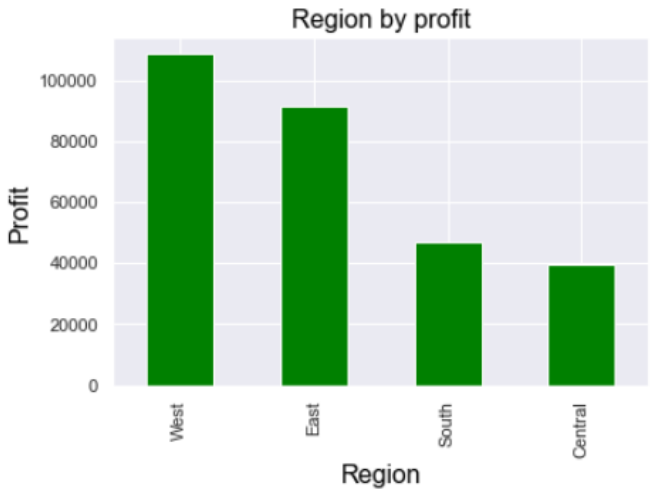
In [30]:
```
df.groupby('Region')['Sales'].sum().sort_values(ascending=False).head(20).plot.bar(color="green")
plt.xlabel("Region",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Region by sales",fontdict=f1)
plt.show()
```



Observation: The above bar grap shows that the highest sales happen in the west region.

Let us analyse the region wise profit

In [29]:
```
df.groupby('Region')['Profit'].sum().sort_values(ascending=False).head(20).plot.bar(color="green")
plt.xlabel("Region",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Region by profit",fontdict=f1)
plt.show()
```



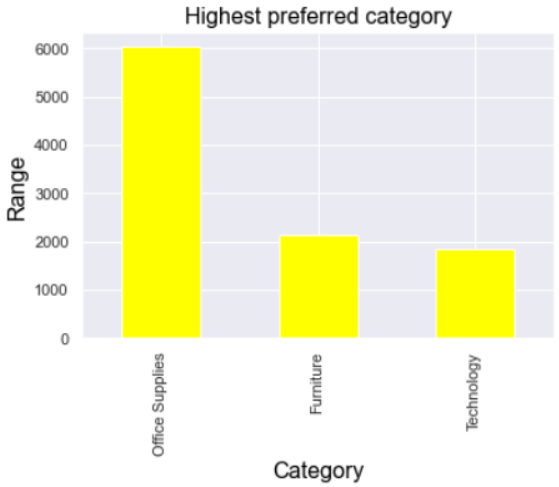
Observation: The above bar grap shows that the central region yeild the highest profit, followed by east, west and south.

Let us analyse the highest preferred category

In [31]:
```
df['Category'].unique()
```

Out[31]:
```
array(['Furniture', 'Office Supplies', 'Technology'], dtype=object)
```

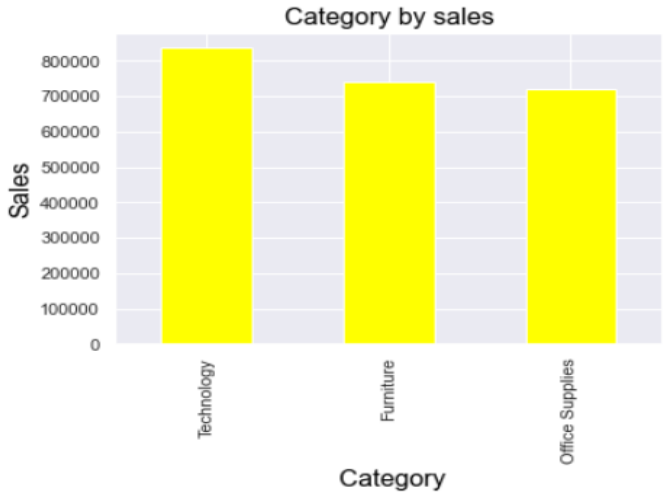
In [32]:
```
df['Category'].value_counts().plot.bar(color="yellow")
plt.xlabel("Category",fontdict=f1)
plt.ylabel("Range",fontdict=f1)
plt.title("Highest preferred category",fontdict=f1)
plt.show()
```



Observation:From the above bar chart we find that, their are 3 types of Sales Category in which Office Supplies leads the data. Furniture & Technology are on 2nd & 3rd place respectively.

Lets analyse the category wise sales

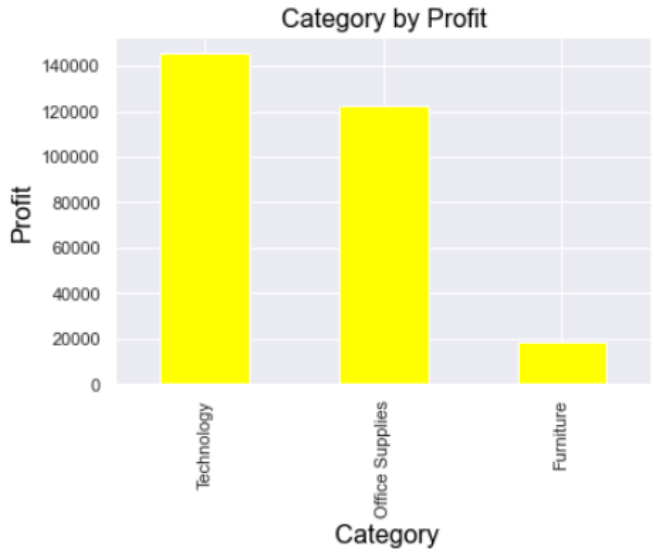
In [33]:
```
df.groupby('Category')['Sales'].sum().sort_values(ascending=False).plot.bar(color="yellow")
plt.xlabel(" Category",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Category by sales",fontdict=f1)
plt.show()
```



Observation: From the above bar chart we find that the highest sales happen in the category of "Technology"

Let us analyse the category wise profit

In [34]:
```
df.groupby('Category')['Profit'].sum().sort_values(ascending=False).plot.bar(color="yellow")
plt.xlabel(" Category",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Category by Profit",fontdict=f1)
plt.show()
```



Observation: From the above bar chart we find that the category yeilding the highest profit is Technology.

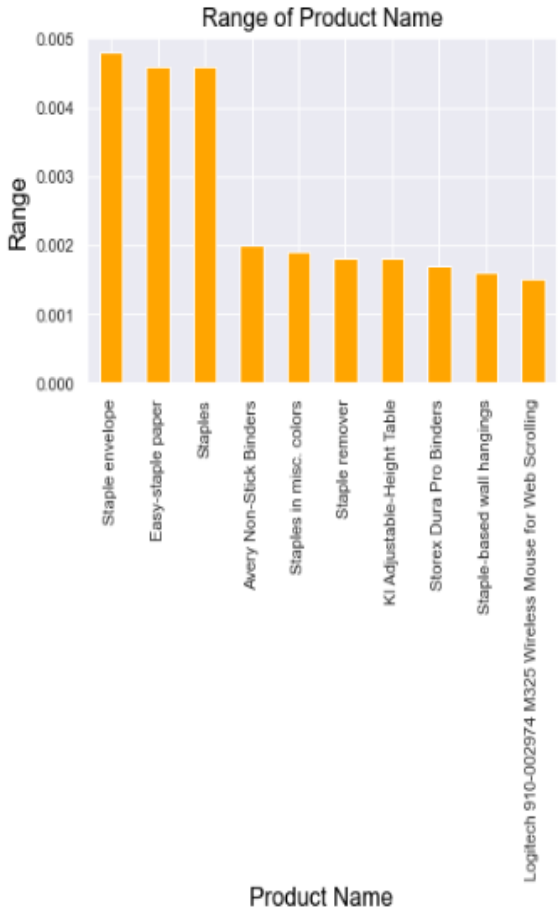
In [35]:
```
df_category=pd.DataFrame(df.groupby('Category')['Sales'].sum())
df_category['Profit']=pd.DataFrame(df.groupby('Category')['Profit'].sum())
print(df_category)
df_category.plot.scatter(x='Sales',y='Profit',color="r")
plt.xlabel("Sales",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Category by Profit and Sales",fontdict=f1)
plt.show()
```

Category	Sales	Profit
Furniture	741999.7953	18451.2728
Office Supplies	719047.0320	122490.8008
Technology	836154.0330	145454.9481

Observation: From the scatter plot we can conclude that Technology has highest sales and profit

Let us analyse the most preferred products

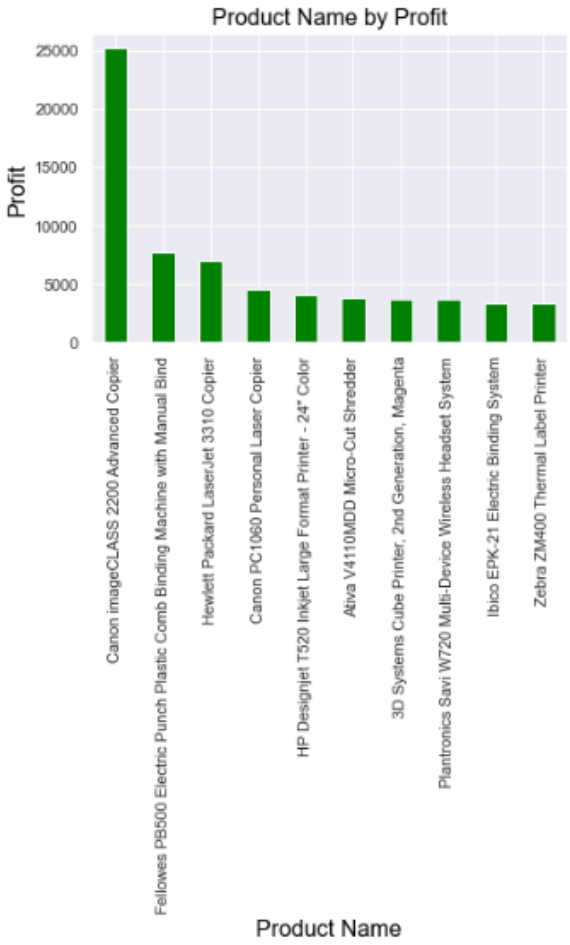
In [36]:
```
df['Product Name'].value_counts(normalize=True).head(10).plot.bar(color="orange")
plt.xlabel("Product Name",fontdict=f1)
plt.ylabel("Range",fontdict=f1)
plt.title("Range of Product Name",fontdict=f1)
plt.show()
```



Observation: From the above bar chart we can conclude that the most commonly prefered products are Staple envelop, easy-staple paper and Staples

Let us analyse the product yielding the highest profit

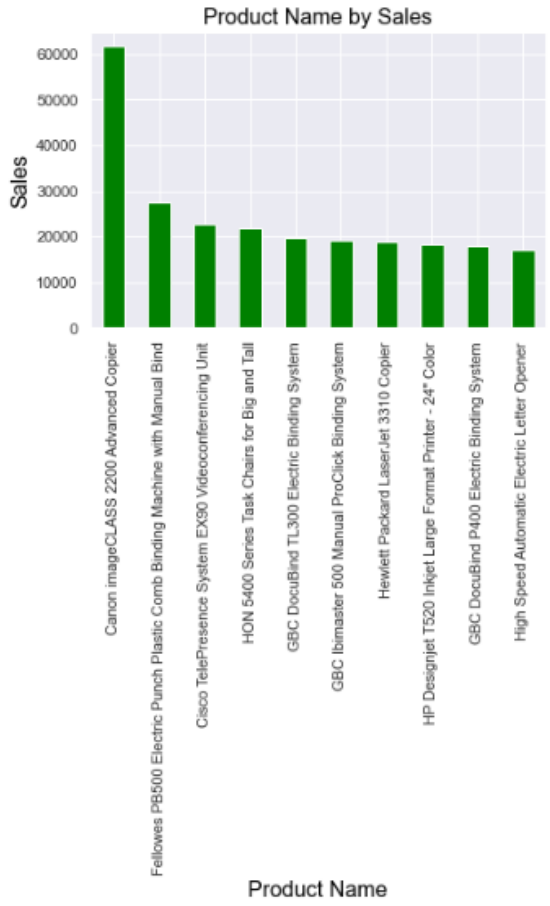
In [37]:
```
df.groupby('Product Name')['Profit'].sum().sort_values(ascending=False).head(10).plot.bar(color="green")
plt.xlabel("Product Name",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Product Name by Profit",fontdict=f1)
plt.show()
```



Observation: From the above bar chart we can conclude that the most profitable product is Canon imageCLASS 2200 Advanced Copier

Let us analyse the name of the highly sold product

In [38]:
```
df.groupby('Product Name')['Sales'].sum().sort_values(ascending=False).head(10).plot.bar(color="green")
plt.xlabel("Product Name",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Product Name by Sales",fontdict=f1)
plt.show()
```



Observation: From the above bar chart we can conclude that the most sold product is Canon imageCLASS 2200 Advanced Copier

Let us analyse the Discount variable

In [39]:
```
df['Discount'].value_counts().plot.bar(color="orange")
plt.xlabel("Discount",fontdict=f1)
plt.ylabel("Count",fontdict=f1)
plt.title("Discount ",fontdict=f1)
plt.show()
```

Observation:From the above bar graph we can conclude that mostly discount is provided between 0% to 20%.

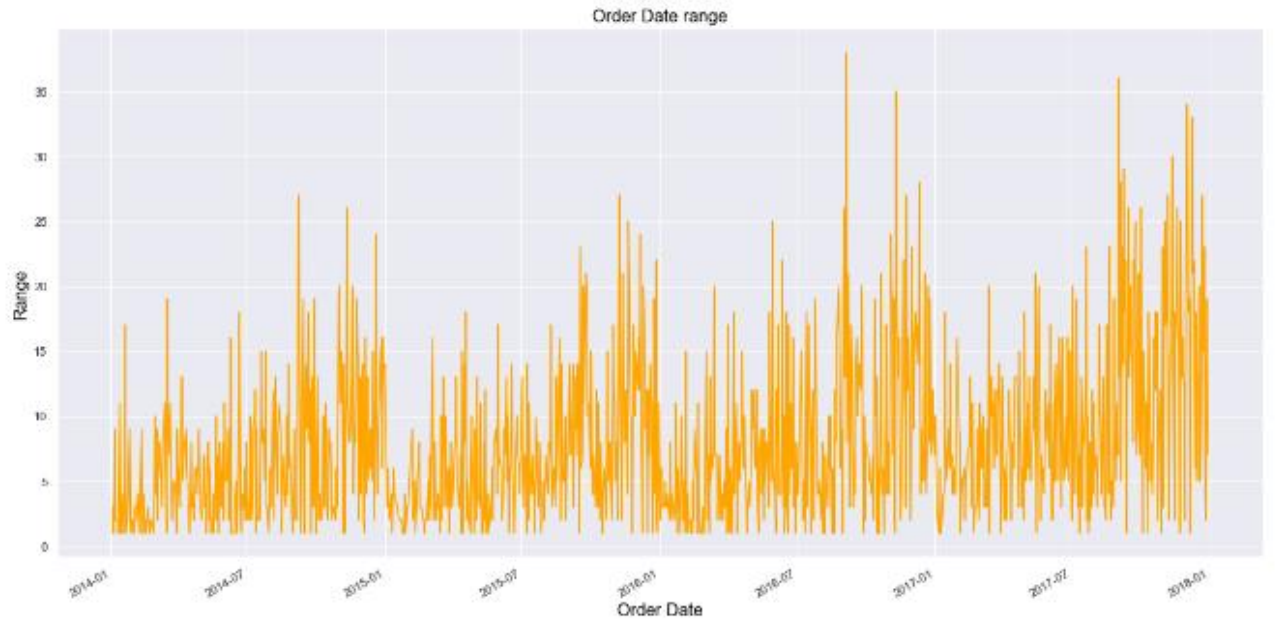
Let us analyse the quantity variable

In [40]:
```
df['Quantity'].value_counts().plot.bar(color="orange")
plt.xlabel("Quantity",fontdict=f1)
plt.ylabel("Count",fontdict=f1)
plt.title("Mostly ordered ",fontdict=f1)
plt.show()
```

Observation:From the above chart we can say that mostly the order quantity is between 2 and 3.

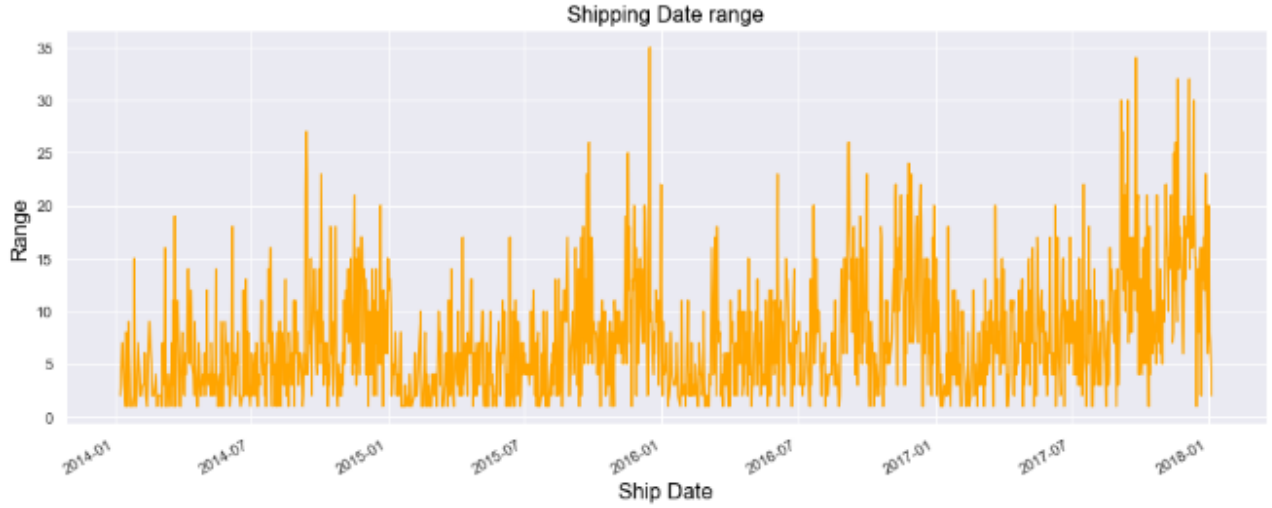
Let us visualize the range of Order date

In [41]:
```
df['Order Date'].value_counts().plot.line(figsize=(20,10),color="orange")
plt.xlabel("Order Date",fontdict=f1)
plt.ylabel("Range",fontdict=f1)
plt.title("Order Date range",fontdict=f1)
plt.show()
```



Let us visualize the range of Shipping date

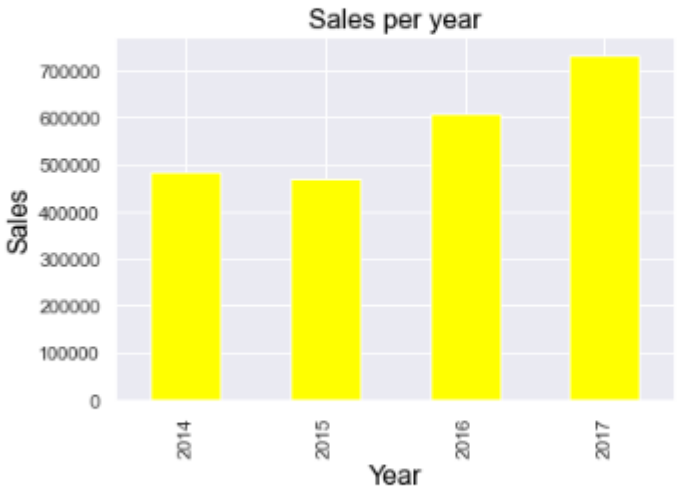
In [42]:
```
df['Ship Date'].value_counts().plot.line(figsize=(16,6),color="orange")
plt.xlabel("Ship Date",fontdict=f1)
plt.ylabel("Range",fontdict=f1)
plt.title("Shipping Date range",fontdict=f1)
plt.show()
```



Observation :From the above graphs we can see that Order Date and Ship Date values cover vast range.

Let us analyse the sales per year

In [43]:
```
df.groupby('OrderY')['Sales'].sum().plot.bar(color="yellow")
plt.xlabel("Year",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Sales per year",fontdict=f1)
plt.show()
```



Observation: From the above graph we can conclude that sales is increasing year by year.

Let us analyse the profit over years

In [44]:
```
df.groupby('OrderY')['Profit'].sum().plot.bar(color="yellow")
plt.xlabel("Year",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Profit per year",fontdict=f1)
plt.show()
```

Observation: Sales and Profit has been growing year by year.

Let us analyse the months having the highest sales

In [45]:
```
df.groupby('OrderM')['Sales'].sum().plot.bar(color="orange")
plt.xlabel("Year",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Sales per Month",fontdict=f1)
plt.show()
```

Observation: Highest sales happen in the months 11,12 and 9

Let us analyse the month having the highest profit

In [46]:
```
df.groupby('OrderM')['Profit'].sum().plot.bar(color="orange")
plt.xlabel("Year",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Profit per Month",fontdict=f1)
plt.show()
```

Observation: We can concluse that most profitable sales happen in the month of December

Let us analyse the sales over the years

In [47]:
```
plt.figure(figsize=(20,5))
df.groupby(['OrderY','OrderM'])['Sales'].sum().plot(color="green")
plt.xlabel("Year",fontdict=f1)
plt.ylabel("Sales",fontdict=f1)
plt.title("Yearly Sales",fontdict=f1)
plt.show()
```

Observation: Sales Order is increasing yearly.

Let us analyse the profit over the years

In [48]:
```
plt.figure(figsize=(20,5))
df.groupby(['OrderY','OrderM'])['Profit'].sum().plot(color="green")
plt.xlabel("Year",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Yearly profit",fontdict=f1)
plt.show()
```

Observation: Yearly profit is increasing.

Let us analyse if the delivery time is affecting the profit

In [49]:
```
df["deliverytime"]=df["Ship Date"]-df["Order Date"]
```

In [50]:
```
df.groupby('deliverytime')['Profit'].sum().plot.bar(color="orange")
plt.xlabel("Delivery time",fontdict=f1)
plt.ylabel("Profit",fontdict=f1)
plt.title("Delivery time v/s profit",fontdict=f1)
plt.show()
```

Observation:Delivery time of 4 days have the higest profit.

Let us analyse the numerical data using a heatmap-Sales,Quantity,Discount & Profit

In [51]:
```
num_df=df[['Sales','Quantity','Discount','Profit']]
sns.heatmap(num_df.corr(), annot=True)
plt.show()
```

Observation: We can conclude the heatmap that their is no strong corelation between the numerical variables

Data Preprocessing

In [52]:
```
# Lets drop the unwanted columns
df.drop(['OrderM', 'OrderY', 'OrderD'],axis=1,inplace=True)
```

Lets change the Profit column-Target Variable, of the dataset into three catorgorical values: Profit=1, Loss=0,Nill =2.

In [53]:
```
def typ(x):
    if x<0.00:
        x=1
    elif x>0.00:
        x=0
    else:
        x=2
    return x
```

In [54]:
```
df['Profit']=df['Profit'].apply(lambda x: typ(x))
df['Profit'].value_counts()
```

Out[54]:
```
0   8058
1   1871
2     65
Name: Profit, dtype: int64
```

In [55]:
```
x=df.drop("Profit",axis=1)
y=df['Profit']
```

In [56]:
```
# Lets use the Label Encoder to convert all the catogerical values to numerical values before applying the ML Algorithms
```

In [57]:
```
from sklearn.preprocessing import LabelEncoder
```

In [58]:
```
x_enc=x.copy()
for i in x.columns:
    le = LabelEncoder()
    x_enc[i]=le.fit_transform(x[i].values)
```

In [59]:
```
x_enc.head()
```

Out[59]:

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Cat
0	0	2500	864	929	2	143	166	0	0	194	...	217	2	12	0	
1	1	2500	864	929	2	143	166	0	0	194	...	217	2	55	0	
2	2	2296	732	787	2	237	201	1	0	266	...	517	3	946	1	
3	3	4372	519	568	3	705	687	0	0	153	...	170	2	319	0	
4	4	4372	519	568	3	705	687	0	0	153	...	170	2	1316	1	

5 rows × 21 columns

In [60]:
```
#Lets split our data set into train and test part using train_test_split
```

In [61]:
```
from sklearn.model_selection import train_test_split
```

In [62]:
```
x_train, x_test, y_train, y_test = train_test_split(x_enc,y, test_size=0.3)
```

Decision Tree

In [63]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [64]:
```
model_dt=DecisionTreeClassifier(criterion='gini', max_depth=9)
model_dt.fit(x_train, y_train)
```

Out[64]:
```
DecisionTreeClassifier(max_depth=9)
```

In [65]:
```
model_dt.score(x_train,y_train)*100 # Score of train data is 97.6%
```

Out[65]:
```
97.56969263759828
```

In [66]:
```
model_dt.score(x_test, y_test)*100 #Score of test data is 95.2%
```

Out[66]:
```
95.0650216738913
```

In [67]:
```
y_pred1=model_dt.predict(x_test)#y_predict
```

In [68]:
```
df_new_dt=pd.DataFrame({"Actual":y_test,"Predict":y_pred1})# lets create a dataframe of actual and predicted
df_new_dt
```

Out[68]:

	Actual	Predict
5746	0	0
2960	0	0
8090	0	0
2984	1	1
3101	0	0

In [69]:
```
# accuracy score
from sklearn.metrics import accuracy_score
```

In [70]:
```
score_dt=accuracy_score(y_test, y_pred1)*100
score_dt
```

Out[70]: 95.0650216738913

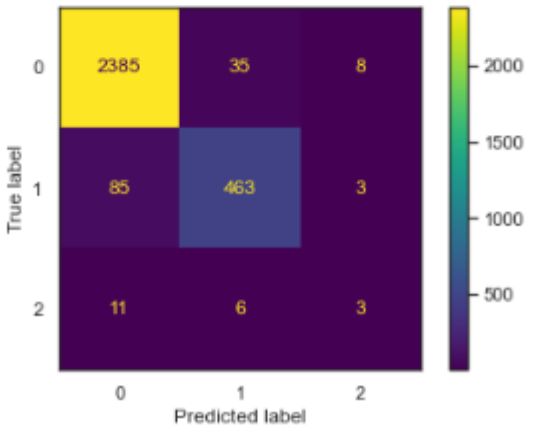
In [71]:
```
# confusion_matrix
from sklearn.metrics import confusion_matrix
```

In [72]:
```
confusion_matrix(y_test,y_pred1)
```

Out[72]:
```
array([[2385,   35,    8],
       [  85,  463,    3],
       [  11,    6,    3]], dtype=int64)
```

In [73]:
```
#plot_confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

In [74]:
```
sns.set_theme(style="white", palette=None)
plot_confusion_matrix(model_dt,x_test,y_test)
plt.show()
```



In [75]:
```
#classification_report
from sklearn.metrics import classification_report
```

In [76]:
```
report_dt =classification_report(y_test,y_pred1)
print(report_dt)
```

```
              precision    recall  f1-score   support

           0       0.96      0.98      0.97      2428
           1       0.92      0.84      0.88       551
           2       0.21      0.15      0.18        20

    accuracy                           0.95      2999
   macro avg       0.70      0.66      0.68      2999
weighted avg       0.95      0.95      0.95      2999
```

Random Forest

In [77]:
```
from sklearn.ensemble import RandomForestClassifier
```

In [78]:
```
model_rf=RandomForestClassifier(n_estimators=100)
```

In [79]:
```
model_rf.fit(x_train, y_train)
```

Out[79]:
```
RandomForestClassifier()
```

In [80]:
```
model_rf.score(x_train,y_train)*100 # training data score
```

Out[80]:
```
100.0
```

In [81]:
```
model_rf.score(x_test,y_test)*100 # test data score
```

Out[81]:
```
95.16505501833944
```

In [82]:
```
y_pred2=model_rf.predict(x_test)
y_pred2
```

Out[82]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [83]:
```
df_new_rf=pd.DataFrame({"Actual":y_test,"Predict":y_pred2})# lets create a dataframe of actual and predicted
df_new_rf
```

Out[83]:

	Actual	Predict
5746	0	0
2960	0	0
8090	0	0
2984	1	1
3101	0	0
...	...	...
1699	0	0
3194	0	0
6134	0	0
5592	0	0
5219	0	0

2999 rows × 2 columns

In [84]:
```
score_rf=accuracy_score(y_test, y_pred2)*100
score_rf
```

Out[84]:
```
95.16505501833944
```

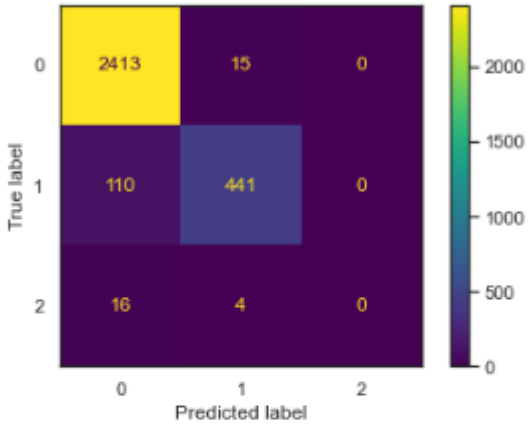
In [85]:
```
# confusion_matrix
from sklearn.metrics import confusion_matrix
```

In [86]:
```
confusion_matrix(y_test,y_pred2)
```

Out[86]:
```
array([[2413,   15,    0],
       [ 110,  441,    0],
       [  16,    4,    0]], dtype=int64)
```

In [87]:
```
#plot_confusion_matrix
from sklearn.metrics import plot_confusion_matrix
```

In [88]:
```
plot_confusion_matrix(model_rf,x_test,y_test)
plt.show()
```



In [89]:
```
#classification_report
from sklearn.metrics import classification_report
```

In [90]:
```
report_rf =classification_report(y_test,y_pred2)
print(report_rf)
```

```
              precision    recall  f1-score   support

           0       0.95      0.99      0.97      2428
           1       0.96      0.80      0.87       551
           2       0.00      0.00      0.00        20

    accuracy                           0.95      2999
   macro avg       0.64      0.60      0.61      2999
weighted avg       0.95      0.95      0.95      2999
```

Naive Bayes- GaussianNB

In [91]:
```
from sklearn.naive_bayes import GaussianNB
```

In [92]:
```
model_nb=GaussianNB()
```

In [93]:
```
model_nb.fit(x_train, y_train)
```

Out[93]:
```
GaussianNB()
```

In [94]:
```
model_nb.score(x_train, y_train) #train dataset score
```

Out[94]:
```
0.9355253752680486
```

In [95]:
```
model_nb.score(x_test, y_test) # test dataset score
```

Out[95]:
```
0.9416472157385796
```

In [96]:
```
y_pred3=model_nb.predict(x_test)
y_pred3
```

Out[96]: array([0, 0, 0, ..., 0, 0, 0], dtype=int64)

In [97]:
```
df_new_nb=pd.DataFrame({"Actual":y_test,"Predict":y_pred3})# lets create a dataframe of actual and predicted
df_new_nb
```

Out[97]:

	Actual	Predict
5746	0	0
2960	0	0
8090	0	0
2984	1	1
3101	0	0
...	...	...
1699	0	0
3194	0	0
6134	0	0
5592	0	0
5219	0	0

2999 rows × 2 columns

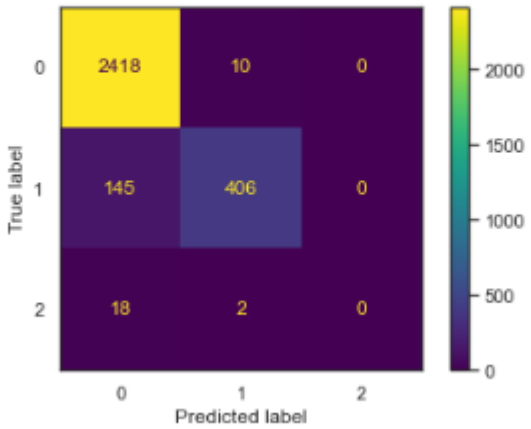
In [98]:
```
score_nb=accuracy_score(y_test, y_pred3)*100 # accuracy_score
score_nb
```

Out[98]: 94.16472157385796

In [99]:
```
confusion_matrix(y_test,y_pred3)#confusion_matrix
```

Out[99]:
```
array([[2418,   10,    0],
       [ 145,  406,    0],
       [  18,    2,    0]], dtype=int64)
```

In [100]:
```
plot_confusion_matrix(model_nb,x_test,y_test)#plot confusion matrix
plt.show()
```



In [101]:
```
report_nb =classification_report(y_test,y_pred3)# Classification report
print(report_nb)
```

```
              precision    recall  f1-score   support

           0       0.94      1.00      0.97      2428
           1       0.97      0.74      0.84       551
           2       0.00      0.00      0.00        20

    accuracy                           0.94      2999
   macro avg       0.64      0.58      0.60      2999
weighted avg       0.94      0.94      0.94      2999
```

K Nearest Neighbor

In [102]:
```
from sklearn.neighbors import KNeighborsClassifier
```

In [103]:
```
model_knn=KNeighborsClassifier(n_neighbors=21)
```

In [104]:
```
model_knn.fit(x_train,y_train)
```

Out[104]:
```
KNeighborsClassifier(n_neighbors=21)
```

In [105]:
```
model_knn.score(x_train,y_train)*100
```

Out[105]:
```
81.98713366690494
```

In [106]:
```
model_knn.score(x_test,y_test)*100
```

Out[106]:
```
81.36045348449483
```

In [107]:
```
y_pred4=model_knn.predict(x_test)
y_pred4
```

Out[107]:
```
array([0, 0, 0, ..., 0, 0, 0], dtype=int64)
```

In [108]:
```
df_new_rf=pd.DataFrame({"Actual":y_test,"Predict":y_pred4})# lets create a dataframe of actual and predicted
df_new_rf
```

Out[108]:

	Actual	Predict
5746	0	0
2960	0	0
8090	0	0
...	...	...

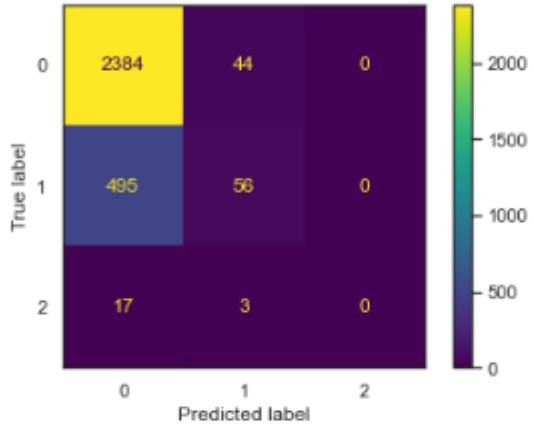
In [109]:
```
score_kn=accuracy_score(y_test, y_pred4)*100 # accuracy_score
score_kn
```

Out[109]:
```
81.36045348449483
```

In [110]:
```
confusion_matrix(y_test,y_pred4)#confusion_matrix
```

Out[110]:
```
array([[2384,   44,    0],
       [ 495,   56,    0],
       [  17,    3,    0]], dtype=int64)
```

In [111]:
```
plot_confusion_matrix(model_knn,x_test,y_test)#plot confusion matrix
plt.show()
```



In [112]:
```
report_knn =classification_report(y_test,y_pred4)# Classification report
print(report_knn)
```

	precision	recall	f1-score	support
0	0.82	0.98	0.90	2428
1	0.54	0.10	0.17	551
2	0.00	0.00	0.00	20
accuracy			0.81	2999
macro avg	0.46	0.36	0.36	2999
weighted avg	0.77	0.81	0.76	2999

Conclusion:

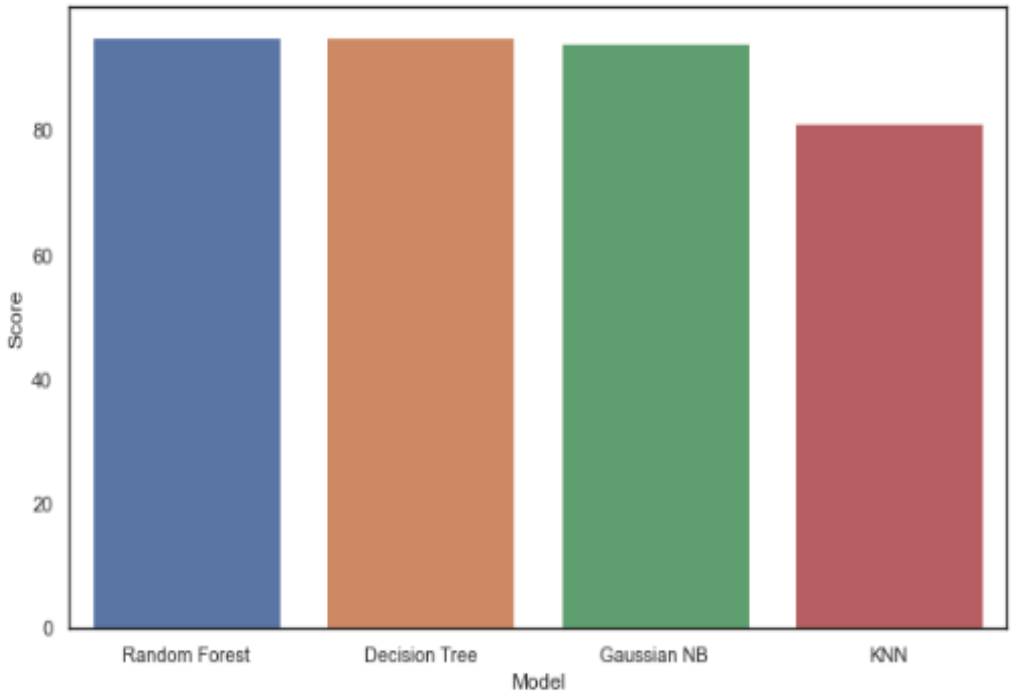
Finding the best method:

In [113]:
```
results=pd.DataFrame({'Model':['Decision Tree','Random Forest','Gaussian NB','KNN'],'Score':[score_dt,score_rf,score_nb,score_]
result_df=results.sort_values(by='Score',ascending=False)
result_df=result_df.reset_index(drop=True)
result_df.head()
```

Out[113]:

	Model	Score
0	Random Forest	95.165055
1	Decision Tree	95.065022
2	Gaussian NB	94.164722
3	KNN	81.360453

In [114]:
```
plt.figure(figsize=(10,6))
sns.barplot(x='Model',y='Score',data=result_df)
plt.show()
```



Accuracy Score of various ML models:

Decision Tree - 95.165%
Random Forest- 95.065%
GaussianNB - 94.16%
KNN-81.36%

Hence, we can conclude that Decision Tree and Random Forest algorithm has the highest Accuarcy Score and is the best suitable ML algorithm for this dataset.

CHAPTER 4

ANALYSIS OF THE RESULT:

1. We can infer from the analysis that the state having the highest sales and profit is California, New York and then followed by Washington.
2. Phones and chairs are the highest selling product. The products having the highest profits are Copies,Phones and Accessories. Whereas Supplies, Bookcases and Tables are being sold at a loss.
3. Mr.Sean Miller provides the highest sales. Ms.Tamara Chand has yield the highest profit.
4. The most preferred and profitable mode of shipping is Standard Class followed by Second Class and then First Class and Same Day.
5. The segment which contributes to highest sales and profit is Consumer, Corporate and then Home Office.
6. Most of the sales happen in West side of country followed by East, Central and South.
7. New York city is leading the list of cities leading sales and profit.
8. Among the customers segment Office Supplies is the most preferred domain .Whereas technology provides the highest sales and profit.
9. The most profitable product is Canon imageCLASS 2200 Advanced Copier. the most commonly preferred products are Staple envelop, easy-staple paper and Staples.
10. Mostly discount is provided between 0% to 20%.
11. Sales and Profit has been growing year by year. Highest sales happen in the months 11,12 and 9. We can concluse that most profitable sales happen in the month of December
12. Delivery time of 4 days have the highest profit.

OBSERVATION:

- Decision Tree model having the highest accuracy score of 95.165%.
- Random Forest model having the second highest accuracy of 95.065%.
- GaussianNB Model having the third highest accuracy of 94.16%.
- KNN Model having the lowest accuracy score of 81.36%.

CHAPTER 5

CONCLUSION:

To summarize, this Kaggle Machine Learning project embarks on a comprehensive analysis of the superstore's sales and customer data. Through the application of data analysis techniques, statistical modeling, and machine learning algorithms, we aim to unlock valuable insights that will help the superstore optimize their operations, improve customer experiences, and achieve a competitive advantage in the dynamic retail landscape. Let's dive into the data and uncover the secrets that lie within.

Our Decision Tree algorithm yields the highest accuracy 95.165%. Random Forest algorithm also has the accuracy score of 95.06%. Both these models top the list in accuracy score. Any accuracy score above 70% is considered good, but be careful because if your accuracy is extremely high, it may be too good to be true(an example of Overfitting).Thus,80% is the ideal accuracy!

REFERENCE:

1.Superstore Dataset-Dataset containing sales and Profit of a superstore-Kaggle-Vivek Chowdhury.

Link : https://www.kaggle.com/datasets/vivek468/superstore-dataset-final/code

2. Superstore Dataset Complete Analysis | Plotly-Kaggle-Olekshi Zhukov.

Link: https://www.kaggle.com/code/zhukovoleksiy/superstore-dataset-complete-analysis-plotly