Documentation On

“Supply Chain Management System”

Prepared by-

Patil Darshanesh

Table of Contents

Table of Figures

Abstract

The following documentation presents an in-depth analysis of a Supply Chain Management System, exploring its various components and their interactions. Through data analysis, visualization, and machine learning techniques, this documentation aims to enhance understanding and decision-making within the supply chain domain. The documentation discusses the project's scope, objectives, methodology, and key findings, offering insights into the potential improvements in efficiency, customer satisfaction, and overall operational effectiveness.

Scope

The Supply Chain Management System project encompasses the development and implementation of a comprehensive digital platform that streamlines the end-to-end supply chain processes. This includes order management, inventory tracking, supplier collaboration, demand forecasting, and logistics optimization.

Aims

- To design an efficient and transparent Supply Chain Management System that optimizes the flow of goods and information.
- To enhance operational efficiency by minimizing lead times, reducing stockouts, and improving order fulfilment accuracy.
- To enable data-driven decision-making through advanced data analysis and visualization techniques.
- To explore the application of machine learning models for predicting and mitigating late deliveries.

Objectives

- Analyse the existing supply chain processes to identify opportunities for improvement.
- Implement data preprocessing techniques to ensure data accuracy and consistency for analysis.
- Utilize visualization tools like Tableau to create insightful dashboards for monitoring key supply chain metrics.
- Apply hypothesis testing to validate the effectiveness of process optimizations and improvements.
- Segment customers based on order history and preferences to tailor supply chain strategies.
- Perform ABC analysis to prioritize products based on their impact on overall supply chain performance.
- Build machine learning models that predict the likelihood of late deliveries.

Workflow of the Project

1. **Dataset Overview:**

   A dataset is a collection of data that is organized in a specific way to be used for analysis or other purposes. It can be in the form of a spreadsheet, a database, or other formats. The dataset overview provides a summary of the data, including its size, structure, and variables.

2. **Data Preprocessing:**

   Data preprocessing is the process of cleaning and transforming raw data into a format that can be easily analyzed. It involves tasks such as removing duplicates, handling missing values, and scaling data. The goal of data preprocessing is to improve the quality of the data and make it more suitable for analysis.

3. **Post-Processed Dataset Overview:**

   Post-processed dataset overview provides a summary of the dataset after it has been cleaned and transformed through data preprocessing. It includes information such as the number of records, variables, and data types. This overview is important for understanding the quality of the data and its suitability for analysis

4. **Visualization using Tableau:**

   Tableau is a data visualization tool that allows users to create interactive and dynamic visualizations of their data. It can be used to create charts, graphs, and other visualizations that help to communicate insights and trends in the data. Tableau is a popular tool for data analysis and is used by many organizations to create reports and dashboards

5. **Hypothesis Testing:**

   Hypothesis testing is a statistical method used to determine whether a hypothesis about a population is true or false. It involves formulating a null hypothesis and an alternative hypothesis, collecting data, and using statistical tests to determine whether the null hypothesis can be rejected. Hypothesis testing is an important tool for data analysis and is used in many fields, including science, business, and social sciences.

6. **Customer Segmentation:**

   Customer segmentation is the process of dividing customers into groups based on their characteristics and behavior. It is used to better understand customer needs and preferences, and to develop targeted marketing strategies. Customer segmentation can be done using a variety of methods, including demographic, geographic, and psychographic segmentation.

7. **ABC Analysis:**

   ABC analysis is a method used to categorize items based on their importance. It is commonly used in inventory management to prioritize items based on their value and usage. The method involves dividing items into three categories: A, B, and C, with A items being the most important and C items being the least important. ABC analysis is a useful tool for managing inventory and optimizing resources

8. **Machine Learning Models for Predicting Late Delivery:**

   Machine learning models can be used to predict late delivery in various industries, including logistics and transportation. These models use historical data to identify patterns and make predictions about future events. Some common machine learning algorithms used for predicting late delivery include decision trees, random forests, and neural networks.

9. **Model Evaluation:**

   Results and discussion involve presenting the findings of the data analysis and discussing their implications. This is an important step in the data analysis process, as it allows stakeholders to understand the insights and make informed decisions based on the data. Results and discussion can be presented in various formats, including reports, presentations, and dashboards.

Importing Library and Dataset:-

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from google.colab import drive
drive.mount('/content/drive')

df=pd.read_csv("/content/drive/MyDrive/pj
notebooks/dataset/DataCoSupplyChainDataset.csv", encoding_errors="ignore")
pd.set_option('display.max_columns',None)
```

The given code imports the necessary libraries for data analysis and visualization in Python. The libraries imported are pandas, numpy, matplotlib.pyplot, seaborn, and google.colab. The dataset is imported using the pandas library's read_csv function. The dataset is located in the Google Drive folder and is loaded into the dataframe df. The encoding_errors parameter is set to "ignore" to handle any encoding errors that may occur while reading the CSV file. The pd.set_option() function is used to set the maximum number of columns to be displayed in the output.

Dataset Overview

Entries: 180,519 Columns: 53

Float64: 15 columns

Int64: 14 columns

Object: 24 columns

Memory Usage: 73.0+ MB

Columns Name:-

['Type', 'Days for shipping (real)', 'Days for shipment (scheduled)', 'Benefit per order', 'Sales per customer', 'Delivery Status', 'Late_delivery_risk', 'Category Id', 'Category Name', 'Customer City', 'Customer Country', 'Customer Email', 'Customer Fname', 'Customer Id', 'Customer Lname', 'Customer Password', 'Customer Segment', 'Customer State', 'Customer Street', 'Customer Zipcode', 'Department Id', 'Department Name', 'Latitude', 'Longitude', 'Market', 'Order City', 'Order Country', 'Order Customer Id', 'order date (DateOrders)', 'Order Id', 'Order Item Cardprod Id', 'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id', 'Order Item Product Price', 'Order Item Profit Ratio', 'Order Item Quantity', 'Sales', 'Order Item Total', 'Order Profit Per Order', 'Order Region', 'Order State', 'Order Status', 'Order Zipcode', 'Product Card Id', 'Product Category Id', 'Product Description', 'Product Image', 'Product Name', 'Product Price', 'Product Status', 'shipping date (DateOrders)', 'Shipping Mode']

Data Pre-Processing

Null data

Null data was found in three columns, Order Zipcode (86% null values, i.e. a total of 155679 records), Product description (100% null values, i.e. a total of 180519 records) and customer zipcode & customer Last Name (<0.1% is say a total of 3 records and 8 record respectively), together these empty cells correspond to 3.5% of the total dataset.

Removing Redundancies

- Benefit per order and Order Profit Per Order are the same
- Sales per customer and Order Item Total are the same
- Category Id and Product Category Id are the same
- Customer Id and Order Customer Id are the same
- Order Item Cardprod Id and Product Card Id are the same
- Order Item Product Price and Product Price are the same

At this point, we can see that even the Order Customer Id and Customer Id are the same. It would make sense to make a function that check for all columns wether the values match to 100% to remove redundancies. Otherwiese one could opt to consult the makers of the dataset for further info

```
from itertools import combinations

def check_redundancies(data):

    redundancy_list = []

    for i in list(combinations(data.columns, 2)):
        if all(data[i[0]] == data[i[1]]):
            redundancy_list.append(i)
            print("{} and {} are the same".format(*i))

    return redundancy_list

redundancies = check_redundancies(df)

df.drop(["Benefit per order","Order Item Total","Category Id","Customer
Id","Order Item Cardprod Id", "Order Item Product Price"]
,axis=1,inplace=True)
```

State Abbreviations

States in the United States are commonly represented using two-letter abbreviations.

Example: California → CA, New York → NY, Texas → TX

Converted standardized state abbreviations to State Name for consistency and compatibility.

```
# Dictionary mapping abbreviations to state names
state_mapping = {
    'PR': 'Puerto Rico',
    'CA': 'California',
    'NY': 'New York',
    'OH': 'Ohio',
    'PA': 'Pennsylvania',
    'MI': 'Michigan',
.
.
    'ID': 'Idaho',
    'AR': 'Arkansas',
    'MT': 'Montana',
    'IA': 'Iowa',
    'AL': 'Alabama'
}

# Replace abbreviations with state names
df['Customer State'] = df['Customer State'].replace(state_mapping)


rows_to_delete = df[(df['Customer State'] == '95758') | (df['Customer State']
== '91732')].index

# Drop the identified rows
df = df.drop(rows_to_delete)
```

Grouping Multiple Regions

When dealing with large datasets, Grouped multiple regions of North America into North America.

```
# For object data, we will use 'index.isin()'

order_regions = df['Order Region'].value_counts(ascending = False)
n_america = ['West of USA ', 'US Center ', 'South of  USA ', 'East of USA',
'Canada', 'Caribbean']
n_america_regions = order_regions[order_regions.index.isin(n_america)]

def handle_region(value):
    if(value in n_america_regions):
        return 'North America'
    else:
        return value

df['Order Region'] = df['Order Region'].apply(handle_region)
```

Country Name Translation

Multilingual datasets country names to be translated to English for compatibility.

```
# As some countries in the list is in Spanaish, for futher use it is translated
in English
corrections = {
    'Indonesia': 'Indonesia',
    'India': 'India',
    'Australia': 'Australia',
    'China': 'China',
    'Japn': 'Japan',
    'Corea del Sur': 'South Korea',
.
.
.
.
    'Repblica de Gambia':'The Gambia',
    'Botsuana':"Botswana",
    'Guinea Ecuatorial':'Equatorial Guinea'
}

df['Order Country'] = df['Order Country'].replace(corrections)
```

Group infrequent values into an "Others" category

“Categories Name” having count less than 500.

“Product Name” having count less than 1500.

```
# We will create a function to insert all categories with count < 500 into
'Others'
categories = df['Category Name'].value_counts(ascending = False)
categories_under500 = categories[categories < 500]

def handle_categories(value):
    if(value in categories_under500):
        return 'Others'
    else:
        return value

df['Category Name'] = df['Category Name'].apply(handle_categories)

# We will group all product names < 1500 as 'Others'

products = df['Product Name'].value_counts(ascending = False)
products_under1500 = products[products < 1500]

def handle_products(value):
    if(value in products_under1500):
        return 'Others'
    else:
        return value

df['Product Name'] = df['Product Name'].apply(handle_products)
```

Irrelevant information for the objective of the Study

```
df.drop(["Customer Email","Customer Password","Product Image","Customer
Fname","Customer Lname","Product Description","Order Zipcode","Customer
Zipcode","Customer Country","Order City","Customer Street","Order
State"],axis=1,inplace=True)
```

While these column may be interesting and informative, they do not directly contribute to the objective of the project. It is important to focus on column that are relevant to the project's goal and avoid getting sidetracked by unrelated information.

Minor Pre-processing

```
df.rename(columns={"Type":"Type Of Payments"},inplace=True)

# Sales
df['Sales'] = df['Sales'].round(2)


# Dictionary mapping abbreviations to status names
status_mapping = {
    1: 'Not Available',
    0: 'Available',
}

# Replace abbreviations with status names
df['Product Status'] = df['Product Status'].replace(status_mapping)
```

Save the data in the csv file

```
df.to_csv("/content/drive/MyDrive/pj
notebooks/dataset/Processed_DataCoSupplyChainDataset.csv",index=False)
```

When saving data in a CSV file, it is important to ensure that the file is properly formatted to avoid common issues that may arise. Some of the common problems with CSV files include delimiter issues, special character encoding issues, and invalid column count errors. To save data in a CSV file, you can use the to_csv() function in Python, as shown in the example code provided in the question. When using this function, it is important to specify the correct file path and name, and to set the index parameter to False if you do not want to include the index column in the CSV file. If the data contains special characters or uses a delimiter other than a comma, you may need to change the delimiter setting or change the file type to ensure that the data is properly formatted.

Post-Processed Dataset Overview

Entries: 180,516 Columns: 34

Float64: 9 columns Int64: 10 columns Object: 16 columns Memory Usage: 49.6+ MB

Columns Name:-

['Type Of Payments', 'Days for shipping (real)', 'Days for shipment (scheduled)', 'Sales per customer', 'Delivery Status', 'Late_delivery_risk', 'Category Name', 'Customer City', 'Customer Segment', 'Customer State', 'Department Id', 'Department Name', 'Latitude', 'Longitude', 'Market', 'Order Country', 'Order Customer Id', 'order date (DateOrders)', 'Order Id', 'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id', 'Order Item Profit Ratio', 'Order Item Quantity', 'Sales', 'Order Profit Per Order', 'Order Region', 'Order Status', 'Product Card Id', 'Product Category Id', 'Product Name', 'Product Price', 'Product Status', 'shipping date (DateOrders)', 'Shipping Mode']

Visualization using Tableau

In today's fast-paced world, data is abundant, and making sense of it is crucial. That's where Tableau comes in. Tableau is a powerful data visualization tool that allows you to transform raw data into insightful visualizations, making complex information easily understandable. In this guide, we'll delve into the world of data visualization using Tableau.

1. Introduction to Tableau: Tableau is a data visualization software that enables you to create interactive and shareable dashboards, reports, and charts. It supports various data sources, making it easy to connect to your data, regardless of its format.

2. Getting Started:

- Installation: Start by downloading and installing Tableau Desktop. You can choose from different versions based on your needs.
- Connecting Data: Tableau supports various data sources, including databases, spreadsheets, and cloud platforms. Connect to your data source using the intuitive interface.

3. Creating Visualizations:

- Sheets: A sheet is a blank canvas where you build your visualizations. You can add different elements such as charts, maps, and text.
- Marks and Filters: Customize your visualizations using marks (data points) and filters to focus on specific data subsets.
- Show Me: Tableau's "Show Me" feature suggests the most suitable visualization types based on your data.

4. Building Dashboards:

- Dashboard Workspace: Combine multiple sheets onto a dashboard canvas to create a comprehensive view.
- Layout and Formatting: Arrange sheets, images, and text boxes using Tableau's drag-and-drop interface. Apply formatting for a polished look.

5. Interactivity:

- Actions: Create interactive dashboards by defining actions. For example, clicking on a chart can filter data in other visualizations.
- Parameters: Allow users to control certain aspects of the visualization, such as choosing different measures or dimensions.

6. Sharing and Collaboration:

- Tableau Server/Online: Publish your visualizations to Tableau Server or Tableau Online for easy sharing within your organization.
- Embedding: Embed Tableau visualizations in websites, blogs, or other online platforms for broader reach.

7. Best Practices:

- Simplicity: Keep visualizations clean and concise. Avoid clutter and unnecessary details.
- Color and Design: Use colors meaningfully and consistently. Choose appropriate chart types for the data you want to convey.
- Storytelling: Create a narrative using dashboards to guide users through insights and findings.

8. Advanced Techniques:

- Calculations: Use calculated fields to perform complex computations or create new measures.
- Mapping: Utilize geographic data to create interactive maps with filters and tooltips.

9. Resources and Learning:

- Tableau Community: Join the Tableau user community to learn from others, ask questions, and share your knowledge.
- Online Tutorials: Tableau offers a range of tutorials, webinars, and courses to enhance your skills.

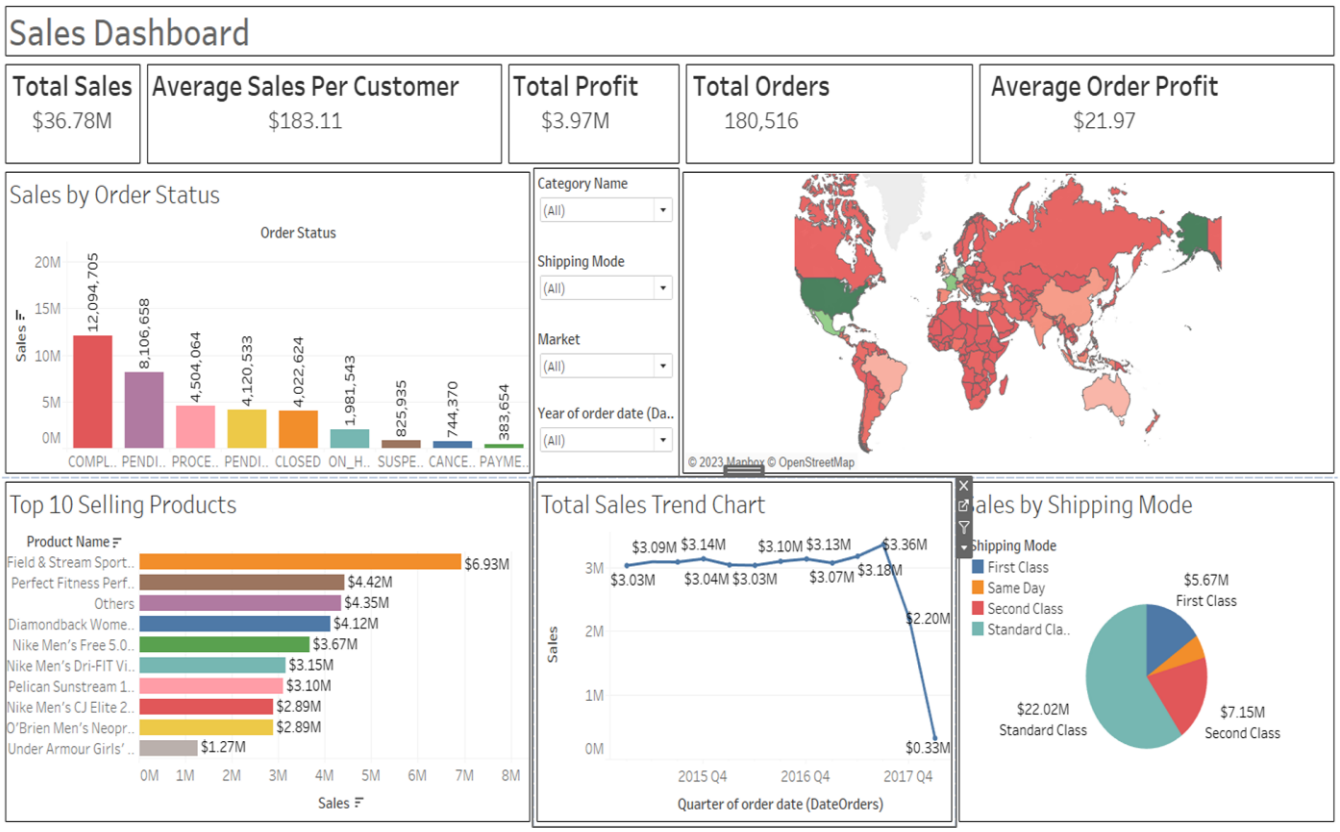

Figure 1 Tableau Sales Dashboard

Tableau Link :-

https://public.tableau.com/app/profile/darshanesh.patil/viz/SalesReport_16925513303360/TotalSalesTrendChart

Correlation Matrix

A correlation matrix is a table that displays the correlation coefficients between variables. It is a statistical technique used to evaluate the relationship between two or more variables in a data set. Each cell in the matrix represents the correlation between two variables, and the matrix is typically "square," with the same variables shown in the rows and columns. The correlation coefficient in each cell indicates the strength and direction of the relationship between the variables. The coefficient ranges from -1 to 1, where -1 represents a perfect negative correlation, 1 represents a perfect positive correlation, and 0 represents no correlation.

A correlation matrix is useful for several purposes, including summarizing data, identifying patterns, and making decisions based on the relationships between variables. It can be used as an input for more advanced analyses and as a diagnostic tool for advanced analyses. Additionally, it is commonly used in regression techniques like simple linear regression, multiple linear regression, and lasso regression models.

When creating a correlation matrix, several key decisions need to be made, including the choice of correlation statistic, coding of the variables, treatment of missing data, and presentation. The matrix is often used in conjunction with other types of statistical analysis. To find important parameters, data correlation is performed.

```
fig, ax = plt.subplots(figsize=(22,13))         # figsize
# Here, the plt.subplots() function is used to create a figure and a set of
# subplots.
# The fig, ax are the variables that will hold the figure and axes objects,
# respectively.
# The figsize parameter sets the width and height of the figure in inches.
# In this case, the figure size is set to 22 inches in width and 13 inches in
# height.

sns.heatmap(df.corr(),annot=True,linewidths=.5,fmt='.1g',cmap= 'coolwarm')
# Heatmap for correlation matrix
# This line generates the heatmap using the sns.heatmap() function.

# df.corr(): This part of the code calculates the correlation matrix of the
# DataFrame df. The corr() function computes the pairwise correlation of
# columns in the DataFrame.
# annot=True: This parameter specifies that the correlation values should be
# annotated on the heatmap cells.
# linewidths=.5: This parameter sets the width of the lines that separate the
# cells of the heatmap.
# fmt='.1g': This parameter controls the formatting of the annotation values.
# In this case, it's set to display the values with one decimal place in
# general format.
# cmap='coolwarm': This parameter sets the color map for the heatmap.
# 'coolwarm' is a diverging color map that ranges from cool colors (e.g., blue)
# to warm colors (e.g., red) to represent positive and negative correlations.
```

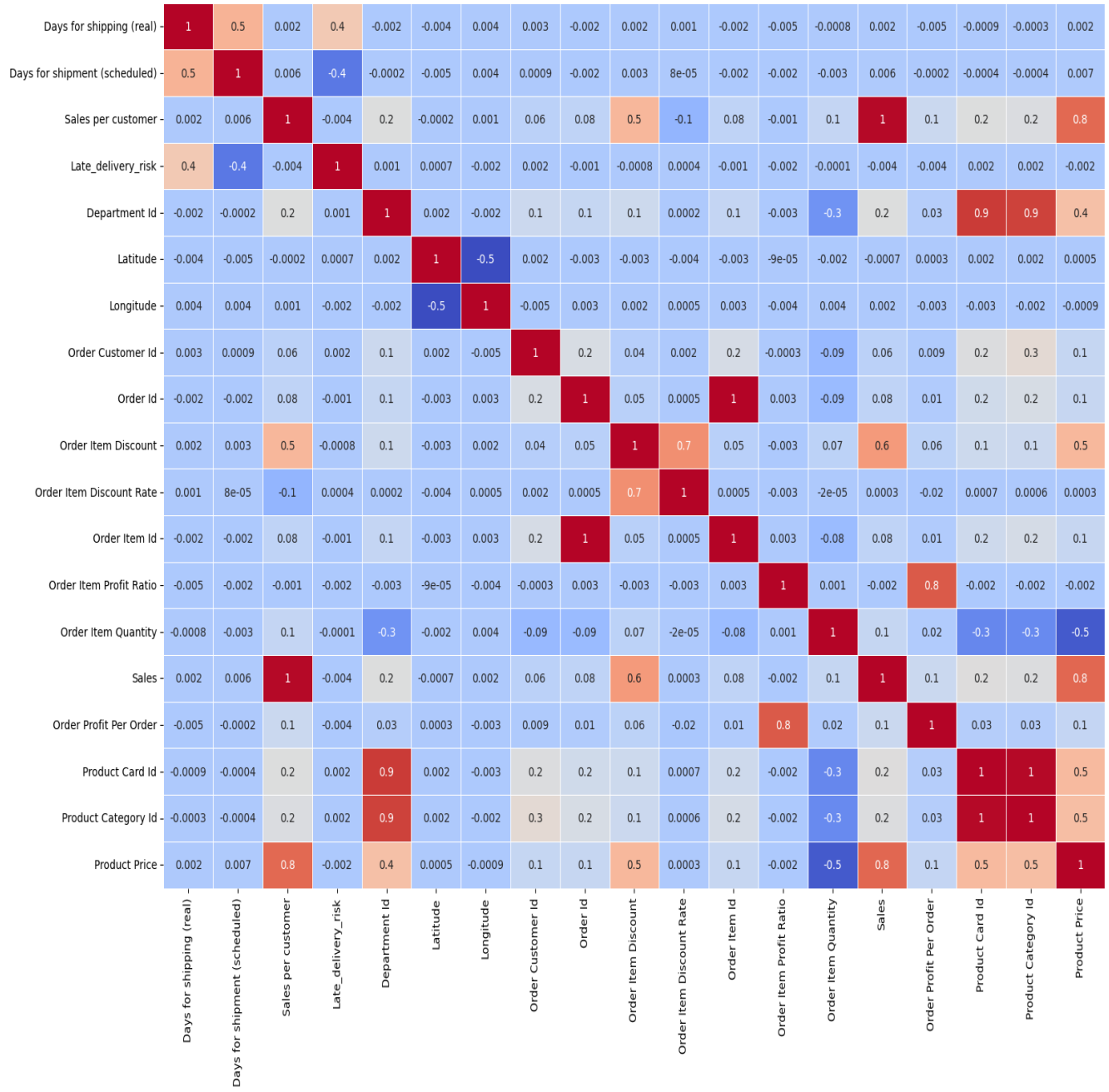

Figure 2 Corelation Matrix

We can observe that product price price has high correlation with Sales,Order Item Total.

Hypothesis Testing

Introduction to Hypothesis Testing

Hypothesis testing is a powerful statistical technique used to make informed decisions based on data. It allows us to evaluate whether there is enough evidence to support or reject a claim about a population parameter, such as a mean or proportion. This process involves formulating two competing hypotheses, the null hypothesis (H_0) and the alternative hypothesis (H_a), and analyzing sample data to draw conclusions about the population.

Steps in Hypothesis Testing:

Step-1 Forming Statements of Null and Alternative Hypotheses

Introduction: Forming the statements of null and alternative hypotheses is a crucial step in hypothesis testing. These statements define the research question and guide the statistical analysis. In this guide, we will learn how to construct the null and alternative hypotheses correctly and provide examples for better understanding.

1. Null Hypothesis (H_0): The null hypothesis represents the *default* assumption or the status quo. In testing hypothesis, the researcher initially assumes *Null Hypothesis to be true*. It states that there is no significant effect, difference, or relationship between groups or variables. It is denoted as H_0 and often includes words like "no," "not," or "equal to.". For numerical comparisons, it uses an equality sign (e.g., =, ≤, or ≥).

Example: If we are testing whether a new drug has an effect on curing a disease, the null hypothesis would be:

H_0: The new drug has no effect on curing the disease.

2. Alternative Hypothesis (H_a or H_1): The alternative hypothesis (Ha) represents the claim you want to test, and it contradicts the null hypothesis. It can be directional (one-tailed) or non-directional (two-tailed).

Example: Using the same drug example, the alternative hypothesis would be:

H_a: The new drug improves the cure rate of the disease.

Rules for Formulating Ha:

For a one-tailed alternative, use > or < in the statement, indicating the direction of the effect.

Example: If the research question suggests that the new drug is expected to improve the cure rate, the alternative hypothesis would be one-tailed:

H_a: The new drug's cure rate is greater than that of the existing drug.

For a two-tailed alternative, use ≠ in the statement, indicating there is a difference but not specifying the direction.

Example: Continuing with the drug example, if the research question is whether the new drug is simply different from the existing drug, the alternative hypothesis would be two-tailed:

H_a: The new drug's cure rate is different from that of the existing drug.

Step 2: Set the Significance Level (α)

Setting the significance level (α) is a critical step in hypothesis testing. It represents the probability of rejecting the null hypothesis when it is actually true. Choosing an appropriate significance level is essential to control the balance between Type I and Type II errors in hypothesis testing.

Type I Error (False Positive): Occurs when we reject the null hypothesis when it is true. It means we mistakenly conclude that there is a significant effect or difference when there isn't.

Type II Error (False Negative): Occurs when we fail to reject the null hypothesis when it is false. It means we miss detecting a significant effect or difference when one actually exists.

		The Truth (Based on Entire Population)	
		Nothing Is There (H_0 Is True)	Something Is There (H_0 Is False)
Your Conclusion (Based on Your Sample)	I Don't See Anything (Nonsignificant)	Right!	Wrong (Type II Error)
	I See Something (Significant)	Wrong (Type I Error)	Right!

Figure 3 Confusion Matrix for Signiificant level

The most common significance levels used are 0.05 (5%) and 0.01 (1%), but researchers can choose other levels based on the context and the consequences of making errors.

Guidelines for Choosing the Significance Level (α):

1. Consider the Field of Research: Different fields may have different conventions for significance levels. In some disciplines, a significance level of 0.05 is commonly used, while in others, a more stringent level like 0.01 is preferred.
2. Impact of Errors: Evaluate the consequences of Type I and Type II errors in your specific scenario. If making a Type I error is more costly or harmful, consider using a smaller significance level (e.g., 0.01). On the other hand, if Type II errors are more concerning, you might opt for a larger significance level (e.g., 0.10).
3. Sample Size: Larger sample sizes can provide more accurate estimates of population parameters, leading to greater confidence in the results. With larger samples, researchers may choose a smaller significance level to reduce the chances of Type I errors.
4. Prior Research: If there is substantial prior evidence supporting a particular hypothesis, a higher significance level may be acceptable.
5. Practical Significance: Consider whether a statistically significant result is practically meaningful in your context. A smaller effect size might require a smaller significance level to detect it.

6. Bias and Data Quality: If there is a possibility of bias or low data quality, a more conservative significance level may be warranted.

Points to Remember:

- *Only* the researcher can decide the appropriate significance level (α) based on the factors mentioned above.
- *Always* report the chosen significance level in research papers or reports to ensure transparency.
- For a given level of significance, if the sample size is increased, the power of the test will increase as well. Conversely, if the sample size is decreased, the power of the test will decrease.

Step 3: Collect Data and Calculate Test Statistic

Collecting appropriate data and calculating the test statistic are crucial steps in hypothesis testing. The choice of the test statistic depends on the type of data and the hypothesis being tested. Let's elaborate on the terms used in the formulas for different test statistics:

1. $\bar{X}$ (Sample Mean): The sample mean represents the average value of a set of observations in a sample. It is calculated by summing up all the values in the sample and then dividing by the number of observations (sample size, n). The sample mean is denoted by the symbol $\bar{X}$.
2. s (Sample Standard Deviation): The sample standard deviation measures the spread or dispersion of data points around the sample mean. It is a measure of how much individual data points deviate from the sample mean. The sample standard deviation is denoted by the symbol s.
3. μ (Population Mean): The population mean represents the average value of the entire population. In hypothesis testing, we often use the population mean as the value we want to test against the sample mean. The population mean is denoted by the Greek letter μ (mu).
4. n (Sample Size): The sample size refers to the number of observations in the sample. It is an essential parameter as it determines the precision of the estimates based on the sample data.
5. Z-Score (Z): The Z-score is used in the one-sample Z-test and represents the number of standard deviations a data point is from the population mean. It measures how far a sample mean deviates from the population mean in terms of standard deviations. In the formula, it is denoted by the symbol Z.
6. t-Score (t): The t-score is used in t-tests (both one-sample and two-sample) and is similar to the Z-score. However, t-tests are used when the population standard deviation is unknown or the sample size is small. The t-score measures how far a sample mean deviates from the population mean in terms of standard errors. In the formula, it is denoted by the symbol t.

7. σ (Population Standard Deviation): The population standard deviation represents the spread of data points around the population mean for the entire population. In many real-world cases, the population standard deviation is unknown, and we rely on the sample standard deviation (s) instead.
8. √n (Square Root of Sample Size): The square root of the sample size (√n) is often used in the denominators of the test statistic formulas. It helps in scaling the standard deviation to the appropriate level based on the number of observations in the sample.

Let's explore some common scenarios and their corresponding test statistics.

1. One-Sample Z-Test:

- Used when we have a single sample and know the population standard deviation (σ).
- Test Statistic: The Z-score is given by:

$$Z = \frac{\bar{X} - \mu}{\frac{\sigma}{\sqrt{n}}}$$

2. One-Sample t-Test:

- Used when we have a single sample, but the population standard deviation (σ) is unknown.
- Test Statistic: The t-score is given by:

$$t = \frac{\bar{X} - \mu}{\frac{s}{\sqrt{n}}}$$

3. Two-Sample Z-Test:

- Used when comparing two independent samples, and the population standard deviations (σ1 and σ2) are known and equal.
- Test Statistic: The Z-score for the difference between means is given by:

$$Z = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

4. Two-Sample t-Test:

- Used when comparing two independent samples, and the population standard deviations (σ1 and σ2) are unknown or different.
- Test Statistic: The t-score for the difference between means is given by:

$$t = \frac{(\bar{X}_1 - \bar{X}_2) - (\mu_1 - \mu_2)}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

5. Chi-Square Test:

- Used to compare observed frequencies with expected frequencies in categorical data.

- Test Statistic: The chi-square statistic is calculated based on the difference between observed and expected frequencies.

6. F statistic test:

The F-statistic is a fundamental concept in statistics, commonly used in analysis of variance (ANOVA) and regression analysis. It's a crucial tool for determining whether the variations between groups or models are statistically significant or if they could have occurred by chance.

$$F = \frac{\text{Variance Between Groups/Models}}{\text{Variance Within Groups/Models}}$$

Where:

- Variance Between Groups/Models: This measures the differences between the means of the groups or models being compared.
- Variance Within Groups/Models: This quantifies the variability of data points within each group or model.

Step 4: Calculate the P-value and Interpret the Result:

In hypothesis testing, the p-value is a crucial measure that quantifies the strength of evidence against the null hypothesis. It represents the probability of obtaining the observed results or more extreme results under the assumption that the null hypothesis is true. The smaller the p-value, the stronger the evidence against the null hypothesis.

Calculating the P-value:

The method to calculate the p-value varies depending on the statistical test being used and the nature of the hypothesis being tested.

Interpreting the P-value:

The p-value is compared to the chosen significance level (α) to make a decision in hypothesis testing:

- If the p-value is less than or equal to α ($p \leq \alpha$), then we reject the null hypothesis (H_0).
- If the p-value is greater than α ($p > \alpha$), then we fail to reject the null hypothesis (H_0).

Always Remember:

- A *small p-value* (typically less than the chosen significance level α) suggests strong evidence against the null hypothesis, leading to its rejection.
- A *large p-value* indicates weak evidence against the null hypothesis, leading to the failure to reject it.

Be Cautious of Misinterpretation:

- The p-value does not provide the *probability that the null hypothesis is true or false*. It only quantifies the strength of evidence against the null hypothesis based on the observed data.

- A p-value close to 1 does not mean that the null hypothesis is *proven true*. It simply suggests that there is not enough evidence to reject it.

Library used:-

```
from scipy import stats # for hypothesis testing
```

Hypothesis 1: Impact of delivery time on order profitability

Null hypothesis: There is no significant difference in order profitability between orders with shorter actual delivery time and those with longer delivery time.

Alternative hypothesis: Orders with a shorter actual delivery time are more profitable than those with a longer delivery time.

```
# calculating the delivery time difference
df["Delivery_Time_Difference"] = df["Days for shipping (real)"] - df["Days for
shipment (scheduled)"]

# split the data into two groups based on the delivery time difference
short_delivery = df[df["Delivery_Time_Difference"] < 0]
long_delivery = df[df["Delivery_Time_Difference"] >= 0]

# Conducting a two-sample t-test to compare the mean order profit per order
between the two groups
t_stat, p_value = stats.ttest_ind(short_delivery["Order Profit Per Order"],
long_delivery["Order Profit Per Order"], equal_var=False)

alpha = 0.05

print("Hypothesis 1: Impact of delivery time on order profitability")
print("t-statistic:", t_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Hypothesis 2: Relationship between sales per customer and delivery status
t-statistic: 1.8831012266781173
p-value: 0.05968849992613582
Fail to reject the null hypothesis
```

In the context of hypothesis, it means that there is insufficient evidence to conclude that customers who receive their orders on time have higher sales per customer than those who do not.

Hypothesis 2: Relationship between sales per customer and delivery status

Null hypothesis: There is no significant difference in sales per customer between orders delivered on time and those that are not.

Alternative hypothesis: Customers who receive their orders on time have higher sales per customer than those who do not.

```
# create two groups based on the delivery status
on_time1 = df[df["Delivery Status"] == "Shipping on time"]
on_time2 = df[df["Delivery Status"] == "Advance shipping"]
on_time=pd.concat([on_time1, on_time2], axis=0)
not_on_time = df[df["Delivery Status"] == "Late delivery"]

# Conducting a two-sample t-test to compare the mean sales per customer between the two groups
t_stat, p_value = stats.ttest_ind(on_time["Sales per customer"], not_on_time["Sales per customer"], equal_var=False)

print("Hypothesis 2: Relationship between sales per customer and delivery status")
print("t-statistic:", t_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Hypothesis 2: Relationship between sales per customer and delivery status
t-statistic: 1.8831012266781173
p-value: 0.05968849992613582
Fail to reject the null hypothesis
```

Based on the output, the t-statistic is 1.88 and the p-value is 0.0598, which is higher than the significance level of 0.05.

Therefore, we fail to reject the null hypothesis that there is no significant difference in sales per customer between orders delivered on time and those that are not.

This means that there is insufficient evidence to conclude that customers who receive their orders on time have higher sales per customer than those who do not. However, since the p-value is relatively close to the significance level, further investigation may be warranted to determine if there is a potential relationship between delivery status and sales per customer.

Hypothesis 3: Impact of late delivery risk on order profit

Null hypothesis: There is no significant difference in order profitability between orders with high and low late delivery risk.

Alternative hypothesis: Orders with a higher late delivery risk are less profitable than those with a lower late delivery risk.

```
# create two groups based on the late delivery risk
high_risk = df[df["Late_delivery_risk"] == 1]
low_risk = df[df["Late_delivery_risk"] == 0]

# Conducting a two-sample t-test to compare the mean order profit per order
between the two groups
t_stat, p_value = stats.ttest_ind(high_risk["Order Profit Per Order"],
low_risk["Order Profit Per Order"], equal_var=False)

print("Hypothesis 3: Impact of late delivery risk on order profit")
print("t-statistic:", t_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Hypothesis 3: Impact of late delivery risk on order profit
t-statistic: -1.5871340032452448
p-value: 0.11248410145210566
Fail to reject the null hypothesis
```

Based on the given output, the calculated t-statistic is -1.59 and the p-value is 0.11. Since the p-value is greater than the significance level of 0.05, we fail to reject the null hypothesis. Therefore, there is no significant difference in order profitability between orders with high and low late delivery risk.

Hypothesis 4: Relationship between product price and sales:

Null Hypothesis (H0): There is no significant relationship between product price and sales.

Alternative Hypothesis (H1): There is a significant relationship between product price and sales.

```
# Filter the relevant columns
df1 = df[['Product Price', 'Sales']]

# Calculate the correlation coefficient and p-value
corr_coef, p_value = stats.pearsonr(df1['Product Price'], df1['Sales'])

alpha = 0.05

print("Hypothesis: Relationship between product price and sales")
print("Correlation coefficient:", corr_coef)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Hypothesis: Relationship between product price and sales
Correlation coefficient: 0.7899460191767189
p-value: 0.0
Reject the null hypothesis
```

The p-value of 0.0 indicates that there is a very low probability of observing such a strong correlation by chance, which leads to the rejection of the null hypothesis. Therefore, we can conclude that there is a significant relationship between product price and sales.

Hypothesis 5: Impact of shipping mode on order profitability

Null hypothesis: There is no significant difference in order profitability between different shipping modes.

Alternative hypothesis: Certain shipping modes are more profitable than others.

```
# Filter relevant columns for analysis
df_delivery = df[['Shipping Mode', 'Order Profit Per Order']]

#Group the data by shipping mode
shipping_groups = df_delivery.groupby('Shipping Mode')

#Conducting a one-way ANOVA test to compare the mean order profit per order
between the shipping mode groups
f_stat, p_value = stats.f_oneway(*[group['Order Profit Per Order'] for name,
group in shipping_groups])

alpha = 0.05

print("Hypothesis 5: Impact of shipping mode on order profitability")
print("F-statistic:", f_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Hypothesis 5: Impact of shipping mode on order profitability
F-statistic: 1.9760400987761573
p-value: 0.11516423806216304
Fail to reject the null hypothesis
```

Since the p-value is greater than the significance level of 0.05, we fail to reject the null hypothesis. Therefore, there is no significant difference in order profitability between different shipping modes.

Hypothesis 6: Relationship between product category and order profitability

Null hypothesis: There is no significant difference in order profitability between different product categories.

Alternative hypothesis: Certain product categories are more profitable than others.

```
#Filter the relevant columns
df2 = df[['Product Category Id', 'Order Profit Per Order']]

#Group the data by product category
product_groups = df2.groupby("Product Category Id")

#Conducting a one-way ANOVA test to compare the mean order profit per order between the product groups
f_stat, p_value = stats.f_oneway(*[group['Order Profit Per Order'] for name, group in product_groups])

alpha = 0.05

print("Hypothesis 6: Relationship between product category and order profitability")
print("F-statistic:", f_stat)
print("p-value:", p_value)

if p_value < alpha:
    print("Reject the null hypothesis")
else:
    print("Fail to reject the null hypothesis")
```

```
Hypothesis 6: Relationship between product category and order profitability
F-statistic: 50.74877365263413
p-value: 0.0
Reject the null hypothesis
```

Since the p-value is less than the significance level of 0.05, we reject the null hypothesis and conclude that there is a significant difference in order profitability between different product categories. The alternative hypothesis that certain product categories are more profitable than others is supported by the data.

The output shows that the calculated t-statistic is 1.14 and the corresponding p-value is 0.25.Since the p-value is higher than the significance level of 0.05, we cannot reject the null hypothesis. This means that there is no significant difference in order profitability between orders with shorter actual delivery time and those with longer delivery time.

Therefore, based on the test results, we cannot say that delivery time has a significant impact on order profitability.

Customer Segmentation

Customer segmentation is the practice of dividing a company's customer base into distinct groups based on certain shared characteristics and behaviours.

Purpose: It helps businesses better understand their customers, tailor marketing strategies, and provide personalized experiences.

Types of Segmentation:

Demographic: Based on factors like age, gender, income, education, etc.

Geographic: Segmentation by location, such as country, region, city, etc.

Psychographic: Focuses on lifestyle, values, interests, and personality traits.

Behavioural: Grouping based on purchasing habits, usage patterns, etc.

Benefits:

Improved Marketing

Enhanced Customer Experience:

Efficient Resource Allocation:

Better Product Development:

Understanding customer needs and targeting specific clusters of customers based on their need is one way for a supply chain company to increase number of customers and also to gain more profits. Since ,purchase history of customers is already available in the dataset, it can use RFM analysis for customer segmention. Even though there are so many different methods for customer segmentation,RFM analysis is being used because it utilizes numerical values to show Customer recency,frequency and monetary values and also the output results are easy to interpret.

```
data=df
# Sales per customer and Order Item Total are the same
# Calculating total price for which each order
data['TotalPrice'] = data['Order Item Quantity'] * data['Sales per customer']#
Multiplying item price * Order quantity

data['order date (DateOrders)'].max() # Calculating when the last order come
to check recency
```

The last order in the dataset was made on 2017-10-1. So the present time is set slightly above than the last order time for more accuracy of recency value.

```
#Present date was set to next day of the last order. i.e,2018-02-01
import datetime as dt
from datetime import datetime
present = dt.datetime(2017,10,1)
data['order date (DateOrders)'] = pd.to_datetime(data['order date
(DateOrders)'])

# Grouping all values into new data frame named customer segmentation
Customer_seg = data.groupby('Order Customer Id').agg({'order date
(DateOrders)': lambda x: (present - x.max()).days, 'Order Id': lambda x:
len(x), 'TotalPrice': lambda x: x.sum()})
#Changing order dates to int format
Customer_seg['order date (DateOrders)'] = Customer_seg['order date
(DateOrders)'].astype(int)
# Renaming columns as R_Value,F_Value,M_Value
Customer_seg.rename(columns={'order date (DateOrders)': 'R_Value',
                        'Order Id': 'F_Value',
                        'TotalPrice': 'M_Value'}, inplace=True)
Customer_seg.head()
```

R_Value(Recency) indicates how much time elapsed since a customer last order.

F_Value(Frequency) indicates how many times a customer ordered.

M_Value(Monetary value) tells us how much a customer has spent purchasing items.

```
quantiles = Customer_seg.quantile(q=[0.25,0.5,0.75]) #Dividing RFM data into
four quartiles
quantiles = quantiles.to_dict()
```

The total data is divided into 4 quantiles. The R_Value should be low because it indicates recent customer activity and F_value, M_Value should be high since they indicate frequency and total value of purchase. Function is defined to indicate quantiles as numerical values.

```
# R_Score should be minimum so 1st quantile is set as 1.
def R_Score(a,b,c):
    if a <= c[b][0.25]:
        return 1
    elif a <= c[b][0.50]:
        return 2
    elif a <= c[b][0.75]:
        return 3
    else:
        return 4
# The higher the F_Score,M_Score the better so 1st quantile is set as 4.
def FM_Score(x,y,z):
    if x <= z[y][0.25]:
        return 4
    elif x <= z[y][0.50]:
```

```
        return 3
    elif x <= z[y][0.75]:
        return 2
    else:
        return 1

# New column for R_Score to indicate numerical score between 1 to 4.
Customer_seg['R_Score'] = Customer_seg['R_Value'].apply(R_Score,
args=('R_Value',quantiles))
# New column for F_Score to indicate numerical score between 1 to 4.
Customer_seg['F_Score'] = Customer_seg['F_Value'].apply(FM_Score,
args=('F_Value',quantiles))
# New column for M_Score to indicate numerical score between 1 to 4.
Customer_seg['M_Score'] = Customer_seg['M_Value'].apply(FM_Score,
args=('M_Value',quantiles))
Customer_seg.head()
```

The individual scores of R,F,M are known.A column for combined RFM score is created.

```
#Adding R,F,M Scores to one new column
Customer_seg['RFM_Score'] = Customer_seg.R_Score.astype(str)+
Customer_seg.F_Score.astype(str) + Customer_seg.M_Score.astype(str)
Customer_seg.head()
```

How many different customer segments are there in total can be found using .unique() and len method.

```
count=Customer_seg['RFM_Score'].unique()
print(count)# Printing all Unique values
len(count)# Total count
```

It can be seen that there are 33 different customer segments. To make it easier for segmentation individual R,F,M scores are added together

```
# Calculate RFM_Score
Customer_seg['RFM_Total_Score'] =
Customer_seg[['R_Score','F_Score','M_Score']].sum(axis=1)
Customer_seg['RFM_Total_Score'].unique()
```

There are 9 values in total for customer segmentation.Appropriate names were assigned for each value seperately.

```
# Define rfm_level function
def RFM_Total_Score(df):

    if (df['RFM_Total_Score'] >= 11):# For RFM score with values 11,12
        return 'Champions'
    elif (df['RFM_Total_Score'] == 10):# For RFM score with value 10
        return 'Loyal Customers'
    elif (df['RFM_Total_Score'] == 9): # For RFM score with value 9
        return 'Recent Customers'
    elif (df['RFM_Total_Score'] == 8): # For RFM score with value 8
         return 'Promising'
    elif (df['RFM_Total_Score'] == 7): # For RFM score with value 7
        return 'Customers Needing Attention'
    elif (df['RFM_Total_Score'] == 6): # For RFM score with value 6
        return 'Cant lose them'
    elif (df['RFM_Total_Score'] == 5): # For RFM score with value 5
        return 'At Risk'
    else:                              # For RFM score with value less than
5

        return 'Lost'
# Create a new variable RFM_Level
Customer_seg['Customer_Segmentation'] =Customer_seg.apply(RFM_Total_Score,
axis=1)
# Print the header with top 5 rows to the console
Customer_seg.head()
```

Order Customer Id	R_Value	F_Value	M_Value	R_Score	F_Score	M_Score	RFM_Score	RFM_Total_Score	Customer_Segmentation
1	669	1	2362.250061	4	4	3	443	11	Champions
2	13	10	2842.700073	2	2	2	222	6	Cant lose them
3	106	18	6143.760057	3	1	1	311	5	At Risk
4	257	14	4370.629991	4	2	2	422	8	Promising
5	334	7	2993.790032	4	3	2	432	9	Recent Customers

Figure 4 Customer Segment with top 5 rows

How many customers are present in each segment?

```
# Calculate average values for each RFM_Level, and return a size of each
segment
Customer_seg['Customer_Segmentation'].value_counts().plot.pie(figsize=(10,10)
,startangle=135, explode=(0,0,0,0.1,0,0,0,0),autopct='%.1f',shadow=True)
plt.title("Customer Segmentation",size=15)
plt.ylabel(" ")
plt.axis('equal')
plt.show()
```

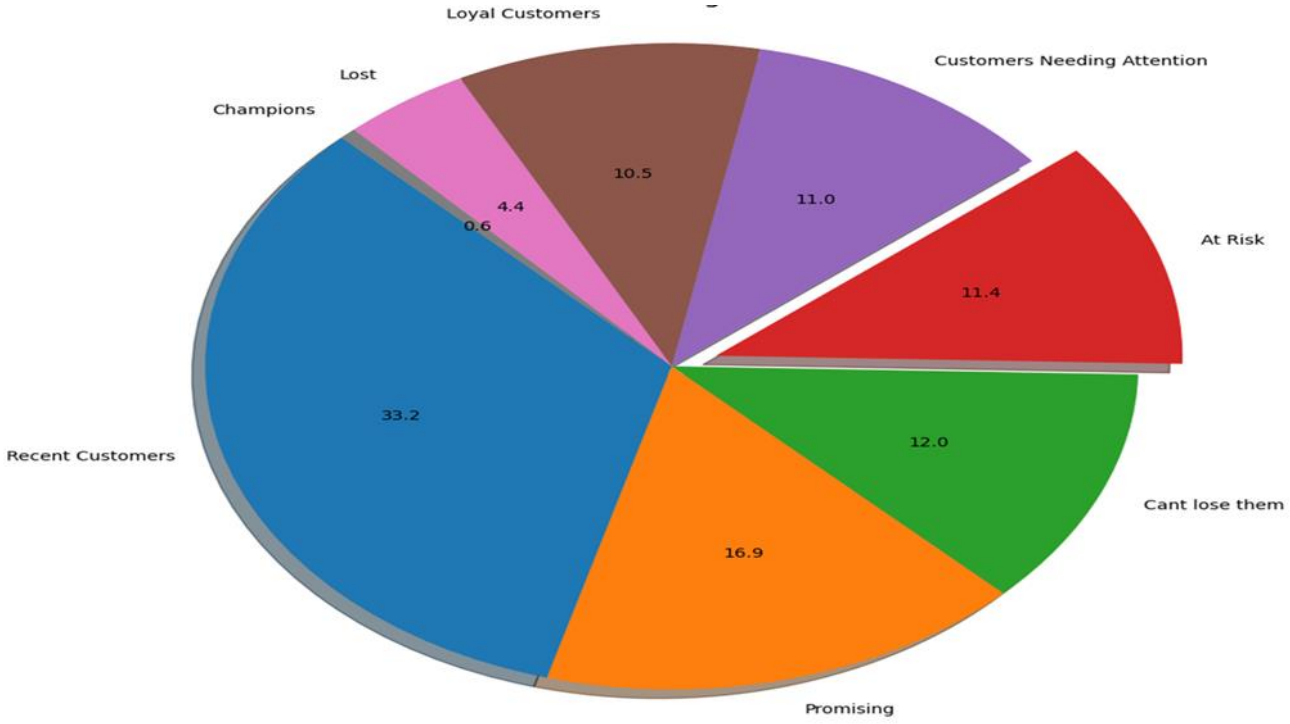

Figure 5 Customers percentage in each segment

Since total customers are divided into 9 segments it can be seen that, 11.4% customers are at risk of losing them as customers and 11% customers needs attention else even they will be lost eventually.It can be seen that 4.4% of customers are already lost.

Our Top 10 Churned best customers who has not purchased anything in a while

```
churned=Customer_seg[(Customer_seg['RFM_Score']=='411')].sort_values('M_Value
', ascending=False).head(10)
churned
```

Order Customer Id	R_Value	F_Value	M_Value	R_Score	F_Score	M_Score	RFM_Score	RFM_Total_Score	Customer_Segmentation
11065	186	41	18641.300091	4	1	1	411	6	Cant lose them
6285	209	37	18287.010097	4	1	1	411	6	Cant lose them
7892	269	29	17620.470196	4	1	1	411	6	Cant lose them
2893	189	24	17536.609842	4	1	1	411	6	Cant lose them
4181	302	29	17333.960094	4	1	1	411	6	Cant lose them
4781	379	31	17048.380088	4	1	1	411	6	Cant lose them
9271	221	35	17044.910217	4	1	1	411	6	Cant lose them
4659	294	27	16973.060024	4	1	1	411	6	Cant lose them
1695	203	33	16916.020176	4	1	1	411	6	Cant lose them
1492	232	38	16617.380169	4	1	1	411	6	Cant lose them

Figure 6 Top 10 Churned best customers who has not purchased anything in a while

These customers used to place orders with huge amounts very frequently but they did not place orders from almost a year which means they are purchasing from other companies. These groups of people should be targeted with offers to gain them back.

Top 10 new best customers who place costly orders often.

```
#The R_Score should be low and F_Score, M_Score should be as high as possible
Customer_seg[(Customer_seg['RFM_Score']=='144')|(Customer_seg['RFM_Score']=='
143')].sort_values('M_Value', ascending=False).head(10)
```

Order Customer Id	R_Value	F_Value	M_Value	R_Score	F_Score	M_Score	RFM_Score	RFM_Total_Score	Customer_Segmentation
18101	-85	1	1500.0	1	4	3	143	8	Promising
18083	-84	1	1500.0	1	4	3	143	8	Promising
18047	-84	1	1500.0	1	4	3	143	8	Promising
18065	-84	1	1500.0	1	4	3	143	8	Promising
18119	-85	1	1500.0	1	4	3	143	8	Promising
18046	-84	1	1485.0	1	4	3	143	8	Promising
18100	-85	1	1485.0	1	4	3	143	8	Promising
18118	-85	1	1485.0	1	4	3	143	8	Promising
18064	-84	1	1485.0	1	4	3	143	8	Promising
18082	-84	1	1485.0	1	4	3	143	8	Promising

Figure 7 Top 10 new best customers who place costly orders often

The above customers has the potential to become best customers this people should be targeted to convert them into loyal customers.All these different segment of customers should be targeted with different tailored advertisments and rewards for increased profits and more responsiveness from customers.

ABC Analysis

ABC analysis, also known as Pareto analysis or the 80/20 rule, is a technique often used in inventory management and supply chain optimization. It categorizes items into three groups based on their value or importance. The pareto rule which states that 80% of outcome comes from 20% of causes. Similarly, here, 80% of the revenue is generated by 20% of all products. High care should be taken that these 20% of the products are always in-stock.

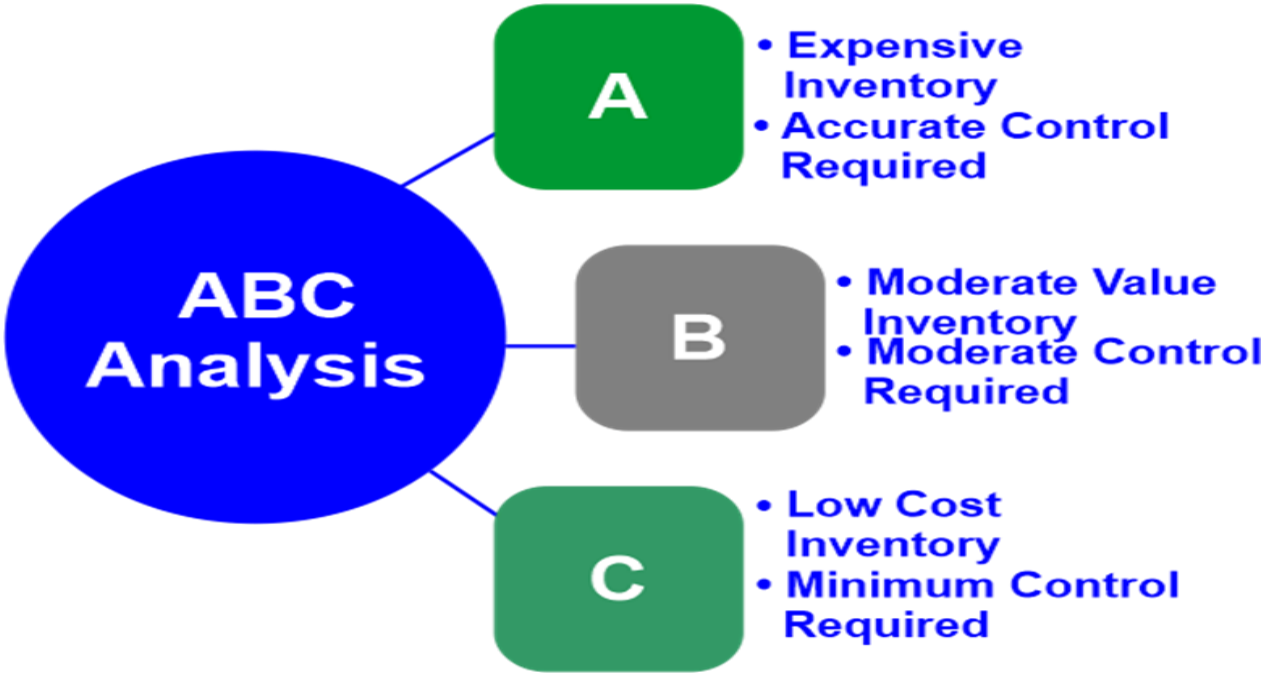

Figure 8 ABC Analysis

Here's how we segment product based on their revenue generating capacity.

- We find percentage of total revenue generated by each product.

- Then we arrange them in decreasing order of percentage and we find cumulative percentage of revenue for each of the products.

Product Class	Revenue Percentage Range	Description
Class A	0% - 75%	Highest revenue-generating products
Class B	75.01% - 95%	Next significant revenue-generating products
Class C	95.01% - 100%	Long tail products contributing less revenue

```
Total_Products=df["Product Name"].nunique()
print("Total Number of products: "+f"{Total_Products}")

from plotly import graph_objects as go
import plotly.express as px
Revenue_ABC=df.groupby(["Department Name","Product
Name"]).agg(Total_Revenue=("Sales per
customer","sum")).sort_values(by="Total_Revenue",ascending=False).reset_index
()
Revenue_ABC["cum_sum"]=Revenue_ABC["Total_Revenue"].cumsum()
Revenue_ABC["cum_per"]=Revenue_ABC["cum_sum"]/Revenue_ABC["Total_Revenue"].su
m()*100
Revenue_ABC["per"]=Revenue_ABC["cum_per"]-Revenue_ABC["cum_per"].shift(1)
Revenue_ABC.loc[0,"per"]=Revenue_ABC["cum_per"][0]
```

```
def ABC(data):
    if data["cum_per"]<=75:
        return "A"
    elif data["cum_per"]>75 and data["cum_per"]<=95:
        return "B"
    elif data["cum_per"]>95:
        return "C"

Revenue_ABC["ABC_Revenue"]=Revenue_ABC.apply(ABC,axis=1)

Bar_graph_Abc=Revenue_ABC[["ABC_Revenue","Product
Name","Total_Revenue"]].groupby("ABC_Revenue").agg(Revenue=("Total_Revenue","
sum"),count=("Product Name","count"))
```

```
Bar_graph_Abc
fig2=go.Figure(go.Bar(x=Bar_graph_Abc.index,
                      y=Bar_graph_Abc["Revenue"],
                     hovertemplate ="%{label}<br>Revenue:%{value}",
                     texttemplate = "Revenue<br>%{value}",
                     marker_color=["green","yellow","red"],
                     showlegend=False))
fig2.add_trace(
    go.Scatter(
    x=Bar_graph_Abc.index,
    y=Bar_graph_Abc["count"],
    name="Number Of Products",
    mode='lines',
    line = dict(color='blue', width=3),
    yaxis="y2",
    marker_line_width = 0
```

```
    ))

fig2.update_layout(
                    title="Revenue Generated By Products in Different ABC
Segments",
                    xaxis=dict(title="Segment" ),
                    yaxis=dict(showgrid=False),
                    yaxis2=dict(title="Number Of Products", anchor="x",
overlaying="y",side="right",dtick=10),
                    legend = dict(x = 1.05, y = 1))
fig2.show()
```

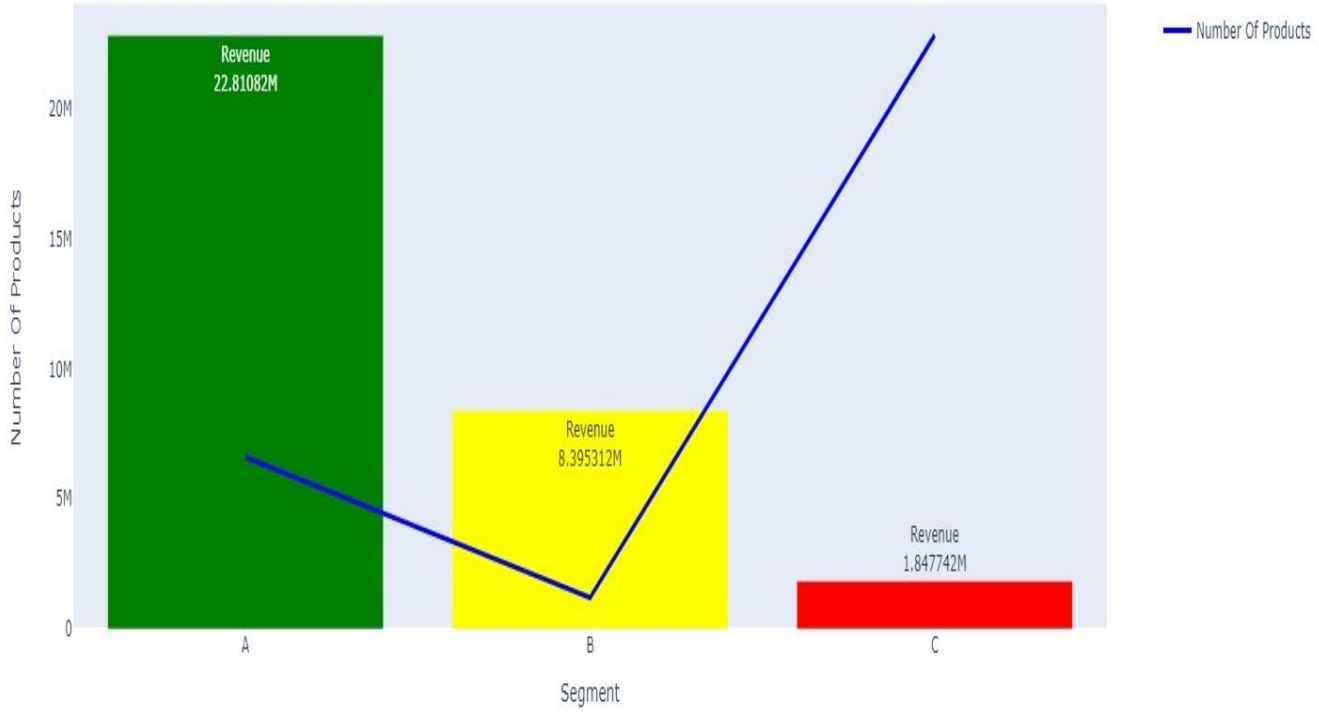

Figure 9 Revenue Generated By Products in Different ABC Segment

Machine learning models for Predicting Late Delivery

The classification models used in this project are Logistic Regression, Linear Discriminant Analysis, Gaussian Naive Bayes, Support Vector Machines, and Random Forest classification to predict "Late Delivery" based on accuracy, recall and F1 score metrics.

Classification is a fundamental concept in supervised learning, where the goal is to predict a categorical target variable based on input features. It involves training a model using labeled data and then using that model to classify new, unseen data into predefined classes.

Evaluation metrics play a crucial role in assessing the performance of machine learning models. They allow us to measure how well a model generalizes to new, unseen data and how effectively it accomplishes its task.

Confusion Matrix:

In the world of machine learning and classification algorithms, the *Confusion Matrix* is a powerful tool that helps us evaluate the performance of our models. It provides a clear and detailed breakdown of how well our model is making predictions on different classes.

A **Confusion Matrix** is a table that allows us to evaluate the performance of a classification model by comparing the actual classes of the data with the predicted classes made by the model. It helps us understand how well the model is distinguishing between different classes.

Structure and Elements of a Confusion Matrix:

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positive (**TP**)	False Negative (**FN**) Type 2 Error
Actual Negative	False Positive (**FP**) Type 1 Error	True Negative (**TN**)

Main Components:

1. **True Positives (TP):** The number of observations correctly predicted as positive by the model.
2. **False Positives (FP):** The number of observations that were predicted as positive but actually belong to the negative class.
3. **True Negatives (TN):** The number of observations correctly predicted as negative by the model.
4. **False Negatives (FN):** The number of observations that were predicted as negative but actually belong to the positive class.

Using the Confusion Matrix:

With the confusion matrix, we can calculate several important metrics to evaluate our model's performance:

1. Accuracy:

Accuracy is a fundamental concept in machine learning that measures how well a model performs on a given dataset. It is the ratio of correctly predicted instances to the total number of instances in the dataset. In this content, we will explore the concept of accuracy in machine learning, its importance, calculation, and potential pitfalls.

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

Points to Remember:

- High accuracy is desirable, but it may not be the only metric to consider depending on the problem domain.
- Accuracy can be misleading, especially when dealing with imbalanced datasets.
- Always consider other evaluation metrics such as precision, recall, F1-score, etc., for a comprehensive analysis.

Limitations of Accuracy:

- Accuracy can be misleading when classes are imbalanced. For instance, if 95% of instances belong to class A and only 5% to class B, a model that predicts all instances as class A would still achieve 95% accuracy.
- Accuracy doesn't reveal the types of errors a model makes, and it treats all misclassifications equally.
- The accuracy metric alone may not be sufficient to evaluate a model's performance, especially in cases where false negatives or false positives have different consequences.

Improving Accuracy: To enhance accuracy, consider the following techniques:

- **Feature Engineering:** Carefully select relevant features and discard irrelevant ones to improve model performance.
- **Hyperparameter Tuning:** Optimize hyperparameters to find the best configuration for your model.
- **Ensemble Methods:** Combine multiple models to leverage their strengths and improve overall accuracy.
- **Data Augmentation:** Increase the diversity of the training data by applying transformations, which can help the model generalize better.
- **Addressing Class Imbalance:** Use techniques like oversampling, undersampling, or generating synthetic samples to balance the classes in the dataset.

2. Precision:

Precision is a crucial metric used to evaluate the performance of a machine learning model, especially in classification tasks. It measures the accuracy of the positive predictions made by the model, also known as the positive predictive value. In simple terms, precision answers the question: "Of all the instances the model predicted as positive, how many were actually positive?"

Formula for Precision:-

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Key Points to Remember:

1. **High Precision** indicates that when the model predicts a positive class, it is likely to be correct.
2. **Low Precision** means that the model is making many false positive predictions.
3. Precision is particularly important when the cost of false positives is high, and we want to minimize the number of false alarms.
4. It is not affected by the number of true negatives, which means that precision can still be high even if the model predicts many negatives correctly.
5. Precision is relevant in cases where the class distribution is imbalanced. For example, in fraud detection, where the number of fraud cases is relatively low compared to non-fraud cases.

3. Recall (Sensitivity or True Positive Rate):

Recall is particularly important in scenarios where the cost of false negatives is high, and missing positive instances can lead to severe consequences. For example, in medical diagnosis, it is crucial to have a high recall to ensure that all cases of a particular disease are identified, even if it means some false positives. It's calculated as:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Where:

- True Positives (TP) are the instances correctly predicted as positive.
- False Negatives (FN) are the instances incorrectly predicted as negative but are actually positive.

Points to Remember:

1. Recall focuses on the ability of a model to find all positive instances, minimizing false negatives.

2. High recall is desirable when the cost of false negatives is significant.
3. Recall values lie in the range of 0 to 1, where 1 indicates perfect recall (all relevant instances are correctly predicted) and 0 indicates no recall (none of the relevant instances are predicted).
4. Unlike precision, recall is not affected by the number of true negatives.

4. F1 Score:

The F1 score is a commonly used metric in machine learning and statistics to evaluate the performance of a binary classification model. It is particularly useful when dealing with imbalanced datasets, where one class is more prevalent than the other. The F1 score takes into account both precision and recall, providing a balanced evaluation of a model's accuracy.

Formula: The F1 score is calculated using the following formula:

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Key Points to Remember:

- The F1 score is a value between 0 and 1, where 1 indicates perfect precision and recall, and 0 indicates poor performance.
- The F1 score is a harmonic mean of precision and recall, giving equal weightage to both metrics.
- It is beneficial to use the F1 score when you want to strike a balance between precision and recall in imbalanced datasets.

```
from sklearn import metrics

import matplotlib.colors as mcolors

ml_df=df

ml_df['Late delivery'] = np.where(df['Delivery Status'] == 'Late delivery',
1, 0)

ml_df.drop('Product Status',axis=1,inplace=True)

# Selects the categorical columns from the training dataset.
cat_columns = ml_df.select_dtypes(include='object').columns

# Selects the numerical columns from the training dataset.
num_columns=ml_df.select_dtypes(exclude='object').columns
```

Label Encoding

Label encoding is a technique used in data preprocessing and machine learning to convert categorical variables into numerical format. It assigns a unique numeric value to each category, making it suitable for algorithms that require numerical input. Let's delve into the details of label encoding and how to use it effectively.

How Label Encoding Works:

Label encoding involves the following steps:

1. Identify Categorical Variables: Determine which features in your dataset are categorical, meaning they have distinct categories or labels.
2. Assign Numeric Labels: For each category in a categorical feature, assign a unique integer. The assignment can be arbitrary, but it's important to maintain consistency.
3. Replace Categorical Values: Replace the categorical values with their corresponding numeric labels in the dataset.

Advantages of Label Encoding:

1. Simplicity: Label encoding is straightforward and easy to implement.
2. Preserves Order: If there's an inherent order among the categories, like low, medium, and high, label encoding captures this relationship.
3. Suitable for Tree-Based Models: Algorithms like decision trees and random forests can effectively use label encoded data.

Considerations and Limitations:

1. Misinterpretation: Some algorithms might interpret label encoded values as having meaningful magnitudes, which could lead to incorrect conclusions.
2. Impact on Model Performance: Label encoding might not be suitable for algorithms that rely on distances or similarities between feature values.

Best Practices:

1. Use with Care: Label encoding is most effective for nominal categorical variables (without a defined order).
2. Feature Engineering: Consider if creating new features or using one-hot encoding might better capture the information in your data.
3. Combining with Other Techniques: In complex datasets, label encoding can be combined with other encoding methods for better results.

```
from sklearn.preprocessing import LabelEncoder

le=LabelEncoder()
def Labelencoder_feature(x):
    le=LabelEncoder()
    x=le.fit_transform(x)
    return x

for i in cat_columns:
    ml_df[i]  = le.fit_transform(ml_df[i])
```

```
late_x=ml_df.drop(['Late delivery','Late_delivery_risk','Delivery
Status','Latitude','Longitude','Department Id','Order Customer Id','Order
Id','Order Item Id','Product Card Id','Product Category Id','order date
(DateOrders)','shipping date (DateOrders)'],axis=1)


late_y=ml_df['Late delivery']
```

Train-Test Split

When developing a machine learning model, one crucial step is to divide your dataset into a training set and a test set. This division is known as the "train-test split." The train-test split is essential to assess the performance of your model accurately and prevent overfitting. In this guide, we'll dive into the details of how to perform a train-test split, its significance, and best practices.

Why Use a Train-Test Split: A train-test split helps you evaluate the performance of your machine learning model on unseen data. If you train and test your model on the same dataset, it might perform well on that data but struggle with new data. By reserving a portion of your data for testing, you simulate real-world scenarios and gain insights into how well your model generalizes.

```
from sklearn.model_selection import train_test_split
xlatedelivery_train,
xlatedelivery_test,ylatedelivery_train,ylatedelivery_test =
train_test_split(late_x,late_y,test_size = 0.3, random_state = 5)
```

Standardizing Data

Standardization is a crucial preprocessing step in data analysis and machine learning. It involves transforming your data in a way that it has a mean of 0 and a standard deviation of 1. This process is especially important when dealing with features that have different scales, as it helps algorithms converge faster and produce more accurate results. One widely used method for standardization is the **Standard Scaler**.

What is the Standard Scaler?

The Standard Scaler, also known as Z-score normalization, is a technique used to transform your data's features to have a mean of 0 and a standard deviation of 1. It does this by subtracting the mean from each data point and then dividing by the standard deviation.

How does it work?

Let's say you have a dataset with a feature, "Feature A," that has varying values. The formula for standardizing a data point x is:

$$z = (x - \mu) / \sigma$$

Where:

- **z** is the standardized value.
- **x** is the original value of the data point.
- **μ** is the mean of the feature.
- **σ** is the standard deviation of the feature.

Benefits of Standardization:

1. **Mean Centering:** Standardization centers the data around zero, making it easier to compare different features.
2. **Equalized Scale:** By giving all features the same scale (mean of 0 and standard deviation of 1), algorithms that rely on distance metrics or gradients can work more effectively.
3. **Outlier Handling:** Standardization reduces the influence of outliers since they are now measured in terms of standard deviations from the mean.

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
xlatedelivery_train=sc.fit_transform(xlatedelivery_train)
xlatedelivery_test=sc.transform(xlatedelivery_test)
```

Considerations:

- It's important to fit the Standard Scaler on the training data and then use the same scaler to transform both the training and test datasets.
- Be cautious when standardizing sparse datasets, as it might result in undesired effects.

The models are evaluated using accuracy, recall, F1 score metrics since the output is in binary classification format. The F1 score is the primary metric used to measure the performance of different models.

```
from sklearn.metrics import accuracy_score, recall_score, confusion_matrix, f1_score
```

1. Random Forests

Definition: Random Forests are an ensemble learning method that combines multiple decision trees to improve the accuracy and robustness of predictions.

Decision Trees are versatile supervised learning algorithms used for both classification and regression tasks. They create a tree-like model where each node represents a decision based on feature(s).

The tree starts from the root node and repeatedly splits data into subsets based on the most informative features. It continues until it reaches leaf nodes, which contain the final predictions.

How it works: Random Forests create several decision trees using different subsets of the training data and features. Each tree independently makes a prediction, and the final result is obtained through a voting process or averaging the outputs.

Example: Imagine you want to predict house prices based on various features like location, size, and number of bedrooms. Random Forests will create multiple decision trees, and each tree will give its estimate of the house price. The final prediction will be the average of these individual estimates.

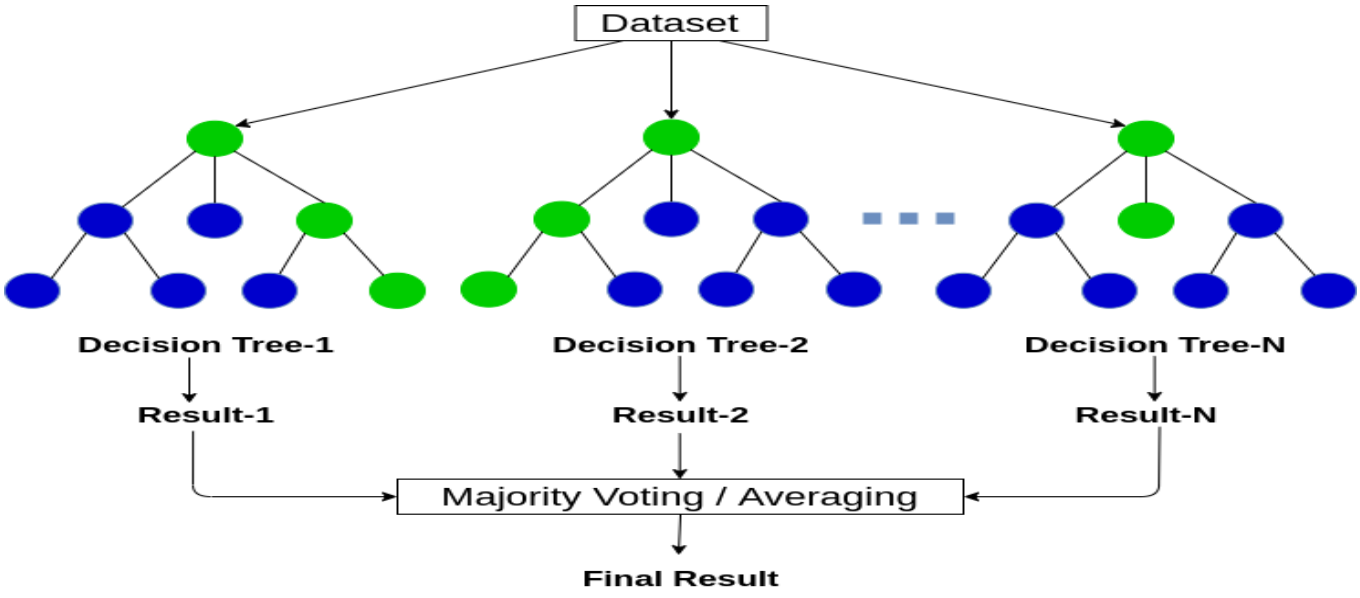

Figure 10 Random Forests diagram

Points to Remember:

- Reduces overfitting compared to a single decision tree.
- Handles a large number of features well.

```
from sklearn.ensemble import RandomForestClassifier

# Create a RandomForestClassifier with specified hyperparameters
RandomForest_model_latedelivery = RandomForestClassifier(n_estimators=100,
max_depth=10, random_state=0)

# Fit the model on the training data
RandomForest_model_latedelivery.fit(xlatedelivery_train, ylatedelivery_train)

# Predict the late delivery status using the trained model
RandomForest_ylatedelivery_pred =
RandomForest_model_latedelivery.predict(xlatedelivery_test)

# Calculate evaluation metrics
rf_accuracy_latedelivery = accuracy_score(RandomForest_ylatedelivery_pred,
ylatedelivery_test)
rf_recall_latedelivery = recall_score(RandomForest_ylatedelivery_pred,
ylatedelivery_test)
rf_conf_latedelivery = confusion_matrix(ylatedelivery_test,
RandomForest_ylatedelivery_pred)
rf_f1_latedelivery = f1_score(ylatedelivery_test,
RandomForest_ylatedelivery_pred)

# Print the results
print('Model parameters used are:', RandomForest_model_latedelivery)
print('Accuracy of late delivery status is:', rf_accuracy_latedelivery * 100,
'%')
print('Recall score of late delivery status is:', rf_recall_latedelivery *
100, '%')
print('F1 score of late delivery status is:', rf_f1_latedelivery * 100, '%')


# Extract values from the confusion matrix
rf_TN = rf_conf_latedelivery[0, 0]
rf_FP = rf_conf_latedelivery[0, 1]
rf_FN = rf_conf_latedelivery[1, 0]
rf_TP = rf_conf_latedelivery[1, 1]

# Define class labels
class_labels = ['Not Late', 'Late']

# Create a custom color map with green and red
cmap_colors = ["#e74c3c","#2ecc71"]  # Green and Red colors
cmap = sns.color_palette(cmap_colors)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 6))
```

```
sns.heatmap(rf_conf_latedelivery, annot=True, fmt="d",
cmap=cmap_colors,cbar=False, xticklabels=class_labels,
yticklabels=class_labels,annot_kws={"fontsize": 14})
plt.xlabel('Predicted',fontsize=14)
plt.ylabel('Actual',fontsize=14)
plt.title('RandomForestClassifier Confusion Matrix - Late Delivery
Prediction',fontsize=14)


# Add labels to the cells
plt.text(0.5, 0.3, 'True Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 0.3, 'False Positive', ha='center', va='center', color='white',
fontsize=14)
plt.text(0.5, 1.3, 'False Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 1.3, 'True Positive', ha='center', va='center', color='white',
fontsize=14)

# Adjust font size of class labels using tick_params
plt.tick_params(axis='both', which='major', labelsize=12)

plt.show()
```

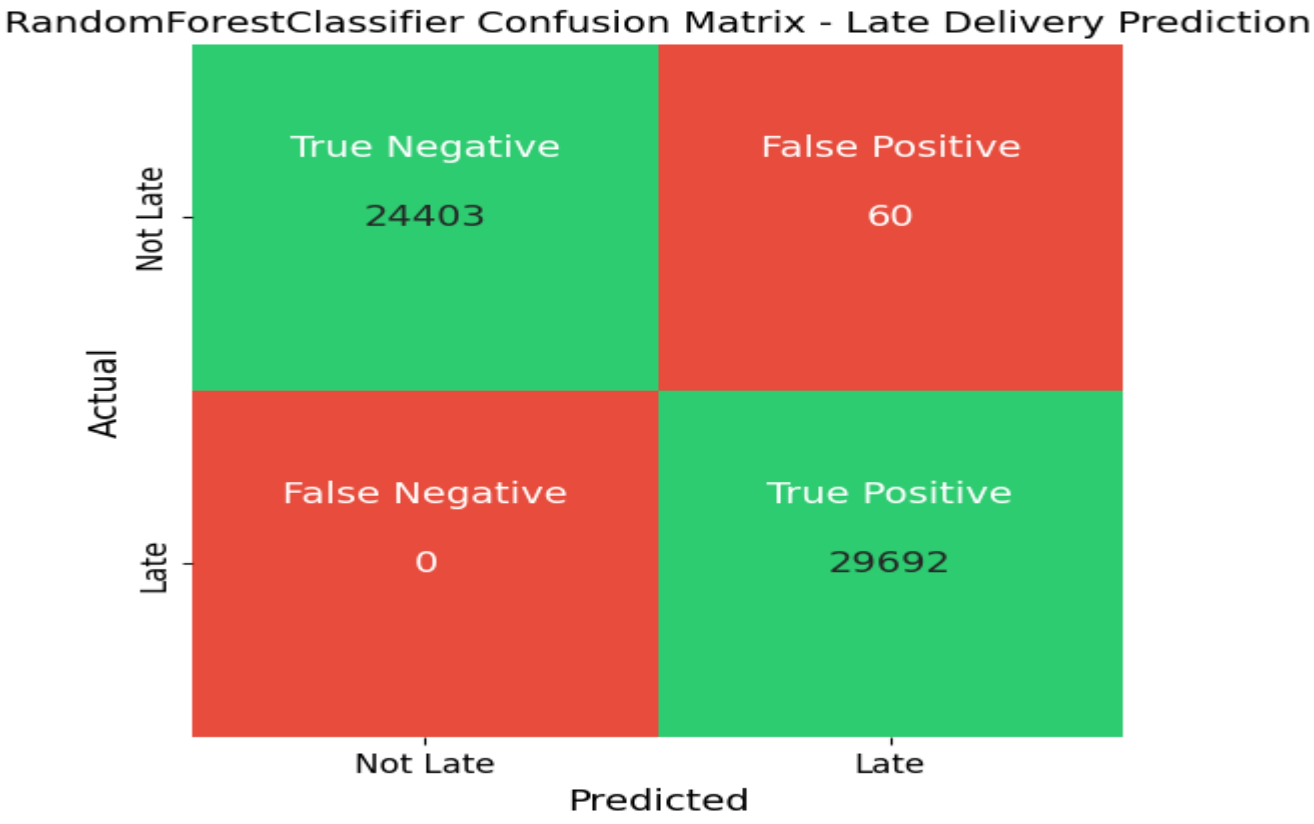

Figure 11 Random Forests Confusion Matrix

2. Support Vector Machines (SVM)

Definition: Support Vector Machines are powerful supervised learning algorithms used for both classification and regression tasks. They aim to find the best hyperplane that separates the data into different classes.

How it works: SVM tries to find the optimal hyperplane by maximizing the margin between the two classes. The data points closest to the hyperplane (support vectors) are crucial for defining the decision boundary.

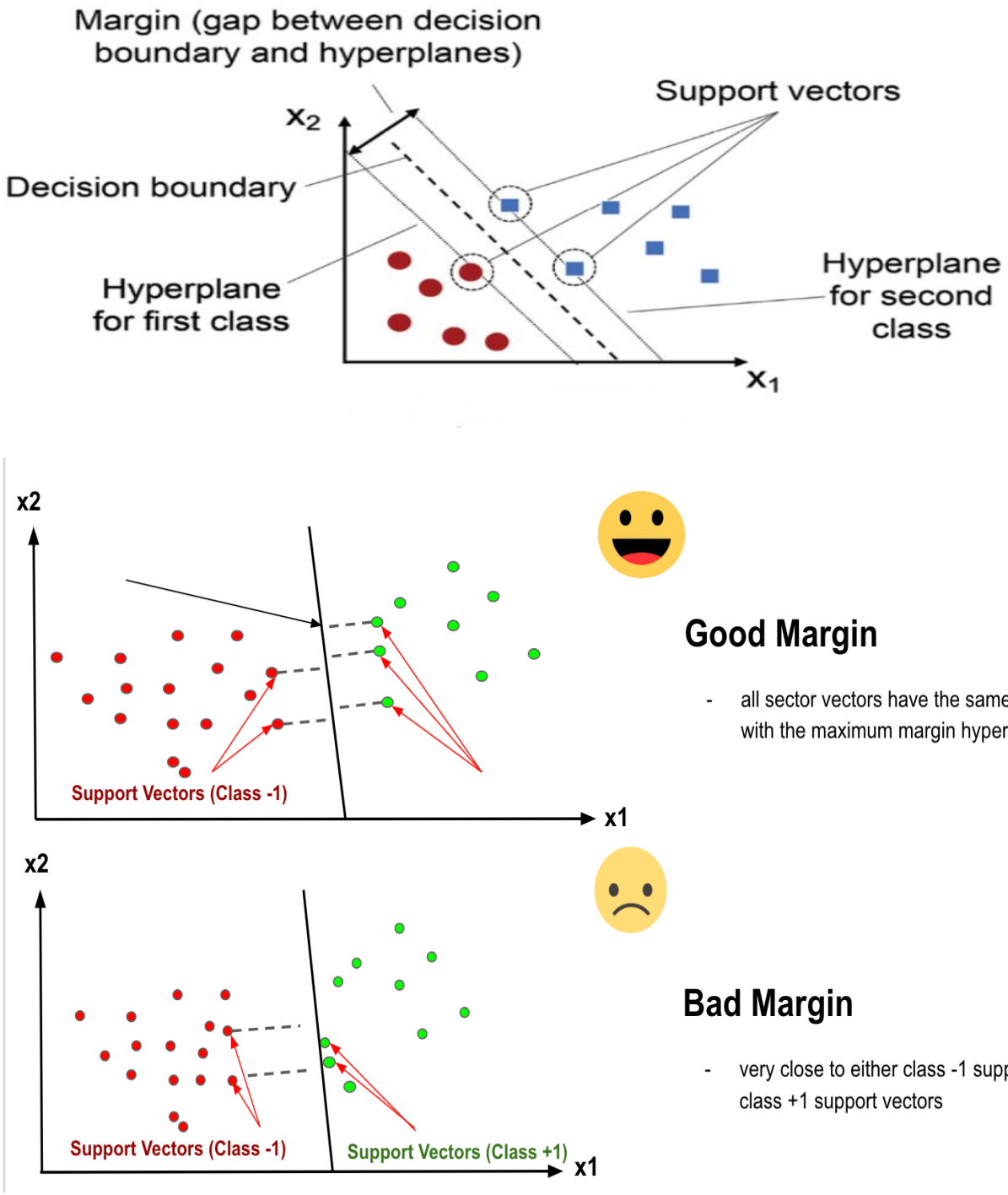

Figure 12 Support Vector Machines diagram

```
from sklearn.svm import LinearSVC

# Create a LinearSVC model
LinearSVC_model_latedelivery = LinearSVC(max_iter=5000)

# Fit the model on the training data
LinearSVC_model_latedelivery.fit(xlatedelivery_train, ylatedelivery_train)

# Predict the late delivery status using the trained model
LinearSVC_ylatedelivery_pred =
LinearSVC_model_latedelivery.predict(xlatedelivery_test)

# Calculate evaluation metrics
lsvc_accuracy_latedelivery = accuracy_score(LinearSVC_ylatedelivery_pred,
ylatedelivery_test)
lsvc_recall_latedelivery = recall_score(LinearSVC_ylatedelivery_pred,
ylatedelivery_test)
lsvc_conf_latedelivery = confusion_matrix(ylatedelivery_test,
LinearSVC_ylatedelivery_pred)
lsvc_f1_latedelivery = f1_score(ylatedelivery_test,
LinearSVC_ylatedelivery_pred)

# Print the results
print('Model parameters used are:', LinearSVC_model_latedelivery)
print('Accuracy of late delivery status is:', lsvc_accuracy_latedelivery *
100, '%')
print('Recall score of late delivery status is:', lsvc_recall_latedelivery *
100, '%')
print('F1 score of late delivery status is:', lsvc_f1_latedelivery * 100,
'%')

# Extract values from the confusion matrix
lsvc_TN = lsvc_conf_latedelivery[0, 0]
lsvc_FP = lsvc_conf_latedelivery[0, 1]
lsvc_FN = lsvc_conf_latedelivery[1, 0]
lsvc_TP = lsvc_conf_latedelivery[1, 1]

# Define class labels
class_labels = ['Not Late', 'Late']

# Create a custom color map with green and red
cmap_colors = ["#e74c3c","#2ecc71"]  # Green and Red colors
cmap = sns.color_palette(cmap_colors)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 6))
```

```
sns.heatmap(lsvc_conf_latedelivery, annot=True, fmt="d",
cmap=cmap_colors,cbar=False, xticklabels=class_labels,
yticklabels=class_labels,annot_kws={"fontsize": 14})
plt.xlabel('Predicted',fontsize=14)
plt.ylabel('Actual',fontsize=14)
plt.title('LinearSVC Confusion Matrix - Late Delivery
Prediction',fontsize=14)


# Add labels to the cells
plt.text(0.5, 0.3, 'True Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 0.3, 'False Positive', ha='center', va='center', color='white',
fontsize=14)
plt.text(0.5, 1.3, 'False Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 1.3, 'True Positive', ha='center', va='center', color='white',
fontsize=14)

# Adjust font size of class labels using tick_params
plt.tick_params(axis='both', which='major', labelsize=12)

plt.show()
```

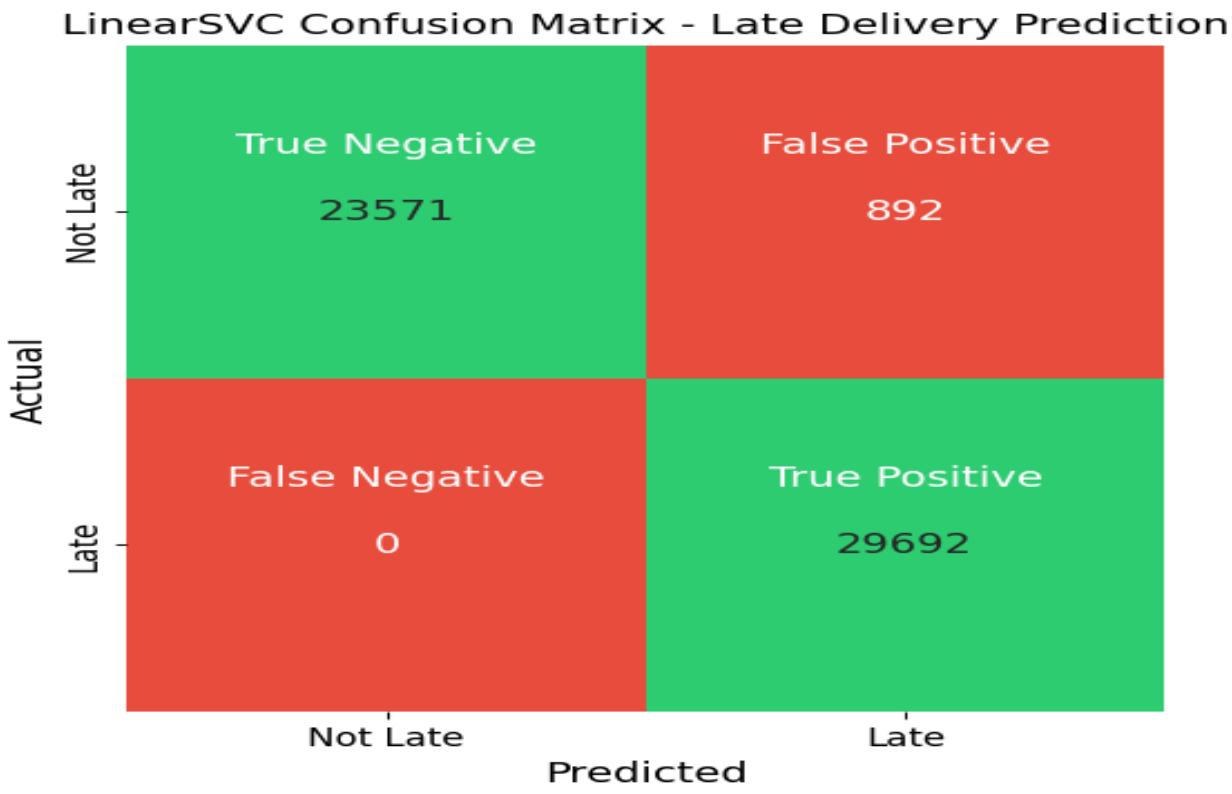

Figure 13 Linear Support Vector Machines (SVM) Confusion Matrix

3. Logistic Regression

Definition: Logistic regression is used for binary classification, where the dependent variable (target) is categorical with two possible outcomes (e.g., yes/no, true/false).

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, **it gives the probabilistic values which lie between 0 and 1**.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems**.

The curve from the logistic function indicates the likelihood of something such as whether the cells are cancerous or not, a mouse is obese or not based on its weight, etc.

Logistic Regression is a significant machine learning algorithm because it has the ability to provide probabilities and classify new data using continuous and discrete datasets.

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:

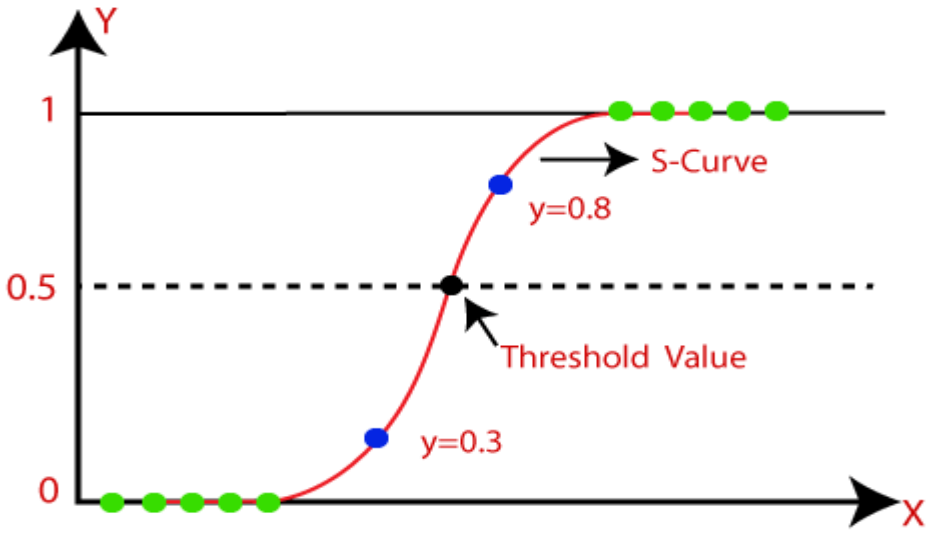

Figure 14 logistic function

Note: Logistic regression uses the concept of predictive modeling as regression; therefore, it is called logistic regression, but is used to classify samples; Therefore, it falls under the classification algorithm.

Logistic Function (Sigmoid Function):

- The sigmoid function is a mathematical function used to map the predicted values to probabilities.
- It maps any real value into another value within a range of 0 and 1.

- The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.
- In logistic regression, we use the concept of the threshold value, which defines the probability of either 0 or 1. Such as values above the threshold value tends to 1, and a value below the threshold values tends to 0.

Assumptions for Logistic Regression:

- The dependent variable must be categorical in nature.
- The independent variable should not have multi-collinearity.

```
from sklearn.linear_model import LogisticRegression

# Create a Logistic Regression model
lr_model_latedelivery = LogisticRegression(solver='lbfgs', random_state=0)

# Fit the model on the training data
lr_model_latedelivery.fit(xlatedelivery_train, ylatedelivery_train)

# Predict the late delivery status using the trained model
lr_ylatedelivery_pred = lr_model_latedelivery.predict(xlatedelivery_test)

# Calculate evaluation metrics
lr_accuracy_latedelivery = accuracy_score(lr_ylatedelivery_pred,
ylatedelivery_test)
lr_recall_latedelivery = recall_score(lr_ylatedelivery_pred,
ylatedelivery_test)
lr_conf_latedelivery = confusion_matrix(ylatedelivery_test,
lr_ylatedelivery_pred)
lr_f1_latedelivery = f1_score(ylatedelivery_test, lr_ylatedelivery_pred)

# Print the results
print('Model parameters used are:', lr_model_latedelivery)
print('Accuracy of late delivery status is:', lr_accuracy_latedelivery * 100,
'%')
print('Recall score of late delivery status is:', lr_recall_latedelivery *
100, '%')
print('F1 score of late delivery status is:', lr_f1_latedelivery * 100, '%')

# Extract values from the confusion matrix
lr_TN = lr_conf_latedelivery[0, 0]
lr_FP = lr_conf_latedelivery[0, 1]
lr_FN = lr_conf_latedelivery[1, 0]
lr_TP = lr_conf_latedelivery[1, 1]

# Define class labels
```

```
class_labels = ['Not Late', 'Late']

# Create a custom color map with green and red
cmap_colors = ["#e74c3c","#2ecc71"]  # Green and Red colors
cmap = sns.color_palette(cmap_colors)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(lr_conf_latedelivery, annot=True, fmt="d",
cmap=cmap_colors,cbar=False, xticklabels=class_labels,
yticklabels=class_labels,annot_kws={"fontsize": 14})
plt.xlabel('Predicted',fontsize=14)
plt.ylabel('Actual',fontsize=14)
plt.title('LogisticRegression Confusion Matrix - Late Delivery
Prediction',fontsize=14)

# Add labels to the cells
plt.text(0.5, 0.3, 'True Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 0.3, 'False Positive', ha='center', va='center', color='white',
fontsize=14)
plt.text(0.5, 1.3, 'False Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 1.3, 'True Positive', ha='center', va='center', color='white',
fontsize=14)

# Adjust font size of class labels using tick_params
plt.tick_params(axis='both', which='major', labelsize=12)
plt.show()
```

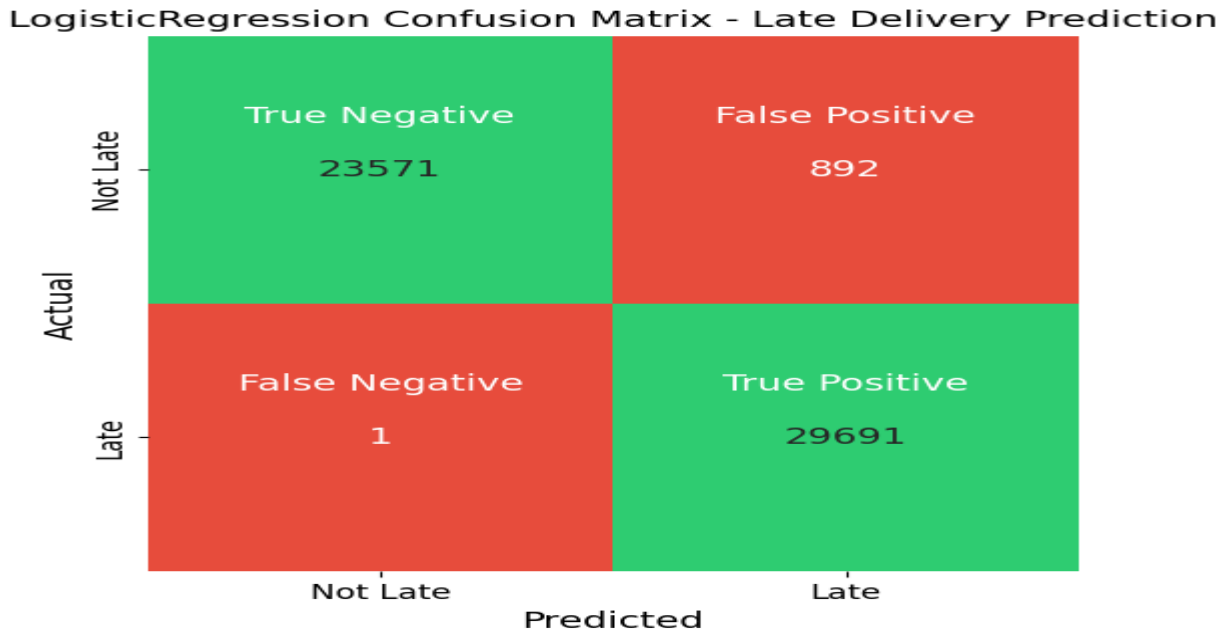

Figure 15 Logistic Regression Confusion Matrix

4. Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is a powerful statistical technique used for dimensionality reduction and classification tasks. It aims to find the linear combinations of features that best separate different classes in a dataset, making it an essential tool in various fields such as machine learning, pattern recognition, and statistics.

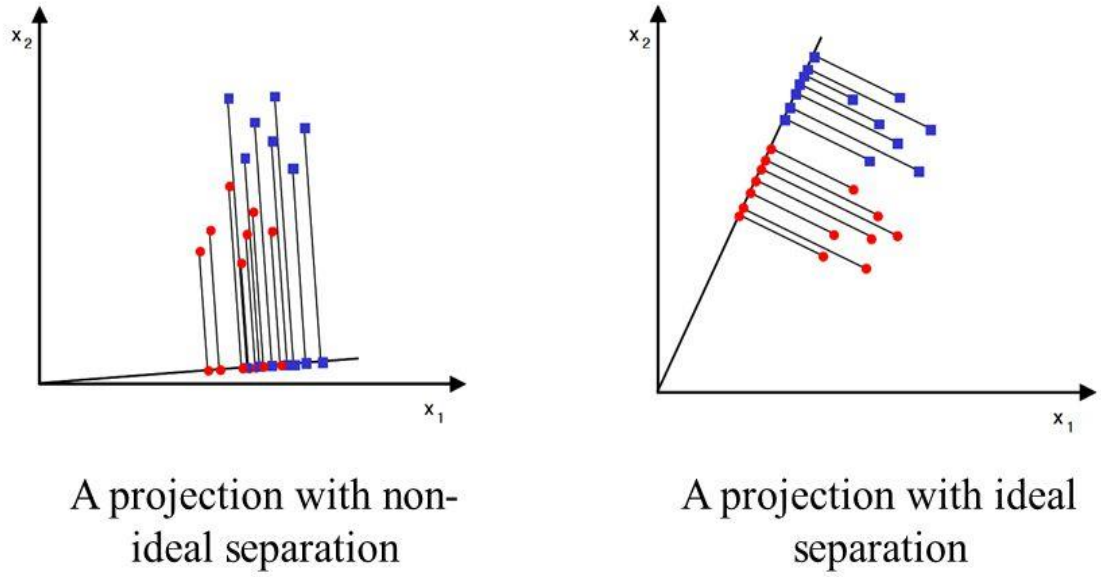

Figure 16 Linear Discriminant Analysis graph

Key Concepts of LDA:

1. **Discriminant Functions:** LDA computes discriminant functions to project data points onto a lower-dimensional space while maximizing the distance between classes and minimizing the variance within each class.
2. **Mean Vectors:** LDA involves calculating mean vectors for each class. These mean vectors help determine the central point of each class in the transformed space.
3. **Scatter Matrices:** LDA employs scatter matrices, namely the within-class scatter matrix and the between-class scatter matrix. The within-class scatter matrix measures the spread of data points within each class, while the between-class scatter matrix gauges the separation between class means.

```
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis

# Create a Linear Discriminant Analysis model
ld_model_latedelivery = LinearDiscriminantAnalysis()

# Fit the model on the training data
ld_model_latedelivery.fit(xlatedelivery_train, ylatedelivery_train)

# Predict the late delivery status using the trained model
```

```
ld_ylatedelivery_pred = ld_model_latedelivery.predict(xlatedelivery_test)

# Calculate evaluation metrics
ld_accuracy_latedelivery = accuracy_score(ld_ylatedelivery_pred,
ylatedelivery_test)
ld_recall_latedelivery = recall_score(ld_ylatedelivery_pred,
ylatedelivery_test)
ld_conf_latedelivery = confusion_matrix(ylatedelivery_test,
ld_ylatedelivery_pred)
ld_f1_latedelivery = f1_score(ylatedelivery_test, ld_ylatedelivery_pred)

# Print the results
print('Model parameters used are:', ld_model_latedelivery)
print('Accuracy of late delivery status is:', ld_accuracy_latedelivery * 100,
'%')
print('Recall score of late delivery status is:', ld_recall_latedelivery *
100, '%')
print('F1 score of late delivery status is:', ld_f1_latedelivery * 100, '%')

# Extract values from the confusion matrix
ld_TN = ld_conf_latedelivery[0, 0]
ld_FP = ld_conf_latedelivery[0, 1]
ld_FN = ld_conf_latedelivery[1, 0]
ld_TP = ld_conf_latedelivery[1, 1]

# Define class labels
class_labels = ['Not Late', 'Late']

# Create a custom color map with green and red
cmap_colors = ["#e74c3c","#2ecc71"]  # Green and Red colors
cmap = sns.color_palette(cmap_colors)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(ld_conf_latedelivery, annot=True, fmt="d",
cmap=cmap_colors,cbar=False, xticklabels=class_labels,
yticklabels=class_labels,annot_kws={"fontsize": 14})
plt.xlabel('Predicted',fontsize=14)
plt.ylabel('Actual',fontsize=14)
plt.title('Linear Discriminant Analysis Confusion Matrix - Late Delivery
Prediction',fontsize=14)


# Add labels to the cells
plt.text(0.5, 0.3, 'True Negative', ha='center', va='center', color='white',
fontsize=14)
```

```
plt.text(1.5, 0.3, 'False Positive', ha='center', va='center', color='white',
fontsize=14)
plt.text(0.5, 1.3, 'False Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 1.3, 'True Positive', ha='center', va='center', color='white',
fontsize=14)

# Adjust font size of class labels using tick_params
plt.tick_params(axis='both', which='major', labelsize=12)

plt.show()
```

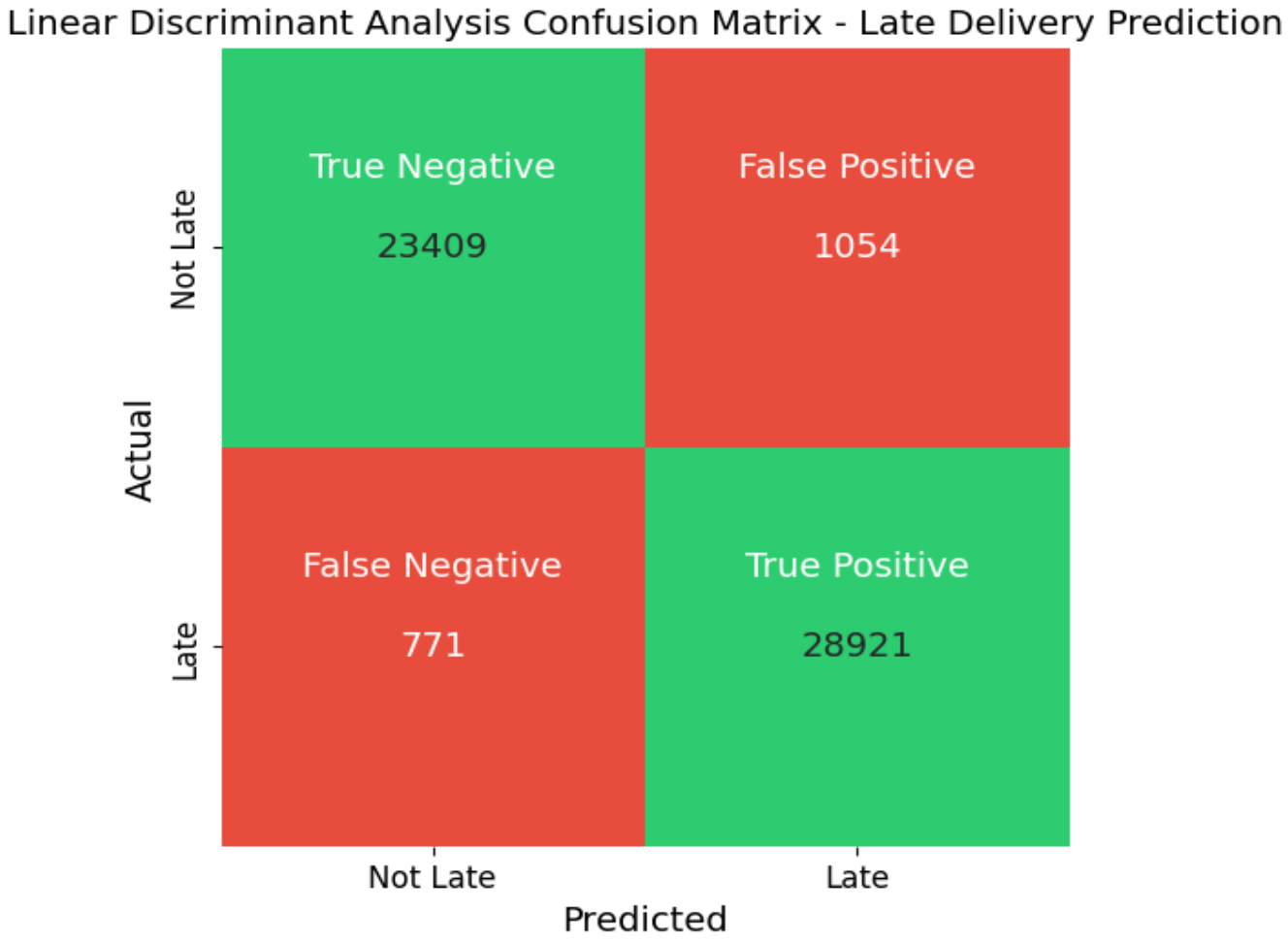

Figure 17 Linear Discriminant Analysis Confusion Matrix

5. Gaussian Naive Bayes

Definition: Naive Bayes is a probabilistic supervised learning algorithm used for classification tasks. It is based on Bayes' theorem and assumes that features are independent of each other, which is a naive assumption in real-world scenarios.

How it works: Naive Bayes calculates the probability of an input belonging to each class and selects the class with the highest probability as the final prediction.

In machine learning, naive Bayes classifiers are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

$$P(A|B) = \frac{P(B|A)\,P(A)}{P(B)}$$

using Bayesian probability terminology, the above equation can be written as

$$\text{Posterior} = \frac{\text{prior x likelihood}}{\text{evidence}}$$

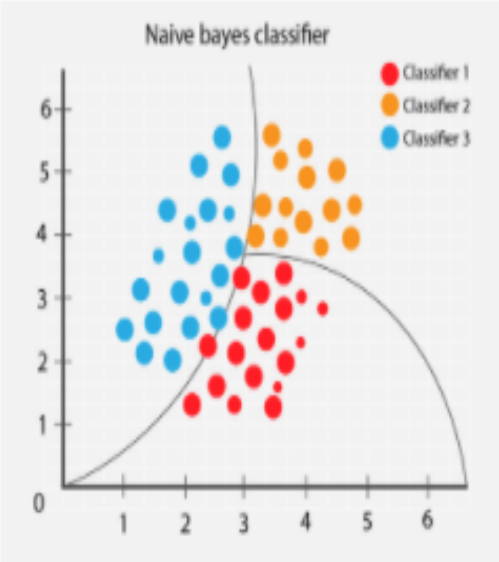

Figure 18 Gaussian Naive Bayes Fomuale and Graph

Points to Remember:

- Fast and simple algorithm.
- Suitable for text classification and spam filtering.
- Assumes independence between features, which may not always hold true.

```
from sklearn.naive_bayes import GaussianNB

# Create a Gaussian Naive Bayes model
gnb_model_latedelivery = GaussianNB()

# Fit the model on the training data
gnb_model_latedelivery.fit(xlatedelivery_train, ylatedelivery_train)

# Predict the late delivery status using the trained model
gnb_ylatedelivery_pred = gnb_model_latedelivery.predict(xlatedelivery_test)

# Calculate evaluation metrics
gnb_accuracy_latedelivery = accuracy_score(gnb_ylatedelivery_pred,
ylatedelivery_test)
gnb_recall_latedelivery = recall_score(gnb_ylatedelivery_pred,
ylatedelivery_test)
gnb_conf_latedelivery = confusion_matrix(ylatedelivery_test,
gnb_ylatedelivery_pred)
gnb_f1_latedelivery = f1_score(ylatedelivery_test, gnb_ylatedelivery_pred)

# Print the results
print('Model parameters used are:', gnb_model_latedelivery)
print('Accuracy of late delivery status is:', gnb_accuracy_latedelivery *
100, '%')
print('Recall score of late delivery status is:', gnb_recall_latedelivery *
100, '%')
print('F1 score of late delivery status is:', gnb_f1_latedelivery * 100, '%')


# Extract values from the confusion matrix
gnb_TN = gnb_conf_latedelivery[0, 0]
gnb_FP = gnb_conf_latedelivery[0, 1]
gnb_FN = gnb_conf_latedelivery[1, 0]
gnb_TP = gnb_conf_latedelivery[1, 1]

# Define class labels
class_labels = ['Not Late', 'Late']

# Create a custom color map with green and red
cmap_colors = ["#e74c3c","#2ecc71"]  # Green and Red colors
cmap = sns.color_palette(cmap_colors)

# Create a heatmap for the confusion matrix
plt.figure(figsize=(6, 6))
sns.heatmap(gnb_conf_latedelivery, annot=True, fmt="d",
cmap=cmap_colors,cbar=False, xticklabels=class_labels,
yticklabels=class_labels,annot_kws={"fontsize": 14})
```

```
plt.xlabel('Predicted',fontsize=14)
plt.ylabel('Actual',fontsize=14)
plt.title('Gaussian Naive Bayes Confusion Matrix - Late Delivery
Prediction',fontsize=14)


# Add labels to the cells
plt.text(0.5, 0.3, 'True Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 0.3, 'False Positive', ha='center', va='center', color='white',
fontsize=14)
plt.text(0.5, 1.3, 'False Negative', ha='center', va='center', color='white',
fontsize=14)
plt.text(1.5, 1.3, 'True Positive', ha='center', va='center', color='white',
fontsize=14)

# Adjust font size of class labels using tick_params
plt.tick_params(axis='both', which='major', labelsize=12)

plt.show()
```

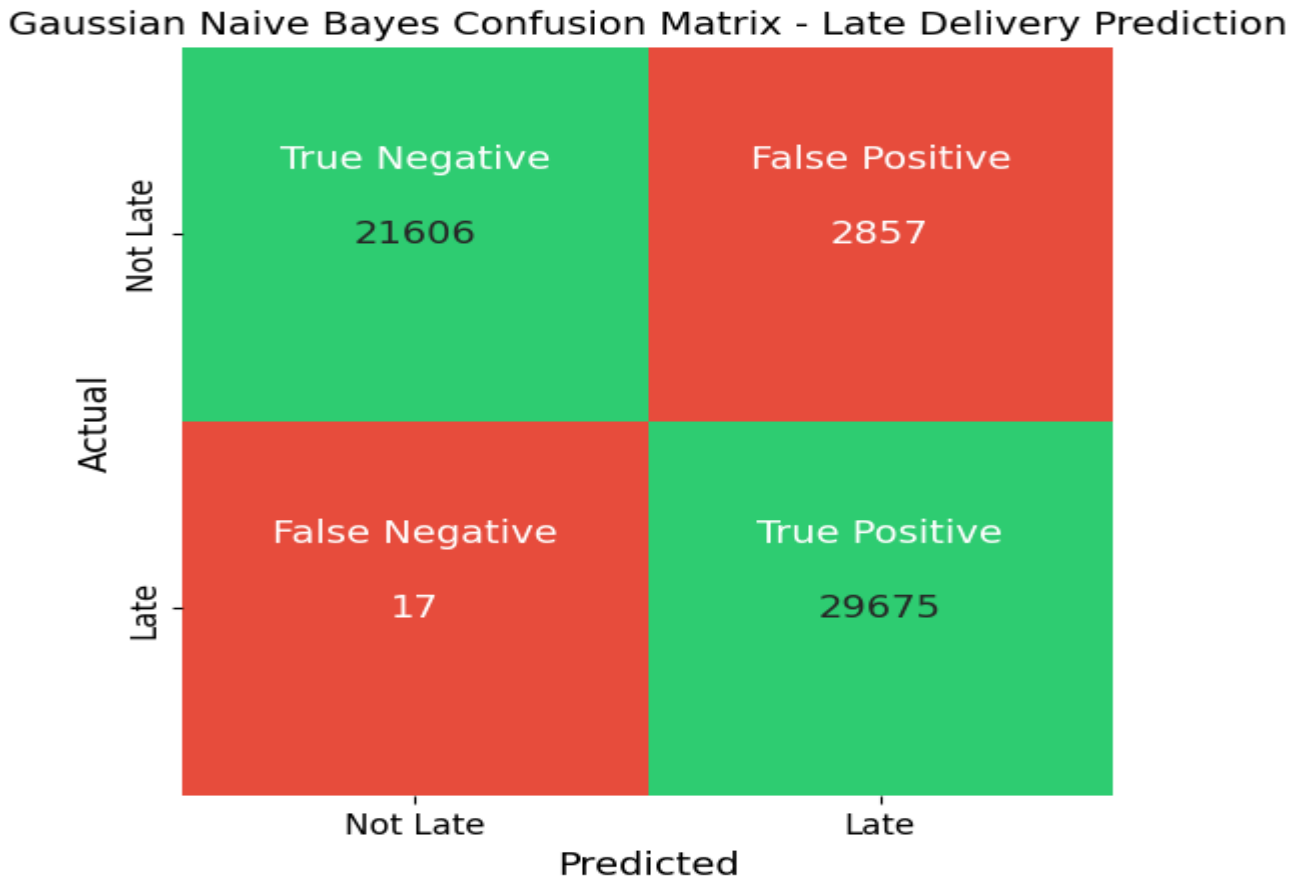

Figure 19 Gaussian Naive Bayes Confusion Matrix

Models Evaluation

In this section, we will analyze the performance of different machine learning classification algorithms on a classification task. The evaluation metrics used for comparison are F1 Score, Accuracy, Recall, True Negative, False Positive, False Negative, and True Positive.

```
from tabulate import tabulate
data = [["Random Forest Classification",
rf_f1_latedelivery*100,rf_accuracy_latedelivery*100,rf_recall_latedelivery*10
0, rf_TN, rf_FP, rf_FN, rf_TP],
        ["Support Vector Machines",
lsvc_f1_latedelivery*100,lsvc_accuracy_latedelivery*100,lsvc_recall_latedeliv
ery*100,lsvc_TN, lsvc_FP, lsvc_FN, lsvc_TP],
        ["Logistic Classification Model",
lr_f1_latedelivery*100,lr_accuracy_latedelivery*100,lr_recall_latedelivery*10
0,lr_TN, lr_FP, lr_FN, lr_TP],
        ["Linear Discriminant
Analysis",ld_f1_latedelivery*100,ld_accuracy_latedelivery*100,ld_recall_lated
elivery*100,ld_TN, ld_FP, ld_FN, ld_TP],
        ["Gaussian Naive Bayes
Model",gnb_f1_latedelivery*100,gnb_accuracy_latedelivery*100,gnb_recall_lated
elivery*100,gnb_TN, gnb_FP, gnb_FN, gnb_TP]]

col_names = ["Algorithms", "F1 Score", "Accuracy", "Recall","True Negative",
"False Positive", "False Negative", "True Positive"]
print(tabulate(data, headers=col_names))
```

Algorithms	F1 Score	Accuracy	Recall	True Negative	False Positive	False Negative	True Positive
Random Forest Classification	99.8991	99.8892	99.7983	24403	60	0	29692
Support Vector Machines	98.5201	98.3529	97.0834	23571	892	0	29692
Logistic Classification Model	98.5185	98.351	97.0833	23571	892	1	29691
Linear Discriminant Analysis	96.9414	96.63	96.4837	23409	1054	771	28921
Gaussian Naive Bayes Model	95.3812	94.693	91.2179	21606	2857	17	29675

Key Observations

- The **Random Forest Classification** algorithm exhibits exceptional performance across all metrics. It achieves the highest F1 Score, Accuracy, and Recall.
- Both **Support Vector Machines (SVM)** and **Logistic Classification Model** demonstrate competitive results, with SVM outperforming slightly in terms of F1 Score and Accuracy.
- The **Linear Discriminant Analysis (LDA)** algorithm provides good accuracy, but it struggles with False Positives and False Negatives.
- The **Gaussian Naive Bayes Model** shows relatively lower performance compared to other algorithms, especially in terms of Recall.

Conclusion

In this comparison of algorithm performance, the **Random Forest Classification** algorithm stands out as the top performer for the given classification task. However, the choice of algorithm should be made based on the specific characteristics of the dataset and the desired trade-offs between different evaluation metrics.

References

https://en.wikipedia.org/

https://machinelearningmastery.com/

https://towardsdatascience.com/

https://www.tutorialspoint.com/

https://www.geeksforgeeks.org /

https://www.scaler.com/

https://www.mygreatlearning.com/

https://www.dataquest.io/blog/

https://www.analyticsvidhya.com/blog/

https://builtin.com/