



# Purchase Pattern Analysis

---

Manu Bhaskar  
Avantika Roy  
Priyam Pal  
Subhas Mukherjee  
Jit Nandi

## Table of Contents

Topic	Pg. No
<b>Executive Summary.....</b>	<b>3</b>
<b>Objectives.....</b>	<b>3</b>
<b>Tasks and Approach/Methodology.....</b>	<b>3</b>
<b>Diliverables.....</b>	<b>4</b>
<b>Dataset Description.....</b>	<b>4</b>
products.csv.....	4
orders.csv.....	4
orders_products.csv.....	5
aisles.csv.....	5
<b>Preprocessing.....</b>	<b>6</b>
products.csv.....	6
orders.csv.....	6
order_products.csv.....	6
aisles.csv.....	7
<b>Exploratory Data Analysis.....</b>	<b>8</b>
Univariate Analysis.....	8
Bivariate Analysis.....	18
Multivariate Analysis.....	24
<b>Data Encoding.....</b>	<b>29</b>
ENCODING - 1: Applying Target Encoding of aisle, department, product_name.....	29
ENCODING - 2: Applying One-hot Encoding on day_of_week.....	30
ENCODING - 3: Applying Cyclic Encoding and Performing Sin-Cosine Transformation on order_hour_of_day.....	31
ENCODING - 4: Applying Ordinal Encoding on add_to_card_order.....	32
ENCODING - 5: Apply Binning on day_since_prior_order.....	33
<b>FEATURE ENGINEERING.....</b>	<b>34</b>
Feature I -- average_days_between_purchases:.....	34
Feature II -- Product_purchase_frequency:.....	35
Feature III -- total_purchases.....	36
Feature IV -- interval_std_dev.....	36
Feature V -- Product_reorder_rate:.....	37
Feature VI -- Users_general_reorder_rate:.....	38

Feature VII – Avg_add_to_cart_order:.....	39
Correlation Matrix - HeatMap.....	40
<b>Market Basket Analysis &amp; Product Bundling.....</b>	<b>42</b>
DATA LOADING AND PREPARATION.....	42
DATA FILTERING AND TRANSFORMATION.....	42
MARKET BASKET ANALYSIS WITH APRIORI ALGORITHM.....	43
FREQUENT ITEM SET.....	43
PRODUCT BUNDLING STRATEGY.....	44
INSIGHTS AND APPLICATIONS FOR PRODUCTBUNDLING.....	47
<b>Customer Segmentation (Clustering).....</b>	<b>48</b>
Collecting and Loading Data.....	48
Merging the Datasets.....	48
Sampling Data for Easier Analysis.....	48
Adding Aisle Names.....	48
Cross-Tabulating Users and Aisles.....	49
Reducing Dimensions with PCA (Principal Component Analysis).....	49
Finding Clusters with K-Means.....	49
Visualizing Clusters.....	50
Finding Popular Products in Each Cluster.....	50
Final Insights.....	51
<b>Next Purchase Prediction (Supervised Learning).....</b>	<b>52</b>
Initial Model Selection:.....	52
Final Model Selection.....	54
Final Model Evaluation.....	56
<b>Hyperparameter Tuning for Purchase Prediction Model.....</b>	<b>59</b>
Model Details - LightGBM.....	59
Hyperparameter Tuning Methodology.....	59
Experiments and Results.....	60
Performance Gains.....	62
<b>Conclusion.....</b>	<b>63</b>

## Executive Summary

The aim of this project is to uncover customer purchase patterns and create a predictive model that will help the business to identify the next likely purchase of a customer. The extrapolated patterns will support the creation of product bundles which would ultimately be used to increase the Average Order Value (AOV) of the products.

The project would require several steps like Exploratory Analysis to identify the patterns and trends in the data. This analysis would help in creating distinct groups of customers based on their purchase pattern and discover frequent itemsets/products bought together and ultimately contribute to prediction of the next likely purchase of customers.

## Objectives

1. Identification of popular product bundles and prediction of next-purchase products to help optimize marketing strategies, such as product bundling and targeted promotions.
2. Increase of average order value (AOV) by encouraging customers to buy bundles of frequently purchased products together.

## Tasks and Approach/Methodology

- I. Exploratory Data Analysis (EDA)
- II. Data Preprocessing and feature Engineering
- III. Popular Product Bundles Identification
- IV. Customer Segmentation(Clustering)
- V. Predicting Next Purchase(Supervised Learning)
- VI. Hyperparameter Tuning

## Diliverables

- I. Eda Report
- II. DataPreprocessing
- III. Market Basket Analysis
- IV. Customer Segmentation
- V. Next- Purchase prediction Model
- VI. Recommendation Engine
- VII. Marketing Insights

## Dataset Description

### products.csv

This dataset contains data on product ID, their names, the aisles they are stored in and the department they are located in. The **product\_id** column is where numerical values are stored, which are unique to each product; no two products share the same ID.

The **product\_name** column stores the names of the products that are available in the store. The **product\_id** column corresponds to these item names. This feature by itself does not serve any purpose to the analysis but helps identify the type of product the customers of the store usually buy.

The **aisle\_id** column stores unique numerical values for aisles. This also helps identify the aisle in which the items are stored.

The **department\_id** also stores unique numerical values that help identify in which department an aisle is present and thus which department an item is present in.

### orders.csv

User Behaviour:

- **User Loyalty:** The user\_id feature can be used to identify repeat customers and analyze their purchasing patterns. Frequent orders indicate a high level of customer loyalty and satisfaction.
- **Order Frequency:** The days\_since\_prior\_order feature provides insights into how often customers make purchases. By analyzing the distribution of this feature, we can identify trends in customer purchasing behavior and identify potential opportunities for targeted marketing campaigns.

### Order Pattern:

- **Order Timing:** The `order_dow` and `order_hour_of_day` features provide insights into when customers are most likely to place orders. This information can be used to optimize staffing schedules, delivery operations, and marketing campaigns.
- **Order Volume:** The `order_number` feature indicates the number of orders placed by each user. Analyzing the distribution of order numbers can help identify high-value customers and potential opportunities for upselling and cross-selling.

### orders\_products.csv

#### Product Information:

- **Product ID:** This unique identifier helps track individual products and their associated attributes.
- **Add-to-Cart Order:** This feature provides information about the order in which products were added to the cart. By analyzing this data, we can gain insights into customer purchasing behavior and preferences. For example, frequently added products might be popular or essential items.

#### Customer Behaviour:

- **Reordered:** This binary feature indicates whether a product was reordered by a customer. By analyzing reordered products, we can identify popular items, customer preferences, and potential opportunities for targeted marketing campaigns.

### aisles.csv

#### Aisles Identification:

- **Reordered:** This binary feature indicates whether a product was reordered by a customer. By analyzing reordered products, we can identify popular items, customer preferences, and potential opportunities for targeted marketing campaigns.
- **Aisle:** This categorical feature provides the textual description of the aisle, such as "Produce," "Dairy," or "Snacks." It offers a more human-readable representation of the aisle.

## Preprocessing

### products.csv

The dataset contains two features, **product\_name** and **product\_id**, with a maximum value of 49688. **product\_name** can be ignored while **product\_id** fulfills the same role in a more computationally efficient manner.

**aisle\_id** has a total of 134 unique values, which means compared to the total number of data, it is fairly less. So, this can be treated as a categorical variable. This indicates that multiple items are stored within the same aisle, making it suitable for categorical grouping.

Similarly, **department\_id** serves as a categorized feature. With fewer unique values than **aisle\_id**, it reflects a hierarchy where each department encompasses multiple aisles, and, by extension, each aisle houses multiple items, making **department\_id** useful for analyzing product distribution across departments.

There are no null values in any of the columns of this dataset, ensuring that the data is complete and thus making the process of preprocessing easier. The dataset also does not contain any duplicate values as well.

### orders.csv

The **order\_id** feature provides a unique ID for each transaction, even for repeat purchases of the same items.

The **user\_id** feature assigns a unique ID to each user, with repeated IDs indicating that users order multiple times.


The **order\_number** feature tracks the total orders per user, reflecting customer engagement and loyalty.

The **order\_dow** and **order\_hour\_of\_day** features capture the day and hour of each order, helping identify peak times for better staff and delivery planning.

Lastly, the **days\_since\_prior\_order** feature shows the interval between purchases, which helps in predicting restocking needs and demand cycles.

### order\_products.csv

The **order\_id** feature contains unique IDs for each order placed by users. The repeated IDs may link to the **product\_id** feature, where repetitions suggest which products are included in each specific order.



The `product_id` feature is a numerical identifier, unique to each product in the dataset, helping to distinguish individual products.

The `add_to_cart_order` feature indicates the product's priority in the cart, showing the order in which each item was added by the user.

The `reordered` feature, a categorical variable, shows whether a product was reordered, offering insights into repeat purchases.

## `aisles.csv`

The `aisle_id` feature provides unique numerical identifiers for each aisle, though it can be treated as categorical since multiple items can belong to the same aisle. Each value represents a distinct aisle, making it easier to categorize products by location.

The `aisle` feature contains the actual names of these aisles, corresponding directly to their `aisle_id` values. Each aisle has a unique name, which helps in identifying and organizing product locations more intuitively.



## Exploratory Data Analysis

### Univariate Analysis

#### Kurtosis

Kurtosis of Aisles Dataset:

	Feature	Kurtosis
0	aisle_id	-1.2

#### Aisles:

The **aisle\_id** has a kurtosis value of -1.2, indicating a platykurtic distribution. This suggests that the data is flatter than a normal distribution, with fewer extreme values and a wider spread of data points. The distribution appears to be more evenly distributed with relatively fewer large deviations from the mean.

#### Merged:

In the merged dataset, features such as **add\_to\_cart\_order** and **order\_number** exhibit high kurtosis, with **add\_to\_cart\_order** showing a sharply peaked distribution and **order\_number** showing a slightly peaked distribution. Features like **aisle\_target\_enc** and **product\_name\_target\_enc** have moderate kurtosis, indicating a distribution closer to normal with light tails. The **days\_since\_prior\_order** and **order\_hour\_of\_day** features show very flat distributions, with kurtosis values close to zero. Lastly, **product\_id**, **order\_id**, **user\_id**, **aisle\_id**, **order\_dow**, and **department\_id** all exhibit negative kurtosis, indicating platykurtic distributions that are flatter with fewer extreme values. The **reordered** feature is likely flat, but its kurtosis value is missing, potentially due to it being a binary variable.

Kurtosis of Order Data Dataset:

	Feature	Kurtosis
9	add_to_cart_order	6.085494
5	order_number	3.201481
11	aisle_target_enc	1.505797
12	product_name_target_enc	1.319419
8	days_since_prior_order	0.057007
7	order_hour_of_day	-0.003614
1	product_id	-1.139452
0	order_id	-1.197453
4	user_id	-1.200443
2	aisle_id	-1.322776
6	order_dow	-1.334320
3	department_id	-1.564641
10	reordered	-1.852469

Kurtosis of Order Products Dataset:

	Feature	Kurtosis
2	add_to_cart_order	5.643873
1	product_id	-1.140816
0	order_id	-1.199128
3	reordered	-1.866989

#### Order Products:

The **add\_to\_cart\_order** feature shows a leptokurtic distribution with a kurtosis of 5.64, indicating a highly peaked distribution with heavy tails and more extreme values compared to a normal distribution. In contrast, **product\_id**, **order\_id**, and **reordered** display platykurtic distributions, meaning they are relatively flat with fewer extreme values or outliers, suggesting a more uniform spread of data around the mean.

### Orders:

The **order\_number** feature shows a leptokurtic distribution with a kurtosis of 3.46, indicating a peaked distribution with more extreme values compared to a normal distribution. On the other hand, **order\_hour\_of\_day** has a near-normal distribution, with a very slight tendency towards a flatter distribution. Features like **days\_since\_prior\_order**, **user\_id**, **order\_id**, and **order\_dow** all exhibit platykurtic distributions, meaning they are flatter with fewer extreme values.

Kurtosis of Orders Dataset:

	Feature	Kurtosis
2	order_number	3.464992
4	order_hour_of_day	-0.009958
5	days_since_prior_order	-0.197252
1	user_id	-1.199822
0	order_id	-1.200000
3	order_dow	-1.297523

Kurtosis of Products Dataset:

	Feature	Kurtosis
2	department_id	-0.987381
0	product_id	-1.200000
1	aisle_id	-1.249013

### Products:

The department\_id, product\_id, and aisle\_id all exhibit negative kurtosis, indicating platykurtic distributions. These distributions are flatter than a normal distribution, with fewer extreme values (outliers). It suggests that the data in these features is more evenly spread out, without significant deviations from the mean.

## Skewness

### Aisles:

The **aisle\_id** has a skewness of 0.0, indicating that its distribution is symmetric. This suggests that the data is evenly spread without any significant bias toward one side, likely resembling a normal distribution.

Skewness of Aisles Dataset:

	Feature	Skewness
0	aisle_id	0.0

Skewness of Order Data Dataset:

	Feature	Skewness
9	add_to_cart_order	1.827582
5	order_number	1.745462
8	days_since_prior_order	1.035303
6	order_dow	0.182775
3	department_id	0.147519
4	user_id	0.006523
0	order_id	-0.003132
1	product_id	-0.020549
7	order_hour_of_day	-0.051270
2	aisle_id	-0.170163
10	reordered	-0.384112
12	product_name_target_enc	-0.974631
11	aisle_target_enc	-1.065976

### Merged:

The skewness analysis shows that several variables exhibit distinct distribution patterns. **Highly positively skewed** variables such as **add\_to\_cart\_order (1.83)**, **order\_number (1.75)**, and **days\_since\_prior\_order (1.04)** indicate that most values are low, with only a few higher ones. **Mildly positively skewed** variables like **order\_dow (0.18)** and **department\_id (0.15)** suggest slight preferences for specific days and departments. Variables with **near zero skew**, including **order\_id (-0.003)**, **product\_id (-0.02)**, **user\_id (0.006)**, and **order\_hour\_of\_day (-0.05)**, demonstrate balanced distributions. **Negatively skewed** variables such as

**aisle\_target\_enc** (-1.07), **product\_name\_target\_enc** (-0.97), **reordered** (-0.38), and **aisle\_id** (-0.17) show a concentration of low values, with a few exceptions.

#### Orders:

The data reveals that the distribution of several key variables shows distinct patterns. The **order\_id** has a skewness of 0.0, indicating a balanced distribution. The **user\_id** is almost symmetrical with a skewness of 0.0063, suggesting an even spread of users. The **order\_number** has a positive skew of 1.81, implying that most orders are placed by a smaller group of users who make more frequent purchases. The **order\_dow** has a skewness of 0.15, indicating a relatively even distribution of orders across days, with a slight preference for one day. The **order\_hour\_of\_day** shows a negative skew of -0.08, suggesting a tendency for orders to be placed later in the day. Finally, the **days\_since\_prior\_order** shows a positive skew of 0.98, indicating that most customers order within a short timeframe, while a smaller group of customers takes longer intervals between purchases. Overall, it is observed that order frequency is skewed, and time-based data shows only slight preferences without major deviations.

#### Skewness of Orders Dataset:

	Feature	Skewness
2	order_number	1.812533
5	days_since_prior_order	0.982260
3	order_dow	0.151306
1	user_id	0.006328
0	order_id	0.000000
4	order_hour_of_day	-0.077689

#### Skewness of Order Products Dataset:

	Feature	Skewness
2	add_to_cart_order	1.818071
0	order_id	-0.000490
1	product_id	-0.021131
3	reordered	-0.364706

#### Order Products:

The key insights reveal that the **order\_id** has a skewness of -0.0005, indicating a nearly symmetrical distribution with a very slight tilt to the left, suggesting that the order IDs are fairly evenly spread across the dataset. Similarly, the **product\_id** has a skewness of -0.0211, which is close to zero, indicating an almost symmetrical distribution with a slight tendency for products to lean toward the lower end. The

**add\_to\_cart\_order** shows a skewness of 1.82, which indicates a strong positive skew, with

most products being added early in the shopping session, though a few are added much later, creating a long right tail. The **reordered** variable has a skewness of -0.3647, suggesting a mild negative skew, where more products are ordered for the first time than reordered, although a reasonable number of repeat purchases are still observed.

### Products:

The key insights show that **product\_id** has a skewness of 0.0, indicating that its distribution is perfectly symmetrical and the product IDs are evenly spread across the dataset. **Aisle\_id** has a skewness of -0.066, suggesting a slight negative skew, with a few aisles having more products, but the overall distribution remains balanced.

**Department\_id** exhibits a skewness of -0.31, indicating a mild negative skew, where most products are concentrated in a few departments, with fewer products found in the others. It can be inferred that **product\_id** is perfectly balanced, while **aisle\_id** and **department\_id** show a slight negative skew, meaning certain aisles and departments have a higher concentration of products, though the skew is not extreme.

### Skewness of Products Dataset:

	Feature	Skewness
0	product_id	0.000000
1	aisle_id	-0.066273
2	department_id	-0.309852

### Plots

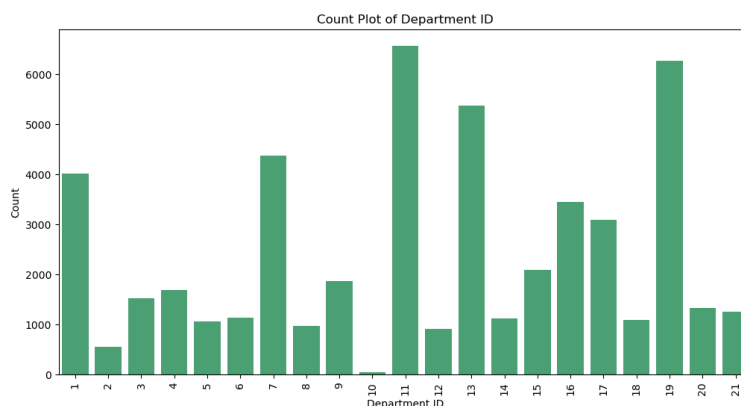


Fig 1: Department ID

The bar plot of Department ID shows the distribution of products across the different departments in the store.

From the graph, it is inferred that departments 11, 13, and 19 have the greatest number of products stored. Similarly, departments 2, 5, 8, and 10 have fairly fewer products stored, with department 10 having the least number of items (supposedly more than 20).

From the graph, it could be inferred that the departments with higher number of items stored, customers might frequent these departments more and the departments with less items less.

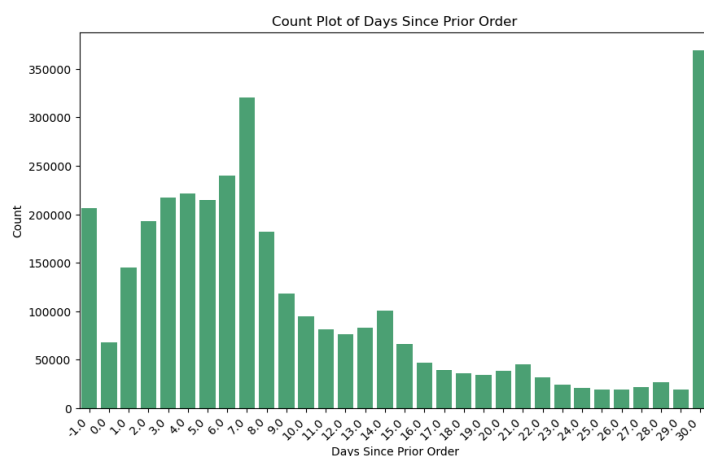


Fig 2: Days Since Prior Order

The bar plot of days since prior order suggests that customers tend to buy items fairly regularly with a maximum gap of 9 days since prior order, suggesting daily or weekly requirement.

Customers who buy after a month are high, suggesting items that are bought could be of monthly requirement.

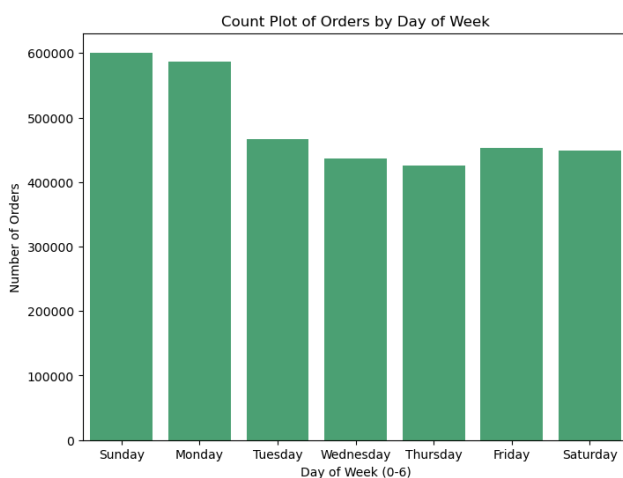


Fig 3 Day of Week

Sunday and Monday have the highest number of products ordered by customers. This trend indicates that users primarily place orders when they are not working or need items at the start of the week. This pattern may reflect users' tendency to restock household items or groceries after the weekend, preparing for the week ahead.

In contrast, there are fewer orders placed in the middle of the week, specifically on Wednesday and Thursday. This dip suggests that users are less likely to place orders when they are occupied with work, which could reduce their focus on shopping.

Overall, the distribution of orders across the week is nearly balanced, highlighting continuous engagement by users and a high demand for products. This steady activity implies that users consistently interact with the platform, maintaining a stable order flow.

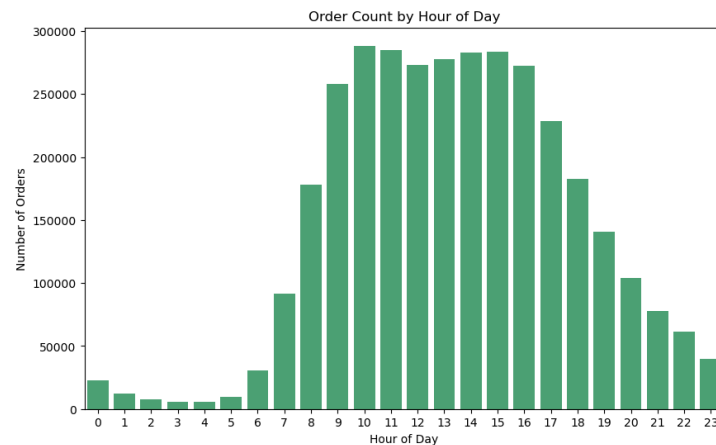


Fig 4: Hour of Day

Most users are actively engaging with the platform and purchasing products from 9 am to 4 pm. This peak period suggests that a majority of users place orders during standard daytime hours, which may coincide with breaks in their daily routines or designated times for household shopping. The increased volume of orders during these hours indicates a significant demand that requires efficient processing and handling to ensure timely order fulfilment.

Conversely, there is a noticeable drop-in order activity from midnight to 6 am, with only a few users placing orders during these early morning hours. The orders made during this period could be driven by specific, urgent needs or unplanned requirements that users need to address immediately. This rare, off-peak ordering behavior may hint at unique scenarios, such as medical emergencies or last-minute needs, where users are motivated to place orders outside regular shopping times.

To efficiently manage the influx of orders, particularly during peak hours, it's crucial to focus on employee engagement from 9 am to 4 pm. By increasing staffing and support resources during this high-traffic period, the platform can better handle the rush of orders, reduce wait times, and improve overall user satisfaction. Ensuring that employees are well-prepared and available during these hours will help streamline operations, meet user expectations, and maintain a positive shopping experience.

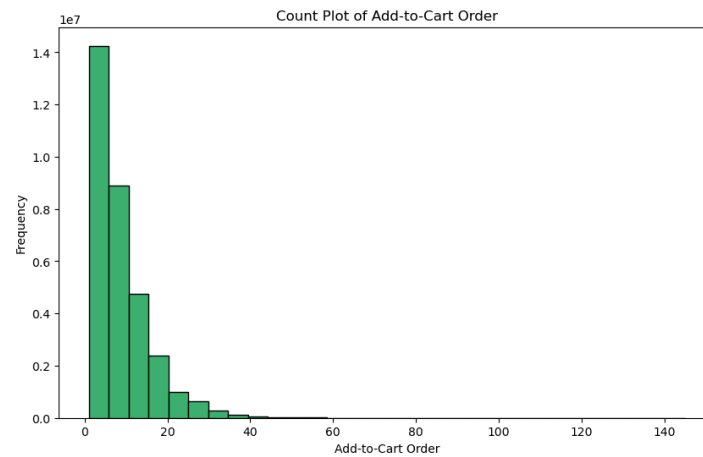


Fig 5: Add-to-Cart Order

From the bar graph of Add to Cart Order, it can be inferred that customers prioritize the first item they take followed by the items after. This helps to realise that customers always take items that they need most at first, followed by items that is not of their top priority.

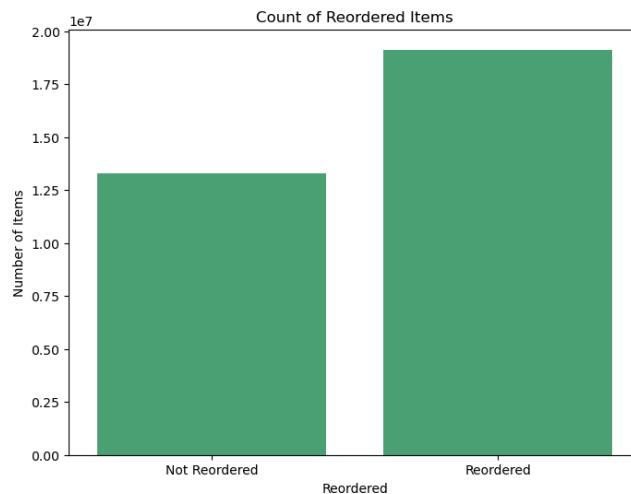


Fig 6: Reordered

The categorical bar graph of reordered items suggests that customers tend to reorder their preferred items more. The graph is more or less balanced suggesting that the items that are not reordered could be a one-time purchase or customers don't need them after the first purchase. This helps in understanding which items to restock on a regular basis.

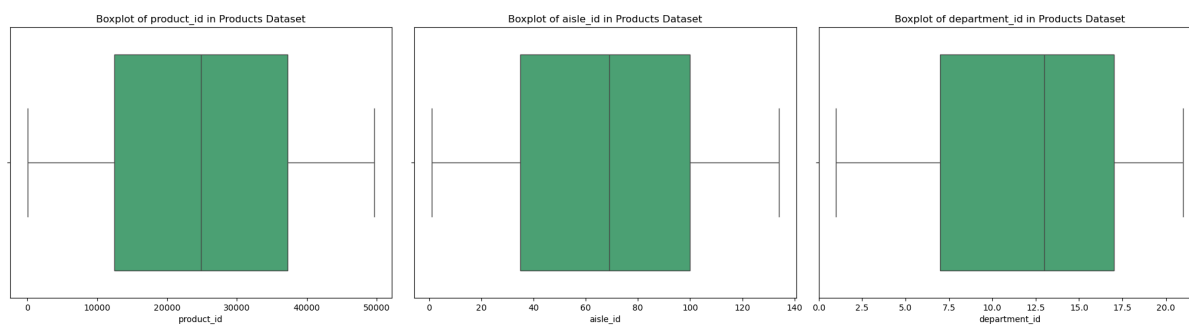


Fig 7: Boxplot on product.csv Dataset

### Product ID:

The boxplot for product ID shows a relatively uniform distribution, with no significant outliers. This suggests that the product IDs are likely assigned sequentially, without any major gaps or clusters.

### Aisle ID:

The boxplot for aisle ID reveals a similar pattern. The data points are evenly distributed, with no noticeable outliers. This indicates that aisles are likely numbered sequentially, without any significant empty spaces.

### Department ID:

The boxplot for department ID also exhibits a uniform distribution. There are no significant outliers, suggesting that departments are likely numbered sequentially without major gaps.

## Conclusion

Based on the boxplots, it can be concluded that the product, aisle, and department IDs in the Products dataset are likely assigned sequentially, without any major gaps or clusters. This suggests a well-organized and systematic approach to product categorization and management within the dataset.



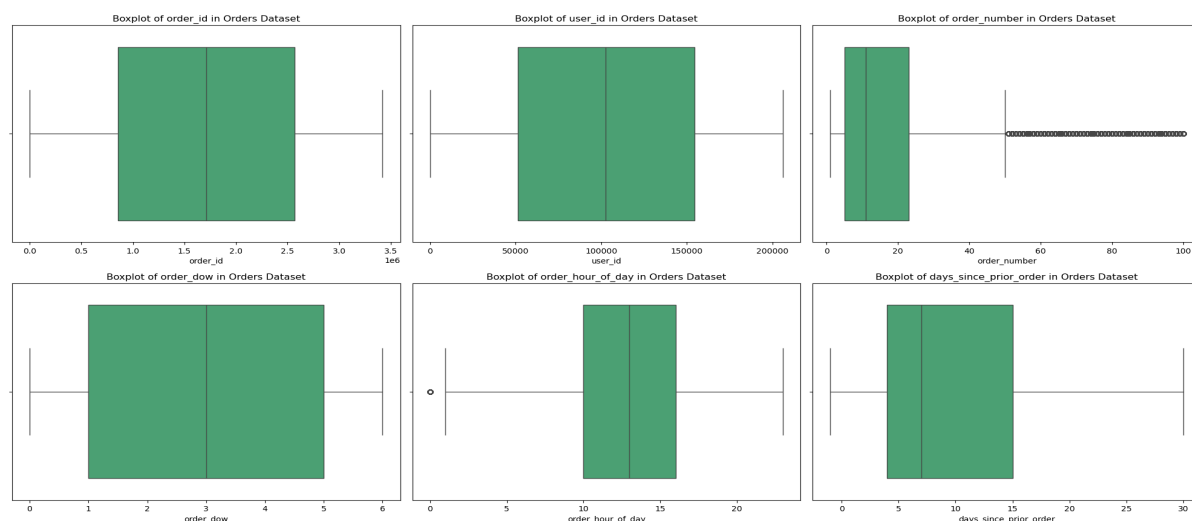


Fig 8: Boxplot on orders.csv Dataset

**Order ID and User ID:** Both variables exhibit uniform distributions, indicating sequential assignment. This suggests a well-structured system for tracking orders and users.

**Order Number:** The right-skewed distribution with outliers suggests that a small number of users place a significantly larger number of orders compared to the average user. This could be due to factors like repeat customers or bulk purchases.

**Order Day of Week and Order Hour of Day:** The uniform distribution for the day of the week indicates consistent order placement throughout the week. The peaked distribution for the hour of the day suggests that most orders are placed during the daytime, possibly due to factors like working hours or daily routines.

**Days Since Prior Order:** The right-skewed distribution with outliers indicates that while most customers have a consistent ordering frequency, there are a few customers who place orders very infrequently. These infrequent customers are represented by the outliers.

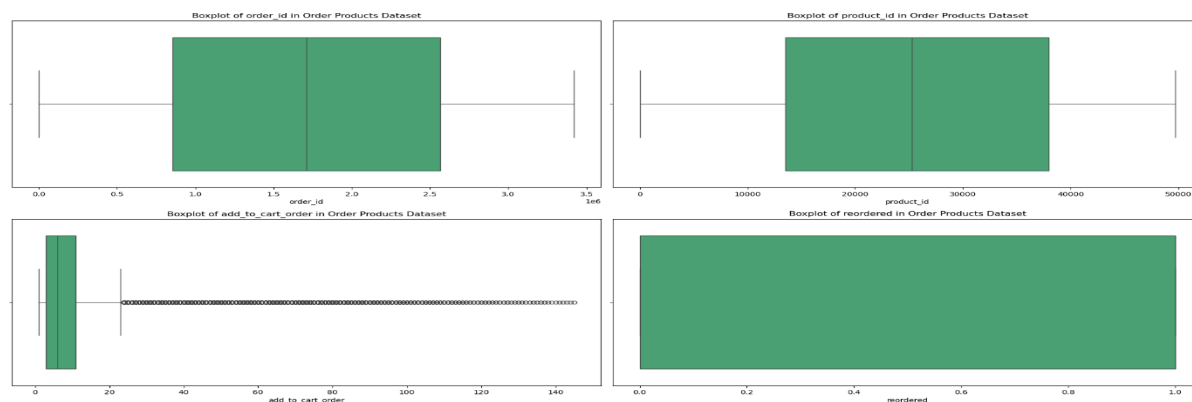


Fig 9: Boxplot on order\_products.csv Dataset

**Order ID:**

- The boxplot shows a uniform distribution with no significant outliers. This suggests that order IDs are likely assigned sequentially.

**Product ID:**

- Similar to order IDs, the boxplot for product IDs also shows a uniform distribution with no outliers. This indicates that product IDs are likely assigned sequentially.

**Add-to-Cart Order:**

- The boxplot reveals a right-skewed distribution with a long tail. This suggests that most products are added to carts in smaller quantities, while a few products are added in much larger quantities.

**Reordered:**

- The boxplot shows a bimodal distribution with two distinct peaks. This suggests that a significant number of products are either reordered frequently or not reordered at all.

**Conclusion**

Based on the boxplots, we can draw the following conclusions:

- Order IDs and product IDs are likely assigned sequentially.
- Most products are added to carts in small quantities, with a few exceptions of larger quantities.
- Products are either frequently reordered or not reordered at all.

## Bivariate Analysis

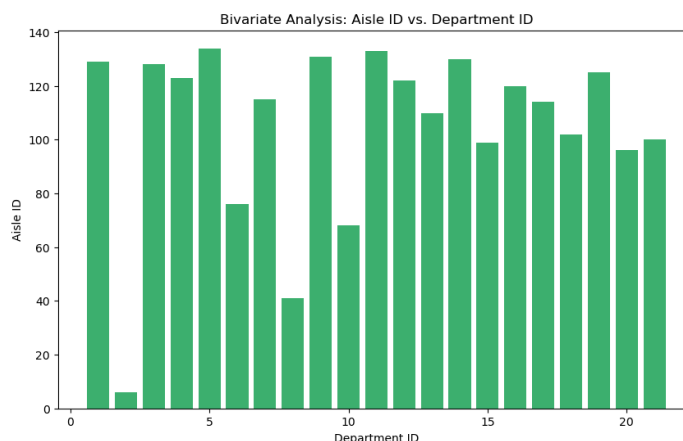


Fig 10: Barplot of Aisle ID vs Department ID

The provided bar chart visualizes the relationship between aisle IDs and department IDs. A wide range of aisle IDs exists within each department, indicating a diverse product assortment within each department.

However, the distribution of aisle IDs across departments appears to be uneven. Some departments have a larger number of aisles compared to others. This suggests that certain departments may cater to a wider range of products or have a more complex product assortment.

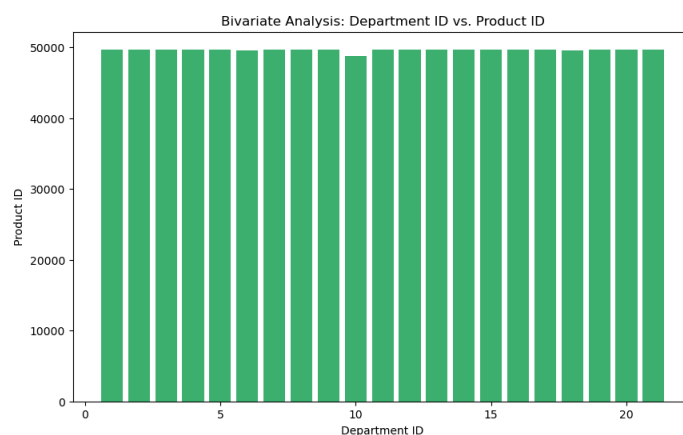


Fig 11: Department ID vs Product ID

**Uniform Product Allocation:** Similar counts of products are allocated across departments, showing balanced product distribution.

**Optimized Inventory Management:** Departments are well-organized, with no single department overly stocked or sparse in product variety.

Improved Shopping Experience: Even distribution across departments allows customers to find diverse products without needing to navigate specific departments for variety.

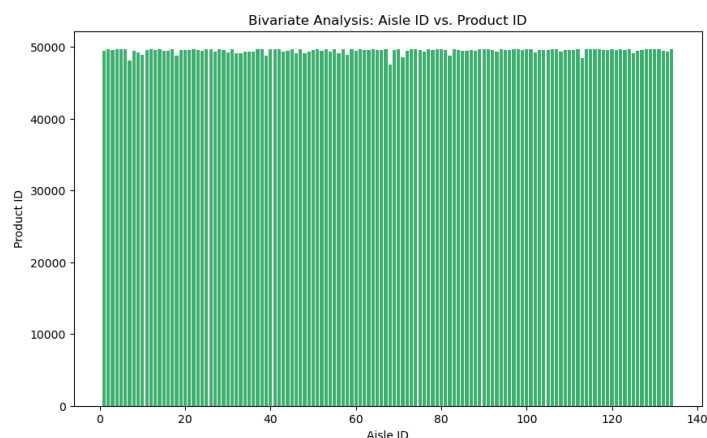


Fig 12: Aisle ID vs Product ID

The bar plot of aisle\_id against product\_id shows a nearly uniform distribution of products across aisles, indicating that most aisles stock a similar number of products. This balanced distribution suggests efficient inventory management, with no single aisle being overloaded or understocked. It also implies diverse product availability across aisles, potentially enhancing customer convenience by spreading product options evenly throughout the store.

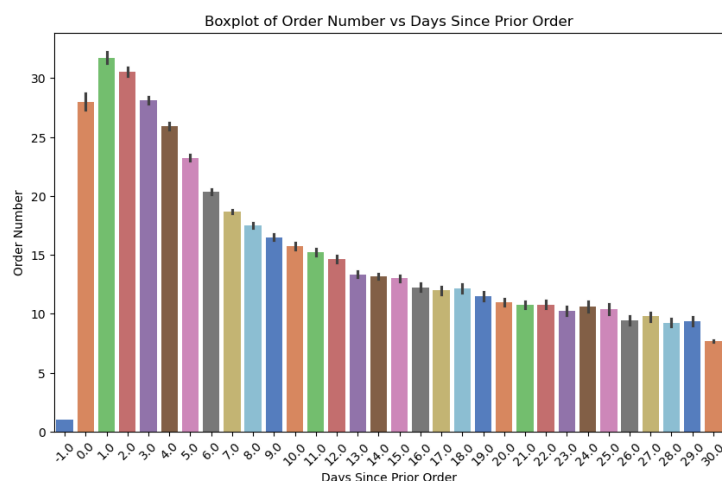


Fig 13: Order Number vs Days Since Prior Order

The boxplot illustrates the relationship between the number of orders and the days since the prior order. The plot reveals a clear downward trend, indicating that as the number of

days since the last order increases, the number of orders placed decreases. This suggests that customer behavior is influenced by recency, with customers more likely to place orders shortly after their previous purchase.

The distribution of order numbers within each day bin varies, with some bins showing a wider spread than others. This suggests that customer behavior may also be influenced by other factors, such as product availability, promotions, or seasonal trends.

Overall, the boxplot provides valuable insights into customer purchasing patterns and can be used to inform marketing strategies, inventory management, and customer retention efforts. By understanding the relationship between time since last purchase and order frequency, businesses can target specific customer segments with personalized offers and promotions to encourage repeat purchases.

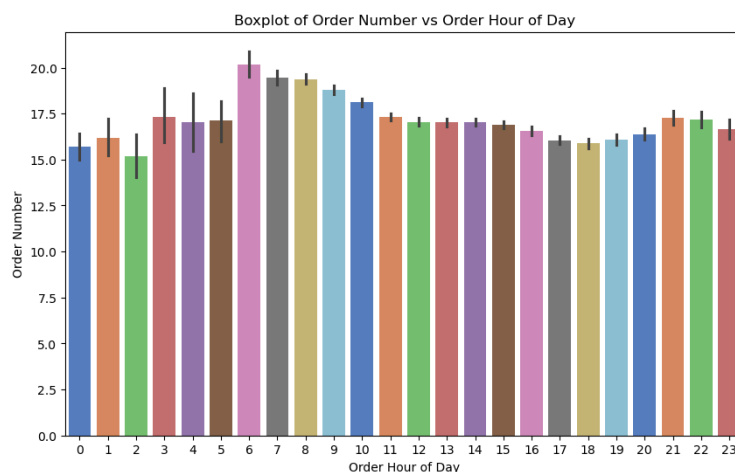


Fig 14: Order Number vs Order Hour of Day

The boxplot illustrates the relationship between the number of orders and the hour of the day. The plot shows a clear peak around the middle of the day, indicating that most orders are placed during this time. This suggests that customer behavior is influenced by daily routines and work schedules.

The distribution of order numbers within each hour bin varies, with some hours showing a wider spread than others. This suggests that other factors, such as promotions, sales, or specific events, may also influence ordering patterns.

Overall, the boxplot provides valuable insights into customer purchasing behavior and can be used to optimize marketing strategies, inventory management, and customer service operations. By understanding the peak ordering hours, businesses can allocate resources effectively and ensure timely order fulfillment.



Fig 15: Order Number vs Day of Week

The boxplot illustrates the relationship between the number of orders and the day of the week. The plot shows a slight variation in the number of orders across different days, with some days having a slightly higher average number of orders than others. This suggests that customer behavior may be influenced by factors such as work schedules, weekends, or specific events.

However, the overall trend indicates that the number of orders is relatively consistent across all days of the week. This suggests that customers are placing orders regularly throughout the week.

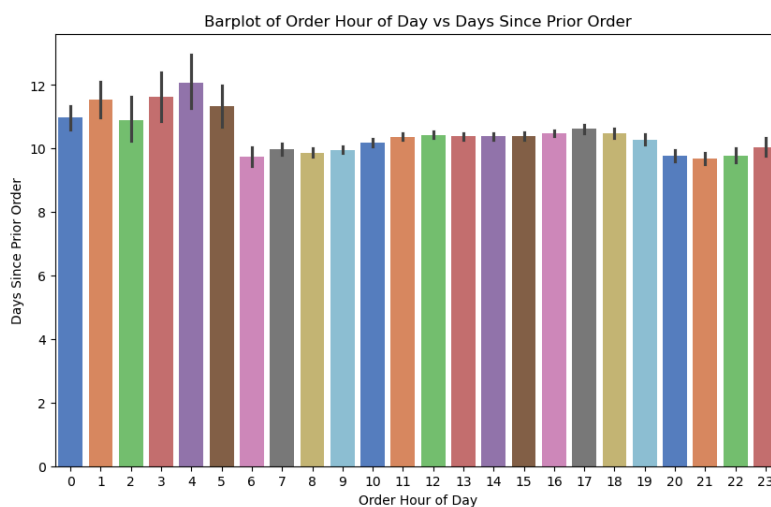


Fig 16: Days Since Prior Order vs Order Hour of Day

The bar plot illustrates the relationship between the order hour of the day and the days since the prior order. The plot shows a slight variation in the average days since the prior order across different hours of the day.

While there is no clear trend, the plot suggests that there might be subtle differences in customer behavior at different times of the day. For example, customers who place orders during certain hours might tend to order more frequently than those who order at other times.

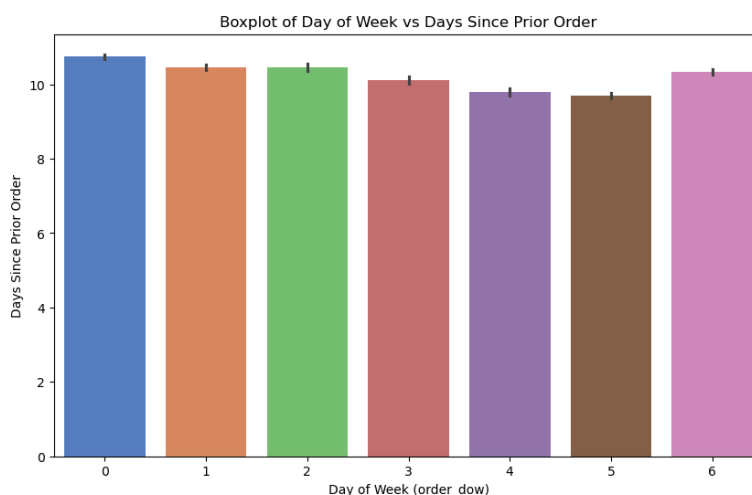


Fig 17: Days Since Prior Order vs Day of Week

The boxplot illustrates the relationship between the day of the week and the days since the prior order. The plot shows a relatively consistent pattern across different days of the week, indicating that customer purchasing behavior is not significantly influenced by the specific day.

While there are slight variations in the average days since the prior order, the overall trend suggests that customers tend to place orders with a similar frequency regardless of the day of the week.

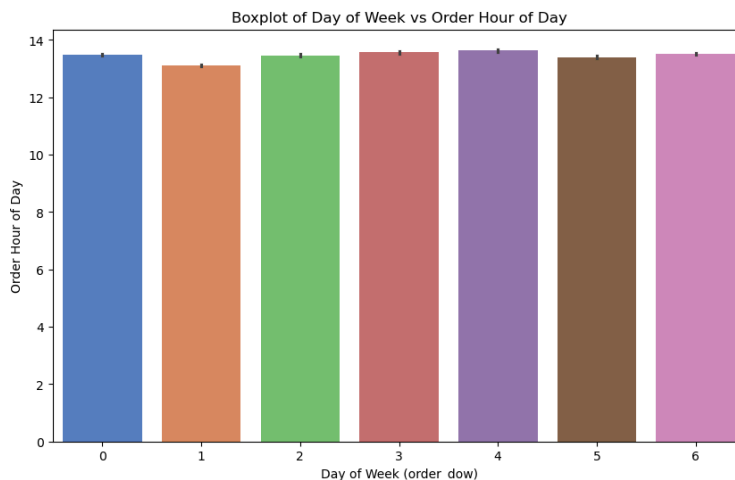


Fig 18: Order Hour of Day vs Day of Week

The boxplot illustrates the relationship between the day of the week and the order hour of the day. The plot shows a relatively consistent pattern across different days of the week, indicating that customer ordering behavior is not significantly influenced by the specific day.

While there are slight variations in the average order hour across different days, the overall trend suggests that customers tend to place orders at similar times regardless of the day of the week. This information can be valuable for businesses in planning staffing schedules, inventory management, and customer service operations.



## Multivariate Analysis

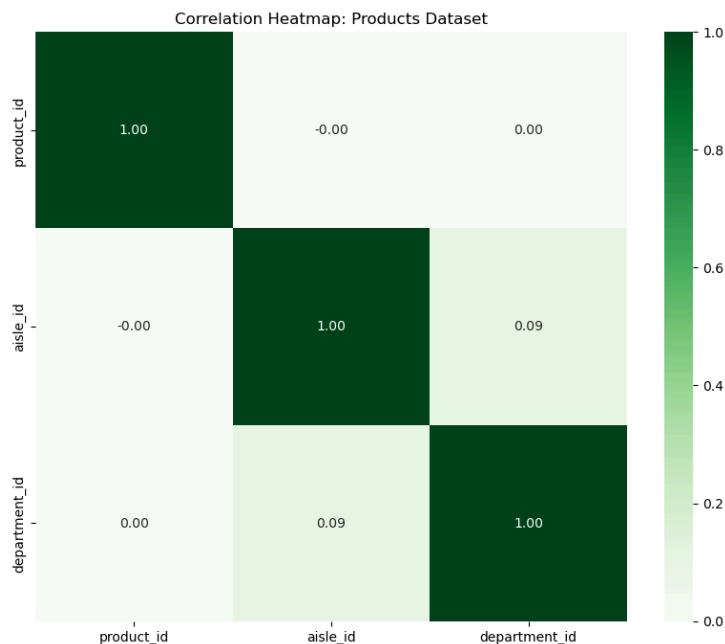


Fig 19:

### 1.Weak Correlations:

- **product\_id** and **aisle\_id** have a very weak negative correlation (-0.00). This suggests that there is no significant linear relationship between these two variables.
- **product\_id** and **department\_id** have a very weak positive correlation (0.00). This indicates a negligible linear relationship between these variables.
- **aisle\_id** and **department\_id** have a slightly stronger positive correlation (0.09). This suggests a weak positive linear relationship, meaning that as the aisle ID increases, the department ID tends to increase slightly.

### 2.Strong Positive Correlations:

- The diagonal elements, which represent the correlation of a variable with itself, are all 1.00. This is expected as a variable is perfectly correlated with itself.

### Overall:

The heatmap indicates that there are no strong correlations between the variables in the Products dataset. This suggests that these variables are relatively independent of each other and may not have significant linear relationships.

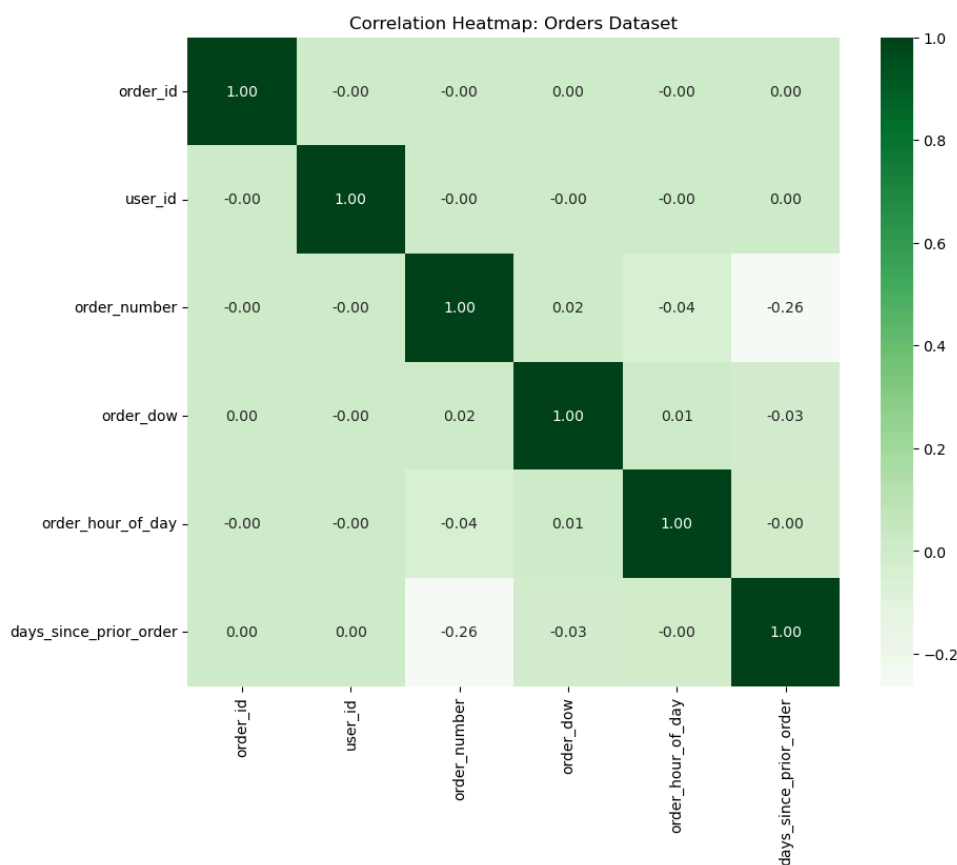


Fig 20:

### 1. Weak Correlations:

- **order\_id** has weak correlations with all other variables, indicating no significant linear relationship.
- **user\_id** has weak correlations with all other variables, suggesting no significant linear relationship between user ID and other factors like order number, day of week, or order hour.
- **order\_number** has a weak negative correlation (-0.26) with **days\_since\_prior\_order**. This indicates that as the number of days since the prior order increases, the order number tends to decrease slightly. This could be due to factors like customer churn or reduced purchasing frequency over time.

### 2. Moderate Negative Correlation:

- **order\_number** has a moderate negative correlation (-0.26) with **days\_since\_prior\_order**. This indicates a moderate negative linear relationship,

meaning that as the number of days since the prior order increases, the order number tends to decrease.

### 3. No Strong Correlations:

- Most of the other correlations are very weak or close to zero, indicating no significant linear relationships between the variables.

### Overall:

The heatmap suggests that there are no strong correlations between the variables in the Orders dataset, except for the moderate negative correlation between order number and days since prior order. This indicates that these variables are relatively independent of each other and may not have significant linear relationships.

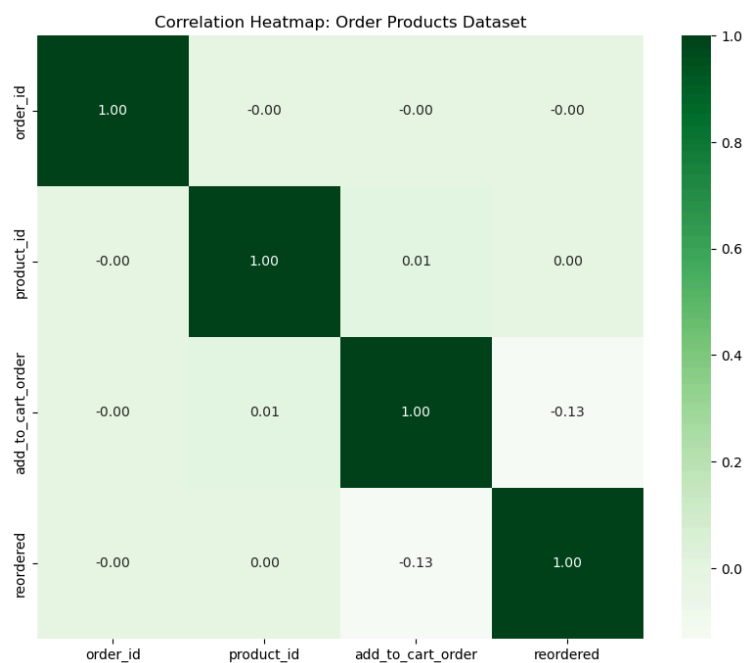


Fig 21:

### 1. Weak Correlations:

- **order\_id** has weak correlations with all other variables, indicating no significant linear relationship.
- **product\_id** has a very weak positive correlation (0.01) with **add\_to\_cart\_order**, suggesting a negligible linear relationship.

- **add\_to\_cart\_order** has a weak negative correlation (-0.13) with **reordered**. This indicates that as the position of the product in the cart increases, the likelihood of it being reordered decreases slightly.

## 2. No Strong Correlations:

- Most of the other correlations are very weak or close to zero, indicating no significant linear relationships between the variables.

## Overall:

The heatmap suggests that there are no strong correlations between the variables in the Order Products dataset, except for the weak negative correlation between **add\_to\_cart\_order** and **reordered**. This indicates that these variables are relatively independent of each other and may not have significant linear relationships.

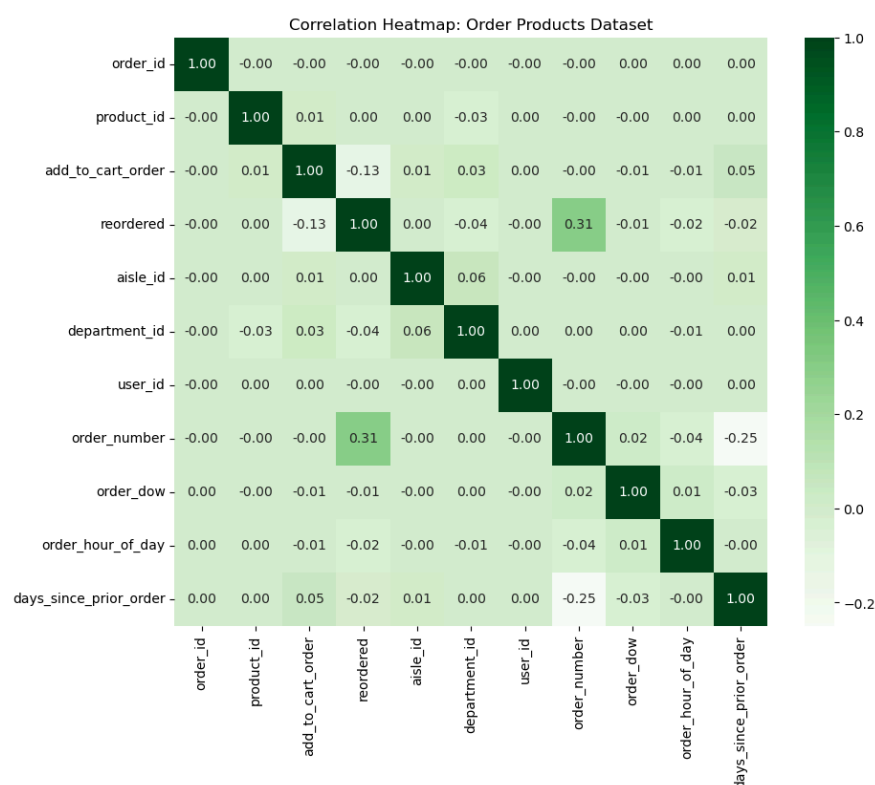


Fig 22:

## 1. Weak Correlations:

- **order\_id** has weak correlations with all other variables, indicating no significant linear relationship.

- **product\_id** has very weak correlations with other variables, suggesting no significant linear relationship.
- **add\_to\_cart\_order** has a weak negative correlation (-0.13) with **reordered**. This indicates that as the position of the product in the cart increases, the likelihood of it being reordered decreases slightly.

## 2. Moderate Negative Correlation:

- **order\_number** has a moderate negative correlation (-0.25) with **days\_since\_prior\_order**. This indicates a moderate negative linear relationship, meaning that as the number of days since the prior order increases, the order number tends to decrease.

## 3. No Strong Correlations:

- Most of the other correlations are very weak or close to zero, indicating no significant linear relationships between the variables.

## Overall:

The heatmap suggests that there are no strong correlations between the variables in the Order Products dataset, except for the moderate negative correlation between order number and days since prior order. This indicates that these variables are relatively independent of each other and may not have significant linear relationships.

## Data Encoding

### ENCODING - 1: Applying Target Encoding of aisle, department, product\_name

What is Target Encoding:

Target encoding replaces categorical values with the mean of the target variable (e.g., reordered) for each category.

Features Being Encoded:

The features **aisle**, **department**, and **product\_name** are encoded using the mean of the reordered column.

Why This Encoding is Needed:

It captures the relationship between categorical features and the target variable, improving model performance for numerical algorithms like Logistic Regression.

aisle_target_enc	department_target_enc	product_name_target_enc
0.547057	0.574195	0.585799
0.590163	0.574195	0.629367
0.671338	0.628133	0.632911
0.46946	0.346935	0.4375
0.493279	0.541577	0.509413
0.671338	0.628133	0.47093
0.706434	0.669969	0.736155
0.718275	0.649793	0.777074
0.718275	0.649793	0.843169
0.68665	0.669969	0.681264

## ENCODING - 2: Applying One-hot Encoding on day\_of\_week

What is One-Hot Encoding:

One-hot encoding converts categorical values into binary columns, where each unique category becomes a separate column with values as 0 or 1.

Features Being Encoded:

The feature **order\_dow** (day of the week) is encoded into new binary columns: **dow\_0**, **dow\_1**, ..., **dow\_6**.

Why This Encoding is Needed:

It allows the model to treat each category as independent and avoids assuming any ordinal relationship, which is essential for algorithms that require numerical input.

dow_0	dow_1	dow_2	dow_3	dow_4	dow_5	dow_6
0	1	0	0	0	0	0
0	0	0	0	1	0	0
0	0	0	0	0	1	0
0	0	0	0	0	1	0
0	0	1	0	0	0	0
1	0	0	0	0	0	0
0	0	0	0	0	1	0
0	0	0	0	0	0	1
0	0	0	0	0	1	0
0	0	0	0	0	0	1

## ENCODING - 3: Applying Cyclic Encoding and Performing Sin-Cosine Transformation on `order_hour_of_day`

### What is Cyclic Encoding:

Cyclic encoding transforms cyclic features (e.g., hours of the day) into sine and cosine components to preserve their cyclical nature.

### Features Being Encoded:

The feature **`order_hour_of_day`** is encoded into two new columns: **`order_hour_sin`** and **`order_hour_cos`**.

### Why This Encoding is Needed:

It helps the model understand the cyclical relationship (e.g., 23:00 is close to 00:00), which regular numerical encoding cannot capture.

<code>order_hour_sin</code>	<code>order_hour_cos</code>
2.58819E-01	-0.965926
8.66025E-01	-0.5
7.07107E-01	-0.707107
1.22465E-16	-1
-9.65926E-01	-0.258819
1.22465E-16	-1
7.07107E-01	-0.707107
-7.07107E-01	0.707107
-8.66025E-01	-0.5
5.00000E-01	-0.866025



## ENCODING - 4: Applying Ordinal Encoding on add\_to\_card\_order

What is Ordinal Encoding:

Ordinal encoding converts categorical or ordered data into integers while preserving their rank or order.

Features Being Encoded:

The feature **add\_to\_cart\_order** is encoded into a new column **add\_to\_cart\_order\_encoded** with integer values representing its order.

Why This Encoding is Needed:

It converts ordered data into a numerical format suitable for machine learning models while retaining the inherent order information.

order_id	product_id	add_to_cart_order	reordered	add_to_cart_order_encoded
2722718	8619	9	0	8
2089674	13870	10	0	9
3024155	2029	1	0	0
2890872	16062	2	0	1
1798802	14335	3	0	2
1688514	40556	24	1	23
430688	14999	2	1	1
2022104	21137	5	1	4
456648	24852	1	1	0
579936	30442	2	1	1

## ENCODING - 5: Apply Binning on day\_since\_prior\_order

What is Binning:

Binning categorizes continuous values into discrete intervals (or bins), assigning each value to a specific category.

Features Being Encoded:

The feature **days\_since\_prior\_order** is binned into categories: 0-7, 8-15, 16-23, 24-31, and Unknown.

Why This Encoding is Needed:

It simplifies continuous data, making it easier to interpret patterns and relationships, especially for models or analysis requiring categorical inputs.

order_id	product_id	days_since_prior_order_binned
2722718	8619	8-15
2089674	13870	24-31
3024155	2029	0-7
2890872	16062	0-7
1798802	14335	Unknown
1688514	40556	0-7
430688	14999	24-31
2022104	21137	0-7
456648	24852	0-7
579936	30442	8-15

## FEATURE ENGINEERING

### Feature I -- `average_days_between_purchases`:

Average time between purchases by each product by each user. This gives insights into the typical frequency of repurchases of a specific product

#### Definition of the Feature:

`average_days_between_purchases` represents the average time interval (in days) between consecutive purchases of a specific product by a user.

#### Role of the Feature:

It helps identify user buying behavior and product repurchase frequency, which is crucial for predicting reorder patterns and understanding product popularity.

order_id	product_id	add_to_cart_order_encoded	days_since_prior_order_binned	average_days_between_purchases
2722718	8619	8	8-15	15
2089674	13870	9	24-31	29
3024155	2029	0	0-7	4
2890872	16062	1	0-7	2
1798802	14335	2	Unknown	7.166667
1688514	40556	23	0-7	5.333333
430688	14999	1	24-31	11
2022104	21137	4	0-7	9.666667
456648	24852	0	0-7	6.375
579936	30442	1	8-15	21

## Feature II -- Product\_purchase\_frequency:

Count the total number of times each product has been purchased by a particular user.

### Definition of the Feature:

*product\_purchase\_frequency* represents the total count of how many times a specific product has been purchased by a user.

### Role of the Feature:

It highlights user preferences for specific products, aiding in understanding product loyalty and predicting future purchases.

order_id	product_id	product_purchase_frequency	total_purchases
2722718	8619	1	1
2089674	13870	1	1
3024155	2029	1	1
2890872	16062	1	1
1798802	14335	6	6
1688514	40556	3	3
430688	14999	4	4
2022104	21137	3	3
456648	24852	8	8
579936	30442	2	2

### Feature III -- total\_purchases

Definition of the Feature:

*total\_purchases* represents the total number of purchases made for each user-product combination.

Role of the Feature:

It tracks the overall purchase behavior for a user-product pair, providing insights into user engagement with specific products.

### Feature IV – interval\_std\_dev

Definition of the Feature:

*interval\_std\_dev* represents the standard deviation of the time intervals between consecutive purchases of a product by a user.

Role of the Feature:

It measures the variability in purchase frequency, helping identify inconsistent purchasing patterns or seasonal buying behavior.

order_id	product_id	product_purchase_frequency	total_purchases	interval_std_dev
2722718	8619	1	1	NaN
2089674	13870	1	1	NaN
3024155	2029	1	1	NaN
2890872	16062	1	1	NaN
1798802	14335	6	6	4.445972
1688514	40556	3	3	1.527525
430688	14999	4	4	11.518102
2022104	21137	3	3	4.932883
456648	24852	8	8	3.159453
579936	30442	2	2	14.142136

### Feature V – Product\_reorder\_rate:

Reorder rate for each product by dividing the number of times a product has been reordered by the total number of orders of product

#### Definition of the Feature:

*product\_reorder\_rate* is calculated by dividing the number of times a product has been reordered by its total number of orders.

#### Role of the Feature:

It measures how frequently a product is reordered, providing insights into product popularity and customer retention tendencies.

order_id	product_id	total_purchases	interval_std_dev	product_reorder_rate
2722718	8619	1	NaN	0.585799
2089674	13870	1	NaN	0.629367
3024155	2029	1	NaN	0.632911
2890872	16062	1	NaN	0.4375
1798802	14335	6	4.445972	0.509413
1688514	40556	3	1.527525	0.47093
430688	14999	4	11.518102	0.736155
2022104	21137	3	4.932883	0.777074
456648	24852	8	3.159453	0.843169
579936	30442	2	14.142136	0.681264

## Feature VI – Users\_general\_reorder\_rate:

The ratio of reordered items to total items of each user which captures the user general tendency to reorder products.

### Definition of the Feature:

*users\_general\_reorder\_rate* is the ratio of items reordered to the total items purchased by a user, reflecting their overall tendency to reorder

### Role of the Feature:

It helps in understanding user loyalty and predicting the likelihood of users reordering products in future purchases.

order_id	product_id	product_reorder_rate	users_general_reorder_rate
2722718	8619	0.585799	0.818182
2089674	13870	0.629367	0.47619
3024155	2029	0.632911	0.176471
2890872	16062	0.4375	0.27907
1798802	14335	0.509413	0.7125
1688514	40556	0.47093	0.78961
430688	14999	0.736155	0.675325
2022104	21137	0.777074	0.62037
456648	24852	0.843169	0.684932
579936	30442	0.681264	0.222222

## Feature VII – Avg\_add\_to\_cart\_order:

Avg. position of each product in the cart when it is purchased.

### Definition of the Feature:

*avg\_add\_to\_cart\_order* represents the average position of a product in the shopping cart across all its purchases.

### Role of the Feature:

It provides insights into user preferences, indicating whether a product is typically added early or late in the shopping process.

order_id	product_id	users_general_reorder_rate	avg_add_to_cart_order
2722718	8619	0.818182	10.91716
2089674	13870	0.47619	9.489091
3024155	2029	0.176471	9.468354
2890872	16062	0.27907	13.520833
1798802	14335	0.7125	8.58804
1688514	40556	0.78961	8.473837
430688	14999	0.675325	6.785266
2022104	21137	0.62037	7.251493
456648	24852	0.684932	4.889006
579936	30442	0.222222	7.992325



## Correlation Matrix - HeatMap

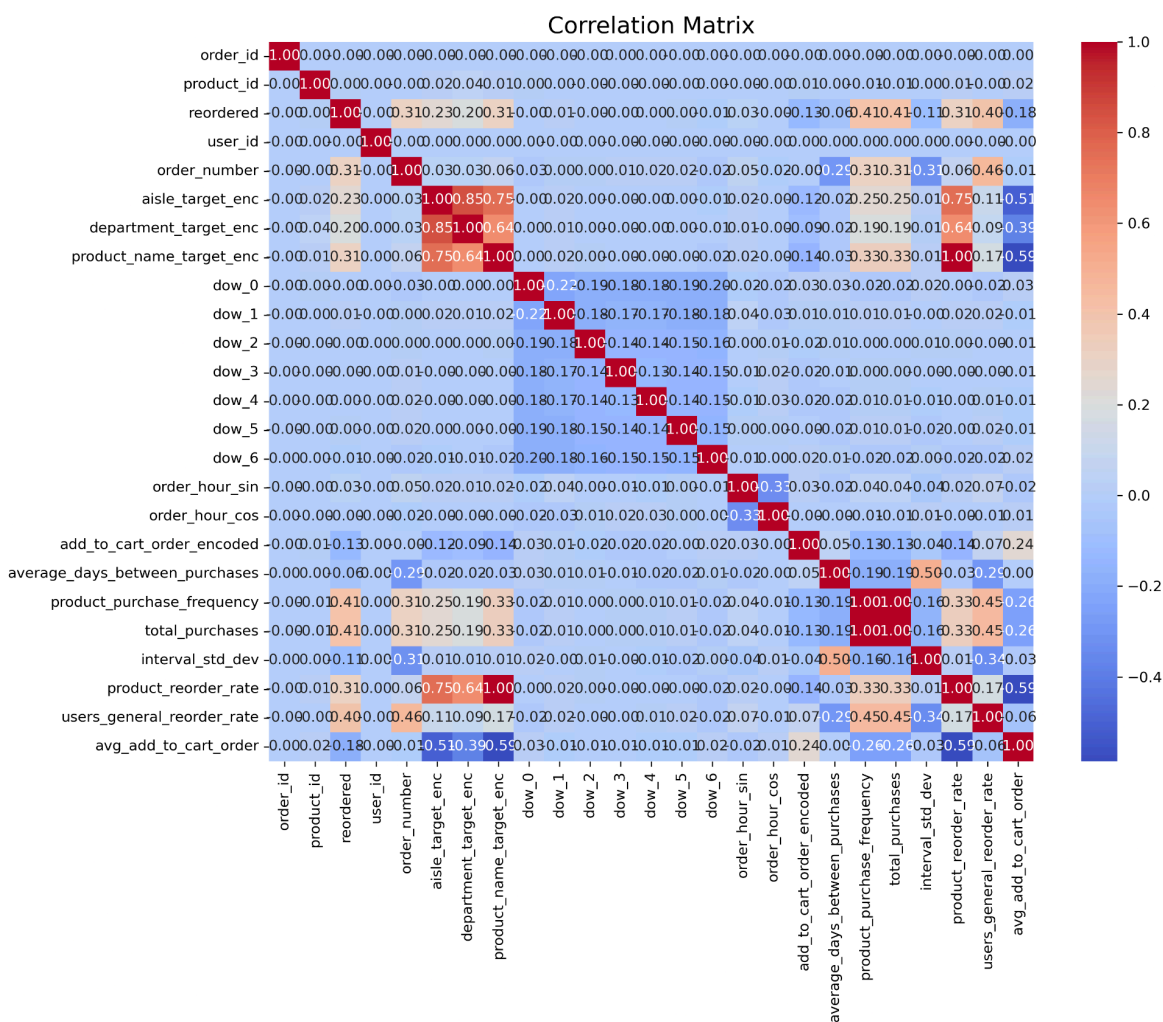


Fig 23:

Insights:

Strongly Correlated Features:

- ❖ aisle\_target\_enc, department\_target\_enc, and product\_name\_target\_enc are highly correlated with product\_reorder\_rate, making them key predictors of reorder behavior.

- ❖ product\_purchase\_frequency is strongly correlated with total\_purchases, indicating both represent user-product purchase activity.

#### Negative Correlations:

- ❖ avg\_add\_to\_cart\_order has a negative correlation with product\_reorder\_rate and users\_general\_reorder\_rate, suggesting that earlier positions in the cart are linked to higher reorder likelihood.

#### Weak Correlations:

- ❖ Day-of-week features (dow\_0, dow\_1, etc.) show minimal correlation with other features, implying limited influence on purchase and reorder patterns.

#### Independent Features:

- ❖ Identifier columns like order\_id, product\_id, and user\_id are uncorrelated with other features, as they serve only as unique identifiers.

#### Temporal Insights:

- ❖ order\_hour\_sin and order\_hour\_cos have weak correlations with reorder metrics, indicating time-of-day has a slight influence on reorder behavior.

#### Reorder Behavior:

- ❖ Features like users\_general\_reorder\_rate and product\_reorder\_rate show significant relationships with metrics like frequency and interval, making them central to understanding reorder tendencies.

## Market Basket Analysis & Product Bundling

### DATA LOADING AND PREPARATION

#### Data Collection:

Datasets `order_products.csv`, `products.csv`, and `orders.csv` are utilized to analyze customer purchasing behavior.

#### Data Cleaning:

The `eval_set` was removed because it contains prior data, including both training and testing sets, with most of the data being from the prior period, which is not relevant for this analysis.

#### Product Frequency Calculation:

Each product's order frequency is calculated, and the top 100 most frequently ordered products are identified. Creation of a Frequent Products List: A list of the top 100 product IDs (`freq_products`) is generated to focus further analysis on popular items.

### DATA FILTERING AND TRANSFORMATION

#### Selecting Frequent Products:

The dataset is filtered to include only orders with the top 100 frequent products, narrowing the focus to high-interest items.

#### Merging Product Names:

Product names are merged with the dataset for clarity in identifying items.

#### Data Reshaping for Basket Analysis:

The data is restructured into a matrix format where each row represents an order and each column represents a product, suitable for basket analysis.

#### Binary Encoding:

Binary encoding is applied to mark the presence of each product in an order, preparing the data for analysis by indicating whether a product was purchased (1) or not (0).

## MARKET BASKET ANALYSIS WITH APRIORI ALGORITHM

### FREQUENT ITEM SET

#### Identification:

The Apriori algorithm is applied with a minimum support threshold of 0.005, capturing item combinations that appear in at least 0.5% of orders, ensuring focus on significant item sets.

#### Association Rule Generation:

Association rules are generated using lift as the primary metric, with a minimum threshold of 1, to highlight strong associations between products.

#### Apriori instead of FP-Growth:

Apriori provides control over minimum support at each stage, allowing analysts to adjust thresholds easily to find frequent itemsets at specific support levels.

FP-Growth often requires a single, predetermined support threshold so that's why even if FP-Growth is convenient for larger dataset due to less tuning but due to adjustment of support thresholds it can be suggested to use Apriori Algorithm.

#### Summary of Itemsets and their Support Values:

support	itemsets
0.016150	(100% Raw Coconut Water)
0.025866	(100% Whole Wheat Bread)
0.015775	(2% Reduced Fat Milk)
0.036160	(Apple Honeycrisp Organic)
0.028616	(Asparagus)

- 100% Raw Coconut Water: Appears in 1.60% of transactions, indicating moderate popularity.
- 100% Whole Wheat Bread: Has a 2.60% support, reflecting a reasonably high frequency of purchase.
- 2% Reduced Fat Milk: Purchased in 1.58% of transactions, showing a moderate presence in the dataset.
- Apple Honeycrisp Organic: The most popular among the listed items, with a 3.62% support, indicating it is frequently bought.
- Asparagus: Appears in 2.83% of transactions, showing a solid level of customer interest.

These itemsets represent a range of product types with varying frequencies of purchase, with organic and fresh produce (like apples and asparagus) showing higher support compared to packaged items like coconut water and bread.

## PRODUCT BUNDLING STRATEGY

### Filtering Rules for Strong Bundling Potential:

Rules are filtered to identify potential product bundles with:

- Lift > 2: Demonstrating a meaningful relationship between products.
- Confidence > 0.2: Indicating a reliable pattern in purchasing behavior.

### Top Product Combination Selection:

The filtered rules are sorted by lift and confidence, with the top 10 high-lift and high-confidence product combinations identified as the most promising bundles.

### Targeted Customer Segments:

The identified product bundles are analyzed to cater to specific customer segments (e.g., organic shoppers, health-conscious buyers), enhancing the effectiveness of the bundles for targeted marketing campaigns.

## Summary of Key Product Bundling Insights

antecedents	consequents	support	confidence	lift
(Lime Sparkling Water)	(Sparkling WaterGrapefruit)	0.005650	0.285395	8.842637
(Bunched Cilantro)	(Limes)	0.005313	0.274347	4.586990
(Organic Cilantro)	(Limes)	0.007471	0.252647	4.224173
(Organic Garlic)	(Organic Yellow Onion)	0.009388	0.201069	4.168026
(Raspberries)	(Strawberries)	0.005149	0.210028	3.454510
(Bag of Organic Bananas, Organic Strawberries)	(Organic Hass Avocado)	0.006408	0.244467	2.691213
(Organic Lemon)	(Organic Hass Avocado)	0.009036	0.242131	2.665496
(Bag of Organic Bananas, Organic Baby Spinach)	(Organic Hass Avocado)	0.005187	0.242119	2.665364
(Organic Cucumber)	(Organic Hass Avocado)	0.007424	0.217136	2.390343
(Organic Strawberries, Organic Hass Avocado)	(Bag of Organic Bananas)	0.006408	0.369319	2.288464
(Organic Tomato Cluster)	(Organic Hass Avocado)	0.005555	0.203161	2.236495
(Organic Kiwi)	(Organic Strawberries)	0.005310	0.249018	2.212082
(Organic Raspberries)	(Organic Strawberries)	0.014402	0.247072	2.194801

antecedents	consequents	support	confidence	lift
(Organic Baby Spinach, Organic Hass Avocado)	(Bag of Organic Bananas)	0.005187	0.349446	2.165319
(Bag of Organic Bananas, Organic Hass Avocado)	(Organic Strawberries)	0.006408	0.241671	2.146817
(Organic Blueberries)	(Organic Strawberries)	0.010104	0.237418	2.109035
(Organic Navel Orange)	(Bag of Organic Bananas)	0.005962	0.328152	2.033378
(Michigan Organic Kale)	(Organic Baby Spinach)	0.005945	0.207253	2.014299
(Organic Whole String Cheese)	(Organic Strawberries)	0.005747	0.226439	2.011514

1. **SPARKLING WATER PAIRING:** Lime Sparkling Water often pairs with Grapefruit SparklingWater (Lift: 8.95).
2. **CILANTRO AND LIMES:** Both regular and organic cilantro frequently co-occur with limes, suggesting a fresh produce bundle.
3. **BERRY COMBOS:** Raspberries and strawberries, as well as organic blueberries and raspberries, show strong associations, ideal for berrypacks.
4. **ORGANIC PAIRINGS:** Organic lemons and avocados, as well as combinations like strawberries, bananas, and avocados, appeal to organic shoppers.
5. **BANANAS AS A COMPLEMENT:** Organic bananas commonly pair with items like organic oranges and avocados, ideal for a health-focused fruit bundle.

These bundles target popular pairings and healthy choices, enhancing product appeal in fresh and organic categories.

## INSIGHTS AND APPLICATIONS FOR PRODUCTBUNDLING

### Suggested Bundles:

Based on strong association rules, bundles are recommended that align with common customer buying patterns, enhancing cross-selling opportunities.

### Cross-Promotion and Marketing Opportunities:

High-lift product pairs provide ideal candidates for cross-promotions, encouraging sales of both primary and complementary items.

### Enhanced Customer Experience:

Bundling frequently purchased items has the potential to simplify shopping, improve customer satisfaction, and encourage repeat purchases.



## Customer Segmentation (Clustering)

### Collecting and Loading Data

Loading Data:

We started by loading four CSV files - orders, products, order\_products, and aisle.

- Orders Data: Contains information about each order.
- Products Data: Lists details about each product.
- Order-Products Data: Shows which products were in each order.
- Aisle Data: Groups products by aisle (category).

### Merging the Datasets

Combining Datasets:

Merged these files to create one complete dataset.

- Merging Orders and Order-Products: Linked them by order\_id so each order shows the products in it.
- Adding Products Data: Merged with the products list on product\_id to include product details.

### Sampling Data for Easier Analysis

Stratified Sampling:

Selected a sample that represents all customer types, using 99% of the data to keep it accurate.

- Sample Size: We kept the sample large to cover most customer behaviors.

### Adding Aisle Names

Mapping Aisle IDs to Names:

Converted the aisle data into a dictionary (list of pairs) to make aisle names easier to read.

- Aisle Mapping: Replaced aisle\_id with the actual aisle name for better understanding.

## Cross-Tabulating Users and Aisles

Creating a User-Aisle Table:

Created a table to see how often each user buys from each aisle.

- Row Normalization: Adjusted each row to compare user data more easily.
- User-Aisle Matrix: Shows each user's shopping habits across different aisles.

## Reducing Dimensions with PCA (Principal Component Analysis)

- Simplifying Data: Used PCA to reduce the number of variables and keep the most important information.
- Normalization: Standardized values so they are all on a similar scale.
- Choosing Key Components: Selected components that capture the most important data patterns.

## Finding Clusters with K-Means

K-Means Clustering:

Used K-means to group customers into segments based on similar buying patterns.

- Elbow Method: Chose the best number of clusters by looking at a graph of within-cluster variation.
- Optimal Cluster Number: Found the right number of clusters that balanced detail with simplicity.

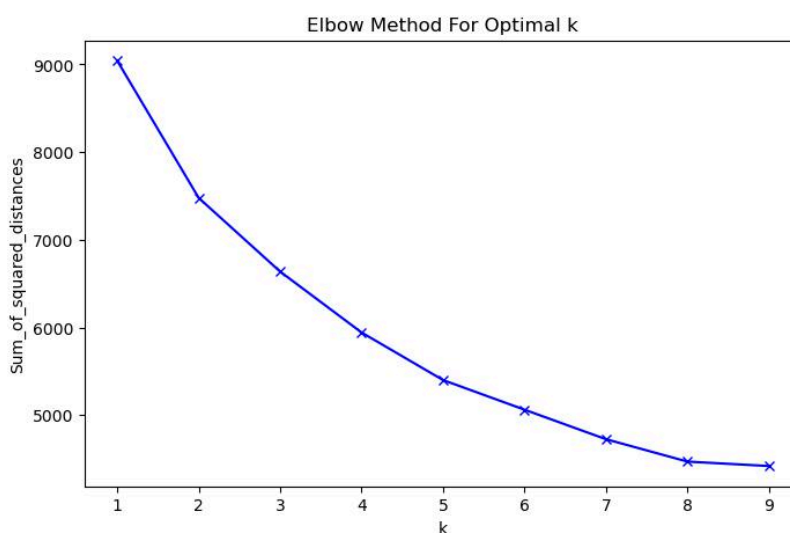


Fig 24:

## Visualizing Clusters

- Plotting Clusters: Made a scatter plot of clusters using the first two principal components to show group differences.
- Understanding Cluster Differences: Note how different clusters stand out, helping us understand each group's buying habits.

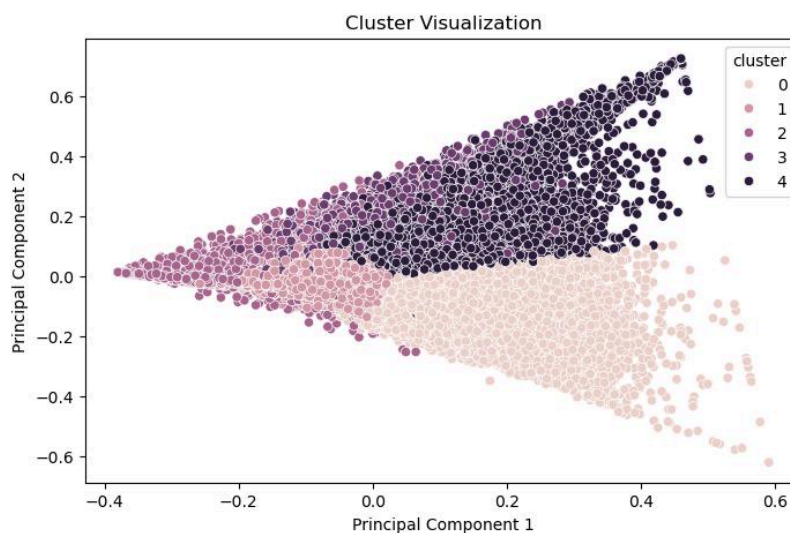


Fig 25:

## Finding Popular Products in Each Cluster

Top Products per Cluster:

Listed the most popular products for each customer group (cluster).

Cluster - 1		Cluster - 2	
Aisles		Aisles	
fresh vegetables	37.082596	fresh fruits	35.683368
fresh fruits	21.657433	fresh vegetables	15.200160
packaged vegetables fruits	11.983578	packaged vegetables fruits	12.193124

Cluster - 3

Aisles	
packaged produce	18.355877
fresh fruits	10.900314
packaged vegetables fruits	4.236706

Cluster - 4

Aisles	
fresh fruits	9.659493
fresh vegetables	7.536574
yogurt	7.163257

Cluster - 5

Aisles	
water seltzer sparkling water	20.909533
fresh fruits	3.283551
soft drinks	2.802056

## Final Insights

1. Cluster 1 has 55851 customers with a very strong inclination towards fresh vegetables followed by fresh fruits.
2. Cluster 2 has 37837 customers with a very strong inclination towards fresh fruits followed by fresh vegetables.
3. Cluster 3 has 7947 customers with a very strong inclination towards packaged produce followed by fresh fruits.
4. Cluster 4 has 99157 customers with moderately strong inclination towards fresh fruits followed by fresh vegetables.
5. Cluster 5 has 5417 customers with strong inclination towards water seltzer sparkling water followed by fresh fruits.

## Next Purchase Prediction (Supervised Learning)

The goal of this part of the project is to help identify popular product bundles and predict the next items that customers are likely to purchase. By doing this, marketing strategies such as targeted promotions and product bundling can be improved, ultimately aiming to increase the average order value (AOV).

To achieve this, patterns in customer behavior are being studied. Associations between products that are frequently bought together are being explored, and models are being used to predict what customers might buy next. This work supports more personalized shopping experiences and better marketing decisions.

For this part of the project, supervised machine learning methods were used to predict the target variable 'reordered' i.e. whether a product will be included in a customer's next order. Various features based on purchase history and user behavior were analyzed to create an effective model. The results can be applied to make shopping more engaging for customers while boosting sales.

### Initial Model Selection:

As the dataset is large only **30%** sample of the preprocessed data using stratified sampling, has been used for initial model selection using cross validation. Then the data is split into a training and testing set with a test size of **20%**. The supervised models that have been used initially for model selection are:

1. **Logistic Regression Variants:**
  - L1 Regularization: For feature selection and interpretability.
  - L2 Regularization: To prevent overfitting.
  - ElasticNet Regularization: Combines L1 and L2 benefits for balanced feature selection and regularization.
2. **Linear Discriminant Analysis(LDA):** Chosen for its efficiency in classifying features while reducing dimensionality.
3. **Quadratic Discriminant Analysis(QDA):** Included for cases where classes have differing variances
4. **Decision Tree:** Used for its simplicity, interpretability, and ability to capture non-linear relationships.
5. **Bernoulli Naive Bayes:** Used for probabilistic classification

6. **Random Forest:** An ensemble model chosen for its robustness, non-linear modeling capability
7. **XGBoost:** Preferred for its efficiency and strong performance on both linear and non-linear relationships.
8. **LightGBM:** Selected for faster training, memory efficiency, and native handling of categorical features.

### Cross-Validation and Evaluation

A **5-fold cross-validation** was conducted to ensure robust performance evaluation. The **F1 score** was used as the primary metric to balance precision and recall, crucial for imbalanced datasets. Each model's average F1 score across the folds was calculated to identify the best-performing model for further refinement. The results are:

Model	Cross_Val_Scores	Mean_Cross_Val
lightGBM	[0.84799736 0.84925832 0.84755292 0.84800397 0.84740065]	0.8480426445
xgboost	[0.84479638 0.8467846 0.84451508 0.84514197 0.84372626]	0.8449928584
Random Forest	[0.82979977 0.8312377 0.8290435 0.82920699 0.81302982]	0.8264635576
Naive Bayes	[0.81615423 0.81740037 0.81529755 0.81576622 0.81608204]	0.8161400825
Logistic Regression with l2 regularization	[0.81503533 0.81589514 0.81499544 0.81561126 0.81444871]	0.8151971756
Logistic Regression with l1 regularization	[0.81503164 0.81589349 0.81500064 0.81560676 0.8144509 ]	0.8151966847
Logistic Regression with elastic net regularization	[0.81503164 0.81589063 0.81499995 0.81560758 0.81444871]	0.8151957032
Decision Tree	[0.81333755 0.81463314 0.81252061 0.8130407 0.81236145]	0.8131786915
Linear Discriminant Analysis	[0.8080748 0.80827769 0.80753246 0.80795217 0.80671055]	0.8077095344
Quadratic Discriminant Analysis	[0.7466837 0.75823929 0.80011634 0.7499824 0.73532752]	0.7580698504

## Model Comparison:

### Logistic Regression Variants (L1, L2, ElasticNet):

- **Performance:** Performed moderately, with F1 scores slightly lower than ensemble models. Provided interpretability, particularly useful for feature selection (L1) and understanding relationships in the data.

### Linear and Quadratic Discriminant Analysis (LDA & QDA):

- **Performance:** LDA performed better than QDA due to its ability to reduce dimensionality efficiently. LDA is computationally efficient, but it assumes linear boundaries, which limits its ability to capture non-linear dependencies.

### Decision Tree:

- **Performance:** Offered decent performance but was prone to overfitting, as shown by its inability to generalize well across folds. Highly interpretable and captured non-linear relationships.

### Bernoulli Naive Bayes:

- **Performance:** Achieved a competitive F1 score of 81.61%. Fast and effective for probabilistic predictions, making it a good baseline.

### Random Forest:

- **Performance:** Robust performance with an F1 score of 82.81%. Naturally handles overfitting by averaging multiple decision trees.

### XGBoost and LightGBM:

- **Performance:** Both models achieved the highest F1 scores among all tested models (84.49% and 84.80%, respectively).

## Conclusion

After cross validation on each model, **four** best models are selected on the basis of their average **F1 Score**. In this dataset, **LightGBM, XGBoost, Random Forest** and **Naive Bayes** are the four best performing models with their respective average F1 scores which are **84.80%, 84.49%, 82.81%** and **81.61%**.

## Final Model Selection

After the initial model selection, the **100%** of the preprocessed data has been used for final model selection. The supervised models that are used in final model selection are **LightGBM, XGBoost, Random Forest** and **Naive Bayes**. The data is split into training and testing sets with a test size of **20%**.

## Cross-Validation and Evaluation

A **5-fold cross-validation** was also conducted in final model selection to ensure robust performance evaluation. The **F1 score** was used as the primary metric to balance precision and recall, crucial for imbalanced datasets. Each model's average F1 score across the folds was calculated to identify the best-performing model for further refinement. The results are:

Model	Cross_Val_Scores	Mean_Cross_Val
lightGBM	[0.8479707 0.84852718 0.84841494 0.84799102 0.84759148]	0.8480990646
xgboost	[0.84505531 0.84539022 0.84529326 0.84475559 0.84403358]	0.844905592
Random Forest	[0.82916487 0.82318999 0.82998041 0.82951172 0.82923899]	0.8282171967
Naive Bayes	[0.81574358 0.81635277 0.81638002 0.81566897 0.81556617]	0.8159423027

### Model Comparison:

#### LightGBM:

- **Performance:** Outperformed all other models with the highest F1 score (84.80%) on the full dataset. Its ability to handle categorical data natively and efficiently process large datasets made it the best fit.

#### XGBoost:

- **Performance:** Achieved the second-highest F1 score (84.49%), close to LightGBM.

#### Random Forest:

- **Performance:** Achieved a slightly lower F1 score (82.81%), still showing robust performance.



### Bernoulli Naive Bayes:

- **Performance:** Had the lowest F1 score (81.61%) among the final models but remained competitive given its simplicity. Extremely fast and computationally efficient, making it a useful lightweight alternative.

### Conclusion

After applying cross validation on the four initially selected models on the whole dataset, **LightGBM** is performing best out of those four models with the maximum average **F1 Score 84.80%**.

### Final Model Evaluation

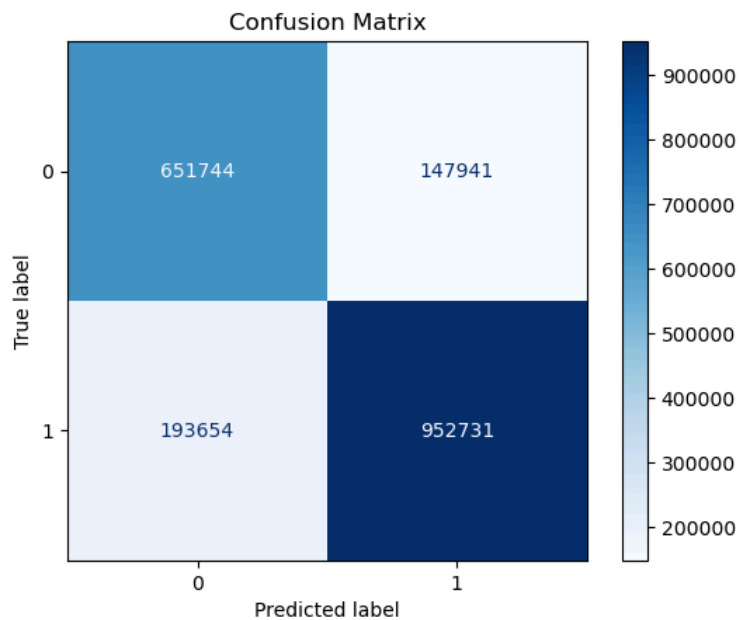
The LightGBM model was selected as the final model based on its superior performance during the evaluation phase. The following metrics were used to assess the model's effectiveness on the test set:

	precision	recall	f1-score	support
0	0.77	0.82	0.79	799685
1	0.87	0.83	0.85	1146385
accuracy			0.82	1946070
macro avg	0.82	0.82	0.82	1946070
weighted avg	0.83	0.82	0.83	1946070

Fig 26:

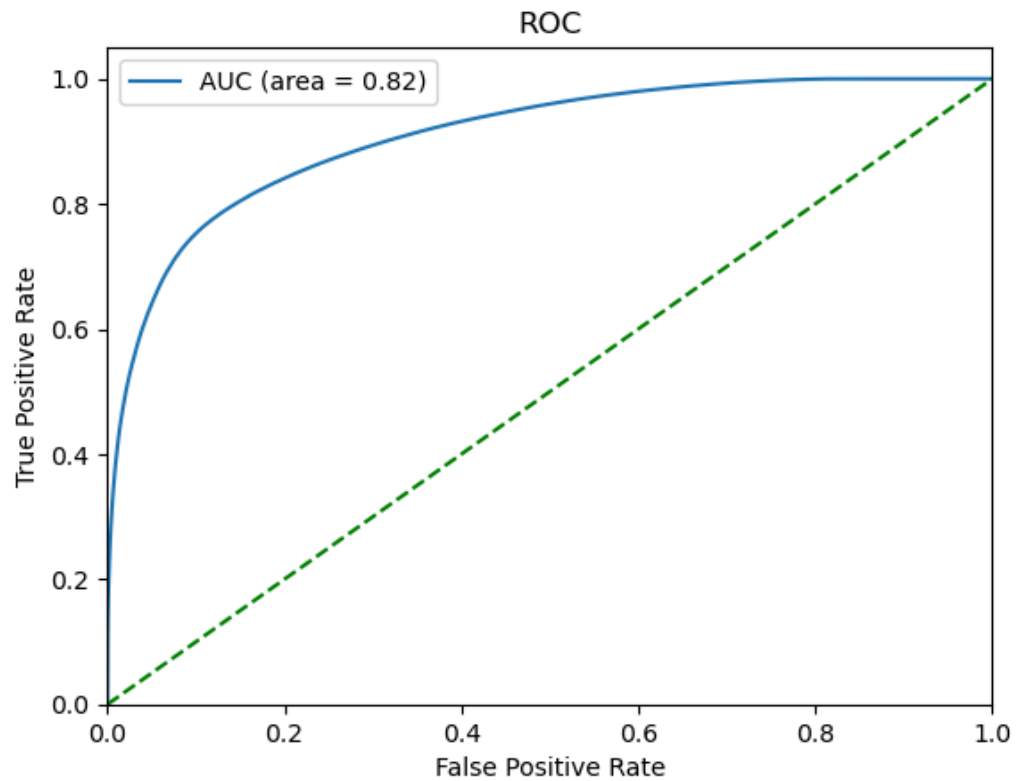
- **Accuracy Score: 0.8244**  
The model correctly classified 82.44% of all instances, demonstrating strong overall performance.
- **Precision Score: 0.8656**  
LightGBM achieved a high precision, indicating that a significant proportion of predicted positive cases were indeed true positives. This is particularly important in reducing false positives.
- **Recall Score: 0.8311**  
The model captured 83.11% of actual positive cases, reflecting its effectiveness in identifying true positives, even at the cost of some false negatives.
- **F1 Score: 0.8480**  
The F1 score, which balances precision and recall, highlights the model's robustness, particularly in managing the trade-off between false positives and false negatives.

- **Confusion Matrix:**



- **True Label 0:** The model correctly predicted 651,744 instances as class 0, and incorrectly predicted 147,941 instances as class 1.
- **True Label 1:** The model correctly predicted 952,731 instances as class 1, and incorrectly predicted 193,654 instances as class 0.
- The matrix displays the raw counts of the predictions, which is useful for understanding the model's performance.

- **ROC-AUC Curve:**



- The Area Under the Curve (AUC) is 0.82, which indicates that the model has a reasonably good performance. AUC values range from 0 to 1, with 1 representing a perfect model and 0.5 indicating a random classifier.
- The closer the ROC curve is to the top-left corner (0, 1), the better the model's performance, as it indicates a high TPR and low FPR. From ROC-AUC curve it can be said that the model has room for improvement

## Hyperparameter Tuning for Purchase Prediction Model

The goal of this part of the project is to optimize the hyperparameters of the model selected in the previous section - 'LightGBM'. The objective is to improve the performance metrics, mainly F1-score, to make a more accurate prediction of the customer's next purchase.

### Model Details - LightGBM

LightGBM is an open-source, high-performance implementation of a gradient boosting framework developed by Microsoft. It is optimized for both speed and memory usage, and it excels for large datasets

#### Why LightGBM for This Project?

1. **Efficiency:** LightGBM's ability to handle large datasets and high-dimensional data made it ideal for analyzing purchase history and customer behavior.
2. **Performance:** Its leaf-wise growth strategy resulted in higher F1 scores compared to other models.
3. **Scalability:** LightGBM's speed and support for distributed computing ensure scalability for real-world applications.

### Hyperparameter Tuning Methodology

#### Hyperparameters Selected

##### Learning Parameters:

- **n\_estimators** - Number of boosting rounds
- **num\_leaves** - Maximum number of leaves in a tree. Larger values increase accuracy but risk overfitting
- **max\_depth** - Maximum Depth of a tree
- **min\_data\_in\_leaf** - Minimum number of samples per leaf. Higher Values prevent overfitting

### Regularization Parameters:

- **lambda\_l1** - L1 regularization term on weights. Adds sparsity to the model
- **lambda\_l2** - L2 regularization term on weights. Helps control overfitting
- **min\_gain\_to\_split** - Minimum loss reduction required to split a leaf node. Larger values make the model training faster

### Data Sampling Parameters:

- **bagging\_fraction** - Fraction of data used for each iteration (for bagging). Helps with overfitting
- **bagging\_freq** - Frequency of bagging. Also helps deal with overfitting

### Tuning Technique - Bayesian Optimization

Bayesian optimization is an advanced and efficient technique used for hyperparameter tuning in machine learning models. Unlike traditional methods like grid search or random search, which involve exploring hyperparameter space blindly, Bayesian optimization uses a probabilistic approach to intelligently navigate and find the best set of hyperparameters. This is particularly useful when tuning complex models like LightGBM or XGBoost, where the hyperparameter space is large and non-linear.

### Data Splitting Strategy

As done in the previous section, only **30%** sample of the preprocessed data using stratified sampling, has been used for hyperparameter tuning using train\_test\_split at 80-20 split for train and test set respectively.

### Metric Used - F1 Score

The **F1** score is the harmonic mean of precision and recall. It thus symmetrically represents both precision and recall in one metric. Tuning the hyperparameter using this metric will help preserve the balance between precision and recall without sacrificing either metric.

## Experiments and Results

### Baseline Performance

The model's performance without any hyperparameter tuning can be seen by observing following metrics

- Accuracy score: 0.8244693150811636
- Precision score: 0.8655902939295267

- Recall score: 0.8310742028201695
- F1 score: 0.8479811593564383

### Distribution of Hyperparameters

Hyperparameter	Distribution	Range
n_estimators	Discrete	[50,1000]
num_leaves	Discrete	[20, 200]
max_depth	Discrete	[-1, 15]
min_data_in_leaf	Discrete	[10, 100]
lambda_l1	Log-Uniform	[0.0001, 10.0]
lambda_l2	Log-Uniform	[0.0001, 10.0]
min_gain_to_split	Log-Uniform	[0.0001, 1.0]
bagging_fraction	Uniform	[0.6, 1.0]
bagging_freq	Discrete	[1, 10]

### Best Parameters

Parameters	Values
n_estimators	971
num_leaves	190
max_depth	14
min_data_in_leaf	92
lambda_l1	0.3439142698852568
lambda_l2	1.3288318289273937
min_gain_to_split	0.24035195998981623
bagging_fraction	0.699417894976562
bagging_freq	9

## Performance Gains

Below is a comparison of the **Base Model** (before tuning) and the **Tuned Model** (after tuning):

Metric	Base Model	Tuned Model	Improvement
Accuracy	82.44%	82.59%	+0.15%
Precision	86.56%	86.76%	+0.20%
Recall	83.10%	83.13%	+0.03%
F1 score	84.80%	84.91%	+0.11%

### Interpretation

- **Accuracy** - The improvement suggests that tuned model correctly classifies 0.15% of data more than that of base model
- **Precision** - The improvement suggests that the number of false positives has reduced in the tuned model.
- **Recall** - The small improvement suggests that the tuned model captures slightly more positive cases.
- **F1 Score** - The improvement of 0.11% suggests that the tuned model balances the classification of both positive and negative cases better than the base model.

Even though tuning resulted in small gains, they may lead to an improvement in making significant business decisions. Even an increase of 0.15% in accuracy may lead to making thousands of more accurate predictions.

**Consistency** - The model has shown improvement in all the metrics, instead of focusing on one aspect over the other, which signifies a robust performance.

## Conclusion

The purchase pattern analysis revealed critical insights into customer behavior, product bundling opportunities, and next-purchase prediction. By leveraging advanced techniques like exploratory data analysis, feature engineering, clustering, and machine learning, the study successfully identified high-value product bundles and customer segments.

The findings support the development of tailored marketing strategies to enhance customer engagement and boost average order value. Techniques such as market basket analysis using the Apriori algorithm provided actionable insights for bundling complementary products, while predictive modeling using LightGBM demonstrated robust performance in forecasting customer purchase tendencies.

With hyperparameter tuning and strategic data processing, the study not only achieved high model accuracy but also reinforced the importance of precision and recall in managing imbalanced datasets. These insights lay the foundation for a data-driven approach to customer personalization, efficient inventory management, and improved shopping experiences.

Future efforts could explore deeper integration of real-time data and customer feedback to refine the models and strategies further, ensuring continued alignment with dynamic market trends and customer expectations.