# Customer Behavior & Market Basket Analysis

Alka

Centennial College

Business Analytics Capstone - BA723

David Parent

2022, August 12

# ABSTRACT

Predictive analytics combines several statistical techniques such as data mining, predictive modeling, and machine learning to make predictions about future events based on historical data. This research paper focuses on the development of an end-to-end automated solution that helps predict which reordered products a user will buy again. First, we perform the data analysis to understand the orders, then segment consumers based on buying behavior and explore the association between products, and at last deploy prediction models. The sequential ensemble model derives an accuracy of 72.82% and a Recall score of 73.44%. The lift chart shows that taking the 30% of the records that are ranked by the model as “most probable 1’s” yields 5 times as many 1’s as would randomly selecting 30% of the records. This model will help Instacart to send personalized push notifications, estimate the basket of the user, and pre-fill the cart with the products that the customer is most likely to buy. This framework will increase the value of the service by saving customers’ time and triggering new purchases through personalized push notifications.

# TABLE OF CONTENT

**ABSTRACT** **2**

**TABLE OF CONTENT** **3**

**1.0 INTRODUCTION** **5**

1.1 Background 5

1.2 Problem Statement 7

1.3 Objective & Measurement 8

1.4 Significance of the Project 9

1.5 Restrictions 10

**2.0 DATA SOURCE** **11**

2.1 Data Set Introduction 11

2.2 Data Dictionary 11

2.3 Initial Data Preparation 14

**3.0 DATA EXPLORATION** **16**

3.1 Preliminary Exploration 16

3.2 Data Exploration & Visualization 18

3.3 Summary 29

**5.0 CUSTOMER SEGMENTATION** **30**

**6.0 ASSOCIATION RULE** **37**

**7.0 FEATURE ENGINEERING** **42**

7.1 Feature Expansion 42

7.2 Data Preparation 45

7.3 Class Distribution 45

**8.0 MODEL EXPLORATION** **49**

8.1 Model Introduction 49

8.2 Model Techniques 51

8.2.1 DECISION TREE 51

8.2.2 RANDOM FOREST 54

8.2.3 LOGISTIC REGRESSION 56

8.2.4 EXTREME GRADIENT BOOSTING (XGBoost) 57

8.2.5 LIGHTGBM 60

8.2.6 CATBoost 61

8.3 Model Performance 63

8.4 Model Selection 67

8.4.1 Model Effectiveness 67

8.4.2 Feature Importance 67

**9.0 GOVERNANCE 71**

9.1 Data Drift 72

9.2 Model Stability Measure 73

9.3 Action 73

**10.0 CONCLUSION & RECOMMENDATION 74**

10.1 Conclusion & Impact on business Problem 74

10.2 Future Work 75

**REFERENCES 76**

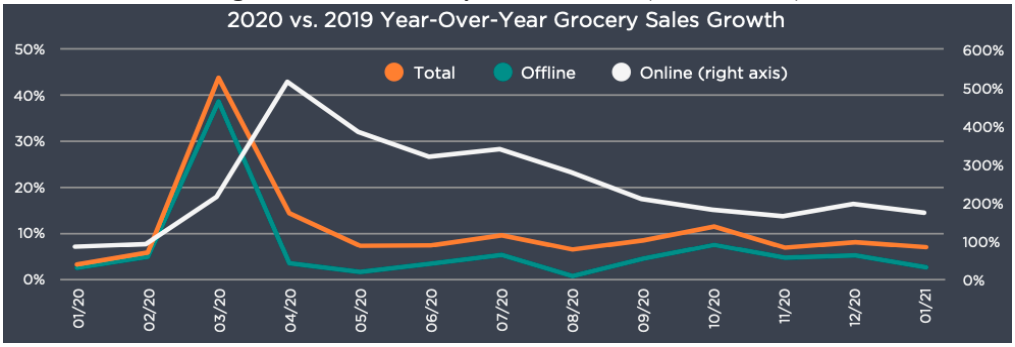
# 1.0 INTRODUCTION

## 1.1 Background

Instacart is an America-based company that provides same-day grocery service in the United States and Canada. Instacart was founded a decade ago in 2012 in the San Francisco Bay Area to deliver groceries online to customers from local grocers. By 2017, Instacart was operating nationwide in the US while also expanding across Canada (Instacart, 2022). Instacart operates on a crowdsourced marketplace model in which a consumer, also known as a 'user', places an order from the desired store, and a 'shopper' near that store shops, packs and delivers the products within the designated time frame (JWorks,2015).

Over the last several years the e-commerce industry has experienced an exponential growth rate which was further accelerated by covid-19 pandemic and lockdowns. According to the 'State of Grocery Report 2021' of 1010Data, e-commerce spending increased 66% year over year in 2020 while online grocery spending saw a year over year growth rate of 133% in 2020 (1010Data, 2021). While the YOY growth rate of overall online grocery sales was 8% in 2021, Instacart's sales grew at 14% (Redman, 2022). The pandemic played a huge role in changing the trend of grocery purchases (Figure 1). Instacart drove this adoption of online groceries with significant growth in users; 46% of Instacart users were new in 2020 (1010Data, 2021).

**Figure 1: YOY Grocery Sales Growth (2020 vs 2019)**




Source: https://1010data.com/media/3019/2021_1010data_stateofgroceryreport.pdf

According to a report by Mckinsey, during the pandemic grocery e-commerce industry experienced growth equivalent to five years in just five months. A survey by Mckinsey shows that the grocery industry expects e-commerce penetration to more than double in the next three to five years. Online grocery delivery is here to stay and has become the 'next normal'. Consumers use e-commerce for grocery shopping for primary grocery purchases, top-off, and stock up on supplies (Chandra et al., 2022) (Figure 2)

**Figure 2: Main occasion to ordering groceries online**


Source:https://www.mckinsey.com/industries/consumer-packaged-goods/our-insights/making-online-grocery-a-winning-proposition

Safety and convenience were the major factors in the growth of e-commerce during the pandemic but now consumers value unique features such as price comparison, product comparison, assortment, personalized promotions, time savior, 24/7 ordering, and customer services. Consumers expect grocery delivery service to be fast, reliable, and flexible. There is a complete portfolio of e-commerce options such as same-day delivery, two-hour delivery, instant delivery, and click and collect. According to Mckinsey's report, consumers are finding ways to save money, be healthier, reintroduce

their joy of cooking, and find the best price & promotions. These needs can be better satisfied with an online-based website rather than a brick-and-mortar store and Figure 3 shows the online features preferences of consumers (Aull et al., 2021).

**Figure 3: Importance of Online interface features**


Source:https://www.mckinsey.com/industries/consumer-packaged-goods/our-insights/making-online-grocery-a-winning-proposition

## 1.2 Problem Statement

Post-pandemic, the grocery delivery industry has matured, scaled, and become standard. While Instacart facilitated the early adoption, competitors are soon catching up. Competitors such as Doordash and Skipthedishes have expanded their food delivery business to grocery and convenience products, while big grocery chains such as Walmart, Metro, Loblaws, Longos, etc. have also started providing their delivery services. So, to remain competitive in the market and retain existing customers, Instacart must provide incredibly better, faster, and quality services to its customers.

After adding all new customers in the past few years, Instacart aims to retain those customers by engaging with customers and building personal relations. Figure 3 shows that quick selection & checkout and easy access & reordering of past purchases are the two most important features preferred by customers.

This project will address following two question in different aspects to solve business problem:

- What changes need to be made to the current website/application interface?
- What changes need to be made to the current product portfolio?

### 1.3 Objective & Measurement

The primary objective of the project is to predict which products will be in the user's next order that a user has already purchased in the past. Determining a user's basket will help solve the following two business objectives:

- Save customer time
- Trigger new purchases by personalized push notifications (Stanley, 2018)

The above objective will help in increasing the value of service for the customers and thereby increasing retention. This project will also identify the customer buying behavior and preferences based on the historical transactional data and will help in understanding different segments of the customers.

The preliminary study will also uncover the association between products and the combinations of the products that are frequently bought together. Another significant aspect of the project is to understand the growing preferences of consumers toward organic, vegan, and gluten-free products. Next, our models will help us identify the features/variables that are significant in the user's decision to reorder a product.

The success of the project will be measured primarily by A/B testing on the following metrics:

- Push notification performance metric: Number of customers who clicked on the call to action.
- Accurate prediction of the products: Number of products reordered by users which are predicted by the model.
- Average shopping time: Average time customer takes when the cart is pre-filled by reordered products or customers are provided with the list of most reordered products.
- Customer retention: Number of returning customers

**1.4 Significance of the Project**

- Prediction of the user's reordered product will also provide a guide for retailers to maintain inventory levels.
- Understanding consumer behavior and the association between products will help retailers to forecast demand and adjust supply chain supply.
- The resulting models can be replicated at scale with large and latest datasets. The results from the model can be used to fine-tune the existing models and avoid pre-existing errors before deploying it on a large scale.
- Additionally, combining the current model with demographic data of consumers will help in detailed customer segmentation and expand the applicability of the model.
- Practical deployment of the model will help in creating better promotional schemes and accurate prediction of the future order and customer behavior can be used to create personalized offers that will trigger new purchases.

### 1.5 Restrictions

Limited RAM was the major restriction in developing predictive models. Processing large datasets and performing any computation on them becomes challenging with limited RAM. Algorithms will be evaluated with 12GB RAM. This challenge was countered using undersampling for predictive models. Undersampling is not the most ideal technique because a lot of significant information may be lost, however, with the current situation it is the most ideal approach. For future work, the whole dataset will be used which not only will account for an accurate representation of the data but also improve the accuracy of the model.

Similarly, the apriori and association model was trained on the top 200 selling products. Training only on 200 products left the rest of 49,488 products out of the training. Due to only studying the 200 most sold products, products from certain departments are never studied. Using complete data would have increased the processing time significantly, therefore a subset of the dataset was used.

Dataset was collected in 2017 and there has been a significant change in the dietary needs of the consumers in the last couple of years. While the dataset presents information about special preferences like organic, vegan, and gluten-free, it is not the most representative of the current market situation. Demographic data is also absent in the development of clustering models. Access to and use of such information could greatly enrich not only the segmentation power but also the nuances of actual customer behavior that eventually leads to customer preference.

# 2.0 DATA SOURCE

## 2.1 Data Set Introduction

Dataset comes from Instacart's first public open-source data released in 2017 in their official blog post. The dataset contains a sample of over three million grocery orders of more than 2,00,000 de-identified Instacart users. The information contains each user's order number between 4 and 100. The information provided is completely anonymous to protect the privacy of the users and retailers. User ID in the dataset is completely randomized and the only information provided is the user's sequence of order and products in that order. Similarly, only products ordered are provided and no retailer ID is stated.

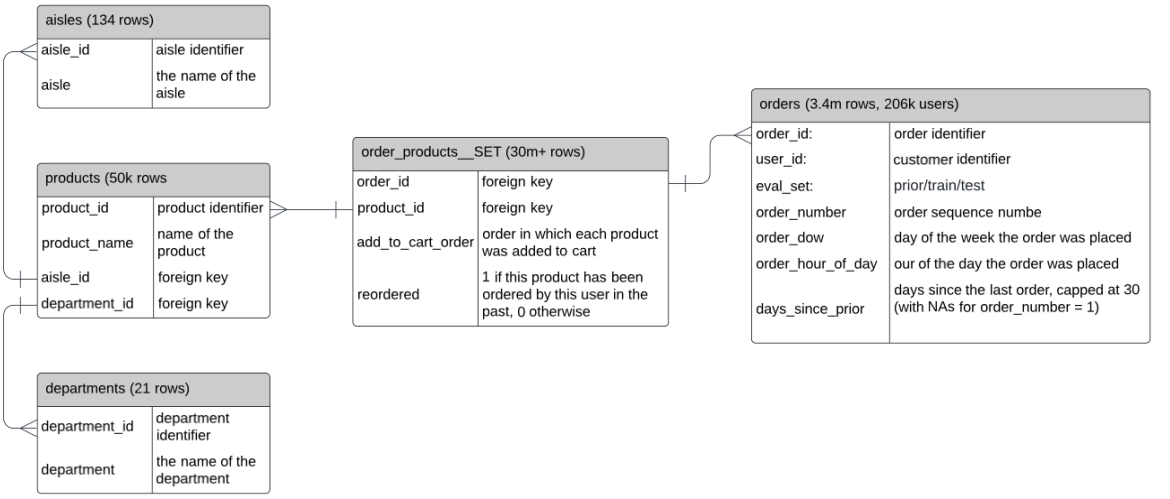
Instacart clarified that the dataset provided is a heavily biased subset of Instacart's production data and not a representative sample of products, users, or their purchasing behavior.

## 2.2 Data Dictionary

Dataset contain following five tables:

- Orders: Contains information about all 3.4 million orders of over 200,000 users
- Products: Contains product information
- Aisle: Contains aisle id and aisle name
- Department: Contains department id and name
- Order_products_prior: Contains information orders prior to the most recent order
- Order_products_train: Contains information about the most recent orders

**Figure 4: Entity Relationship Diagram**




| **orders (3.4m rows, 206k users)** | | |
|---|---|---|
| **column** | **datatype** | **description** |
| order_id | integer | order identifier |
| user_id | integer | customer identifier |
| eval_set | string | which evaluation set this order belongs in (see SET described below) |
| order_number | integer | the order sequence number for this user (1 = first, n = nth) |
| order_dow | integer | the day of the week the order was placed on |
| order_hour_of_day | integer | the hour of the day the order was placed on |
| days_since_prior | integer | days since the last order, capped at 30 (with NAs for order_number = 1) |

| **products (50k rows)** | | |
|---|---|---|
| **column** | **datatype** | **description** |
| product_id | integer | product identifier |
| product_name | string | name of the product |

| | | |
|---|---|---|
| aisle_id | integer | foreign key |
| department_id | integer | foreign key |

| **aisles (134 rows)** | | |
|---|---|---|
| **column** | **datatype** | **description** |
| aisle_id | integer | aisle identifier |
| aisle | string | the name of the aisle |

| **departments (21 rows)** | | |
|---|---|---|
| **column** | **datatype** | **description** |
| department_id | integer | department identifier |
| department | string | the name of the department |

| **order_products__SET (prior/train)** | | |
|---|---|---|
| **column** | **datatype** | **description** |
| order_id | integer | foreign key |
| product_id | integer | foreign key |
| add_to_cart_order | integer | order in which each product was added to cart |
| reordered | integer | 1 if this product has been ordered by this user in the past, 0 otherwise |

where SET is one of the two following evaluation sets (eval_set in orders):

- "prior": orders prior to that users most recent order (~3.2m orders)
- "train": most recent orders (~131k orders) (The Instacart Online Grocery Shopping Dataset 2017 Data Descriptions, 2017)

### 2.3 Initial Data Preparation

We have a large dataset with more than three million records. Memory usage of the 'order_product_prior' table is close to 990 MB. Large datasets increase the processing time and decrease the computation speed. So a function (Figure 6) written by Guillaume Martin to reduce the memory usage in pandas will be used (Martin, 2018). The following function optimizes the memory usage by effective use of datatype. It returns the memory usage of each column in MB and then iterates through each column and then modifies the datatype. The smallest integer in NumPy is int8 which needs only 1 byte and can store values between -128 and 127. So it makes sense in our case to use the smallest integer type and reduce the memory usage significantly. The below function reduces memory usage by more than 65%.

**Figure 5: Result of reduce_mem_usage function**

```
reduce_mem_usage(orders)
reduce_mem_usage(order_prior)
reduce_mem_usage(order_train)
```

```
Memory usage of dataframe is 182.71 MB
Memory usage after optimization is: 45.68 MB
Decreased by 75.0%
Memory usage of dataframe is 989.82 MB
Memory usage after optimization is: 340.25 MB
Decreased by 65.6%
Memory usage of dataframe is 42.26 MB
Memory usage after optimization is: 13.20 MB
Decreased by 68.7%
```

**Figure 6: reduce_mem_usage function**

```
def reduce_mem_usage(df):
    start_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

    for col in df.columns:
        col_type = df[col].dtype

        if col_type != object:
            c_min = df[col].min()
            c_max = df[col].max()
            if str(col_type)[:3] == 'int':
                if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                    df[col] = df[col].astype(np.int8)
                elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                    df[col] = df[col].astype(np.int16)
                elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                    df[col] = df[col].astype(np.int32)
                elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                    df[col] = df[col].astype(np.int64)
            else:
                if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                    df[col] = df[col].astype(np.float16)
                elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                    df[col] = df[col].astype(np.float32)
                else:
                    df[col] = df[col].astype(np.float64)
        else:
            df[col] = df[col].astype('category')

    end_mem = df.memory_usage().sum() / 1024**2
    print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
    print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))
```

# 3.0 DATA EXPLORATION

## 3.1 Preliminary Exploration

**Figure 7: Check for duplicate rows in the dataset.**

```
print(aisle.duplicated().any())
print(dept.duplicated().any())
print(products.duplicated().any())
print(orders.duplicated().any())
print(order_prior.duplicated().any())
print(order_train.duplicated().any())
```

```
False
False
False
False
False
False
```

.duplicated() method checks for duplicate values and returns a boolean value(True/False).

Next, we will look for null values in the data frames using .isna() and sum the number of null values.

**Figure 8: Check for null values in the dataset.**

```
print(aisle.isna().sum())
print(dept.isna().sum())
print(products.isna().sum())
print(order_prior.isna().sum())
print(order_train.isna().sum())
print(orders.isna().sum())
```

There are 206,209 null values in days_since_prior_order column in orders tables. As stated earlier in the data description, day_since_prior_order is null for the user's first order. We have 206,209 users in the dataset and 206,209 first time orders. In a later stage we will replace null values with 0.

In the `orders` dataframe, `order_dow` column will be converted into a more interpretable form.

Similarly, hour_of_the_day will be converted into ‘part_of_day’.

**Figure 9: Converting day of the week column into weekday column**

```
[ ]  def weekday(dow):
       if dow == 0:
         return 'Saturday'
       elif dow ==  1:
         return 'Sunday'
       elif dow == 2:
         return 'Monday'
       elif dow == 3:
         return 'Tuesday'
       elif dow == 4:
         return 'Wednesday'
       elif dow == 5:
         return 'Thursday'
       else:
          return 'Friday'

     orders['order_dow'] = orders['order_dow'].apply(weekday)
```

**Figure 10: Creating part_of_the_day column**

```
[ ]  def part_of_day(hour):
         if hour >= 6 and hour < 10:
             return "Morning"
         elif hour >= 10 and hour < 12:
             return "Day"
         elif hour >= 12 and hour < 14:
             return "Lunch"
         elif hour >= 14 and hour < 17:
             return "Afternoon"
         elif hour >= 17 and hour < 20:
             return "Evening"
         else:
             return "Night"

     orders['part_of_day'] = orders['order_hour_of_day'].map(part_of_day)
```

**Figure 11: New orders table**

```
[ ]  orders.head()
```

| | order_id | user_id | eval_set | order_number | order_dow | order_hour_of_day | days_since_prior_order | part_of_day |
|---|---|---|---|---|---|---|---|---|
| **0** | 2539329 | 1 | prior | 1 | Monday | 8 | NaN | Morning |
| **1** | 2398795 | 1 | prior | 2 | Tuesday | 7 | 15.0 | Morning |
| **2** | 473747 | 1 | prior | 3 | Tuesday | 12 | 21.0 | Lunch |

Next we will merge ‘order_prior’ table with ‘products’, ‘department’ and ‘aisle’ table and at the end, merging ‘merged_orders’ table with ‘orders’ table to create a full dataset table.

**Figure 12: Creating Merged_orders table**

```
#merging order_prior with products, department and aisle dataset
merged_orders = order_prior.merge(products, on = 'product_id', how = 'left').merge(dept,
                on = 'department_id', how = 'left').merge(aisle, on = 'aisle_id', how = 'left')

merged_orders.head()
```

| | order_id | product_id | add_to_cart_order | reordered | product_name | aisle_id | department_id | department | aisle |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 33120 | 1 | 1 | Organic Egg Whites | 86 | 16 | dairy eggs | eggs |
| 1 | 2 | 28985 | 2 | 1 | Michigan Organic Kale | 83 | 4 | produce | fresh vegetables |
| 2 | 2 | 9327 | 3 | 0 | Garlic Powder | 104 | 13 | pantry | spices seasonings |

**Figure 13: Creating full dataset table**

```
#full dataset
dataset = merged_orders.merge(orders, on = 'order_id', how = 'left')
```

```
#reordering column sequence
dataset = dataset[['order_id', 'user_id', 'order_number' ,'product_name', 'reordered', 'order_dow', 'order_hour_of_day','part_of_day',
                   'department', 'aisle', 'add_to_cart_order', 'days_since_prior_order', 'eval_set']]

dataset.head()
```

| | order_id | user_id | order_number | product_name | reordered | order_dow | order_hour_of_day | part_of_day | department | aisle | add_to_cart_order | days_since_pric |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 202279 | 3 | Organic Egg Whites | 1 | Thursday | 9 | Morning | dairy eggs | eggs | 1 | |
| 1 | 2 | 202279 | 3 | Michigan Organic Kale | 1 | Thursday | 9 | Morning | produce | fresh vegetables | 2 | |

## 3.2 Data Exploration & Visualization

In the first step, we will analyze the data through graphs & plots to better understand the data. A better understanding of the data helps in optimum use of the data at a later stage and refines the analytics approach to the problem. This step will help in identifying the pattern in the data and informed analysis for feature engineering and modeling.

Libraries used are: Pandas, NumPy, Matplotlib, Seaborn & Squarify.

## Orders analysis:

**Figure 14: Total number of orders per user**

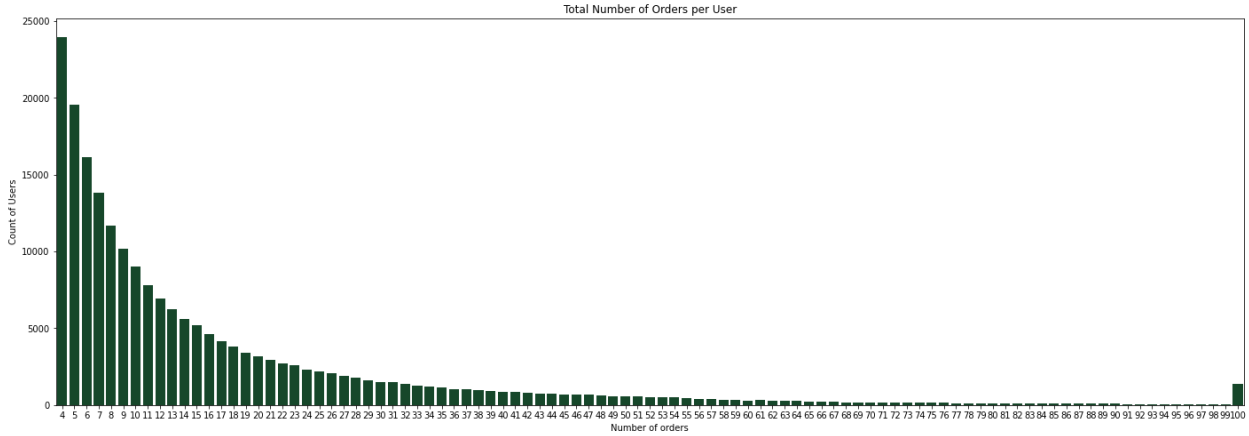



**Figure 15: 'Total Number of Orders Per Day'**

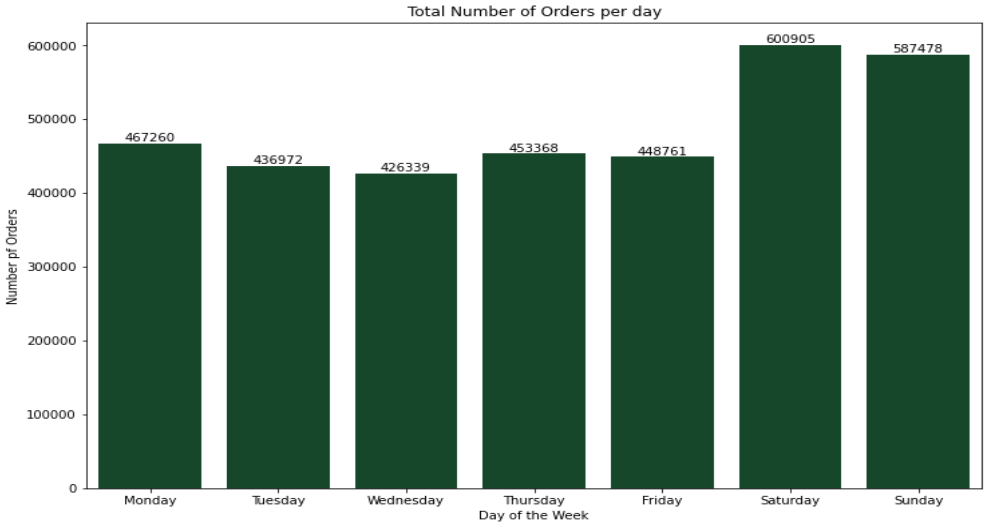



**Figure 16: 'Total Number of Orders Per Part of the Day'**

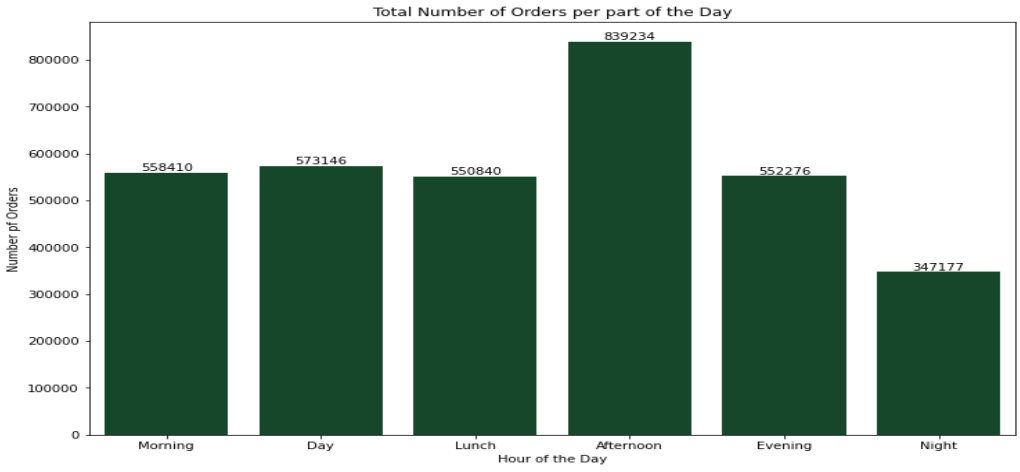


**Figure 17: 'Number of days since prior Order'**

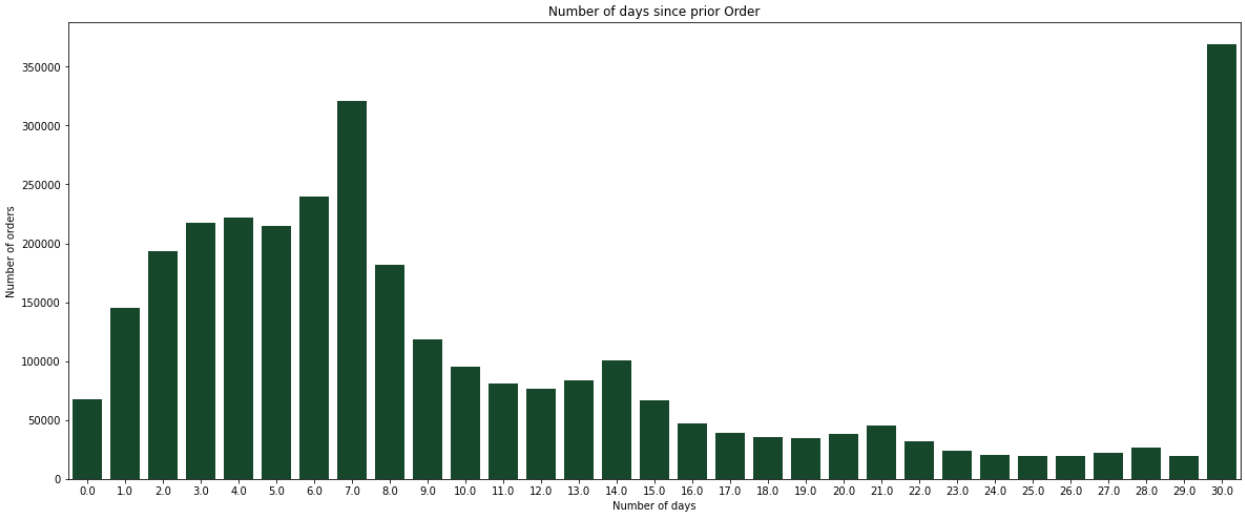
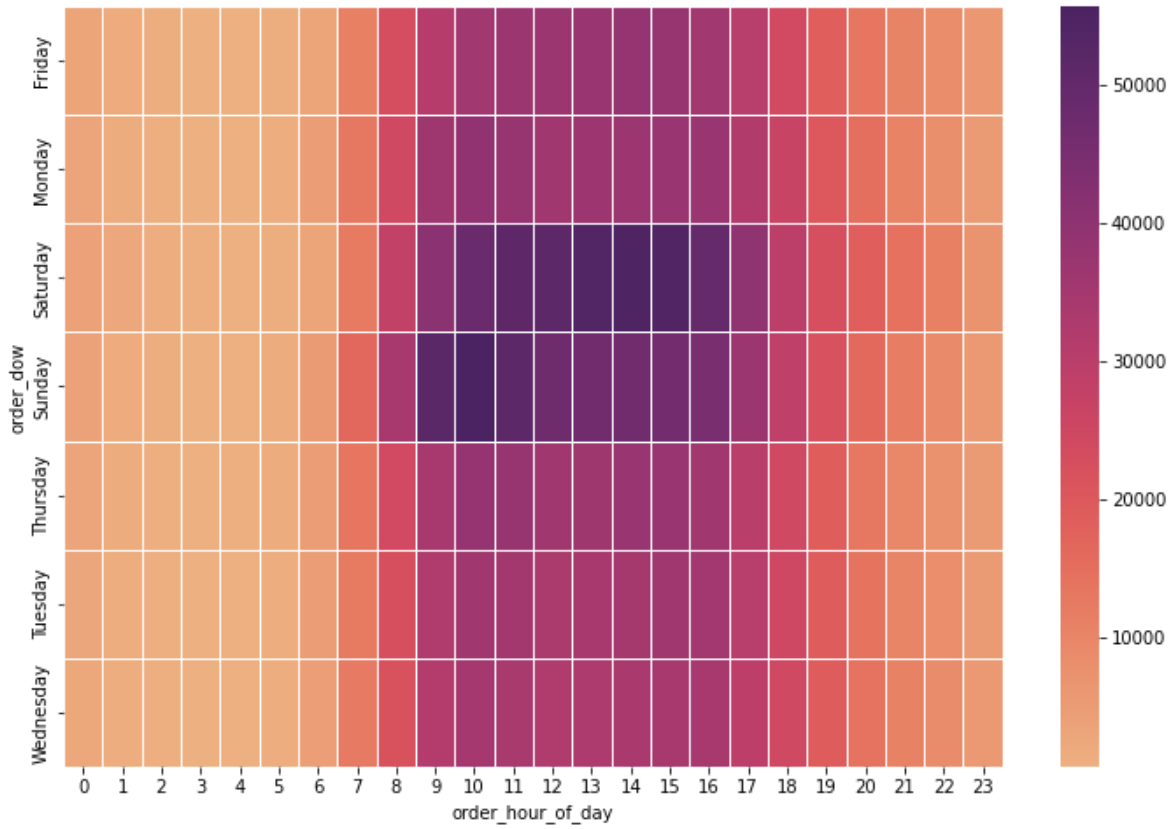



**Figure 18: 'Concentration of Orders based on time and day'**



Key findings:

- While most users have a total number of orders between 4 and 10, few users have ordered 100 times from Instacart. On average 17 orders per user.
- The maximum number of orders are placed during the weekend when people prepare for the week ahead. The busiest time of day is Sunday morning and Saturday afternoon.
- Most orders are placed after 30 days followed by 7 days and another jump at 14 days. Hence, we can determine that users usually shop monthly, weekly, and biweekly. Please note that the number of `days_since_prior_order` is capped at 30 days.

**Product Analysis:**

Created a function 'popularity', to create a dataframe based on a column to determine its popularity and plot a bar graph.

**Figure 19: Popularity function**

```
[ ]  def popularity(column_name):

       ''' Creating a dataframe based on a column to determine its popularity and plotting a bar graph'''

       x = merged_orders.groupby(column_name)['order_id'].count().reset_index().sort_values(by='order_id', ascending = False)

       plt.figure(figsize=(22,8))
       ax = sns.barplot(column_name, 'order_id', data = x,color= c('kale_green'))
       plt.xlim(-1,15)
       plt.xticks(rotation=70)
       plt.xlabel(column_name)
       plt.ylabel('Count of Orders')

       return ax
```

**Figure 20: 'Number of products in an Order'**

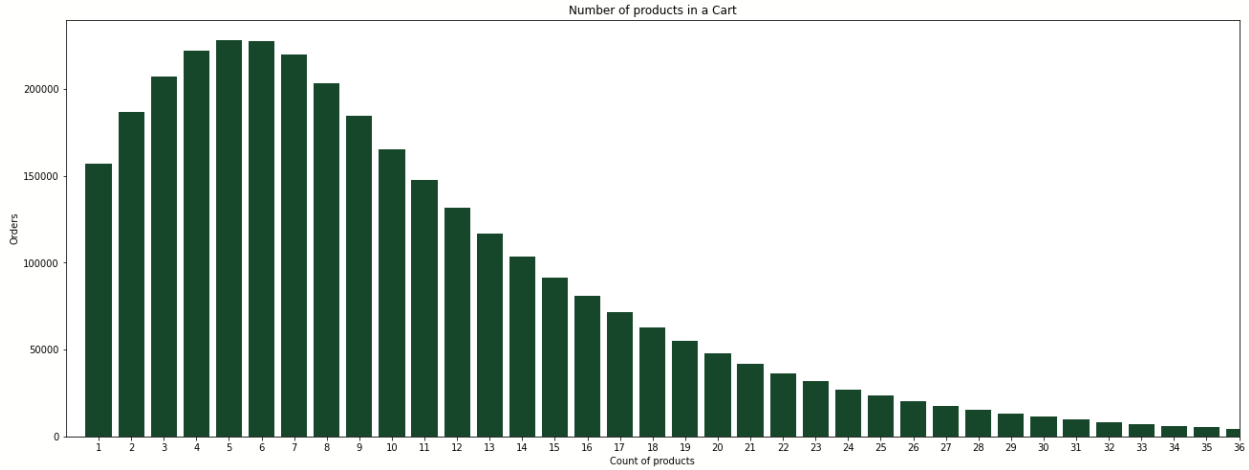



**Figure 21 : 'Top 15 Popular Products'**

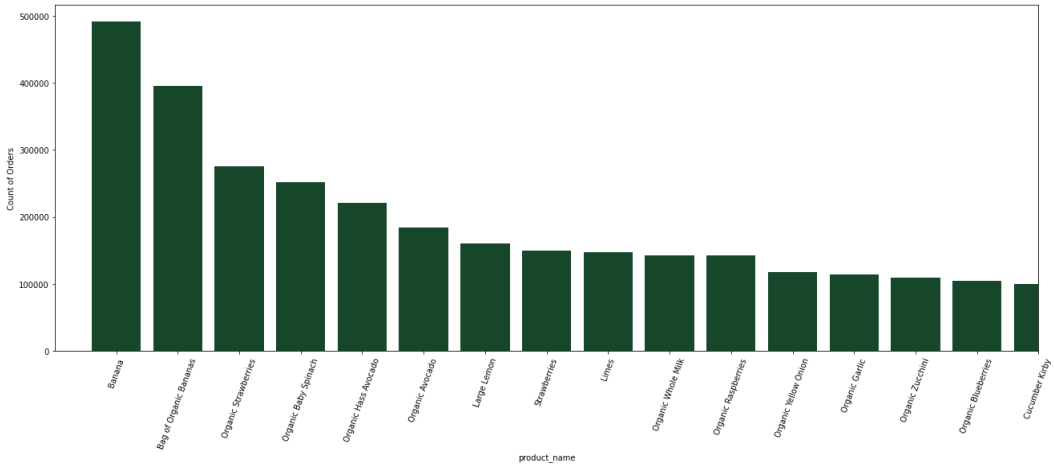



**Figure 22: Top 15 Popular Department**

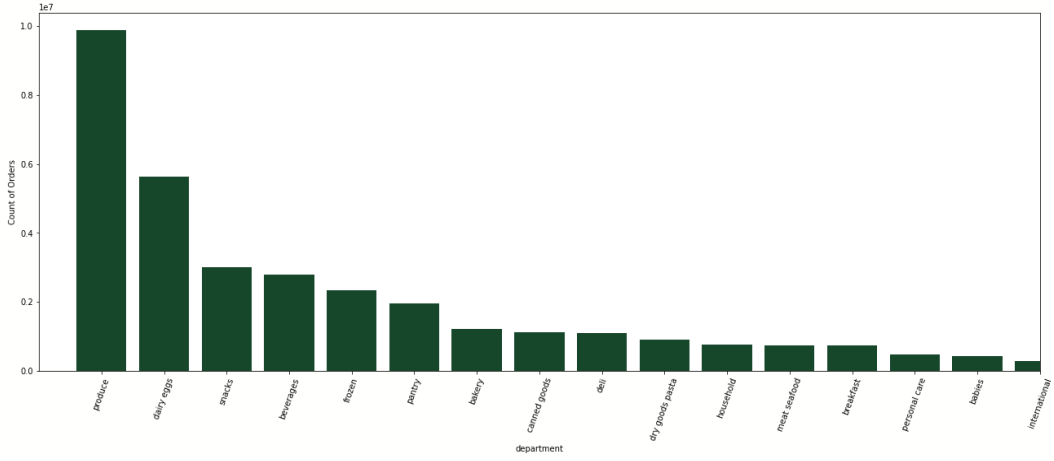


**Figure 23: Top 15 Popular aisle**

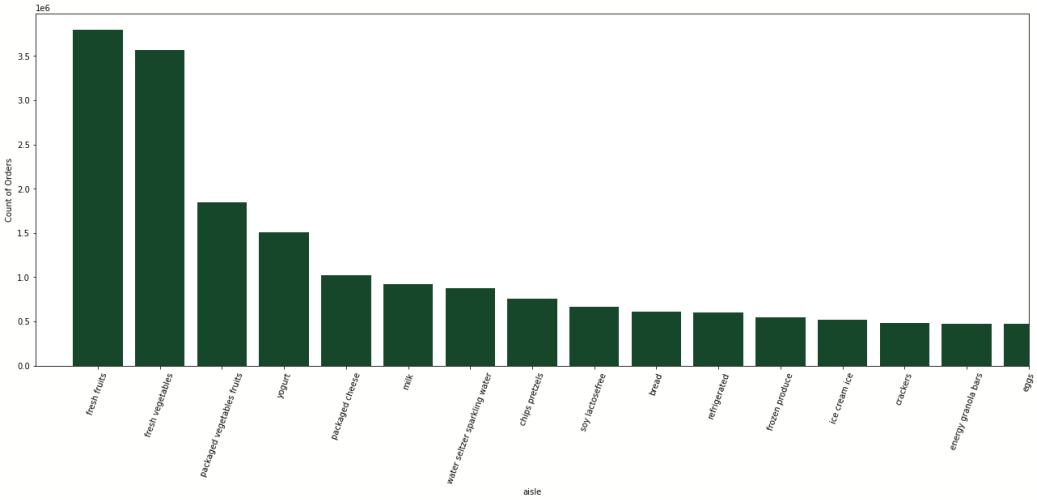
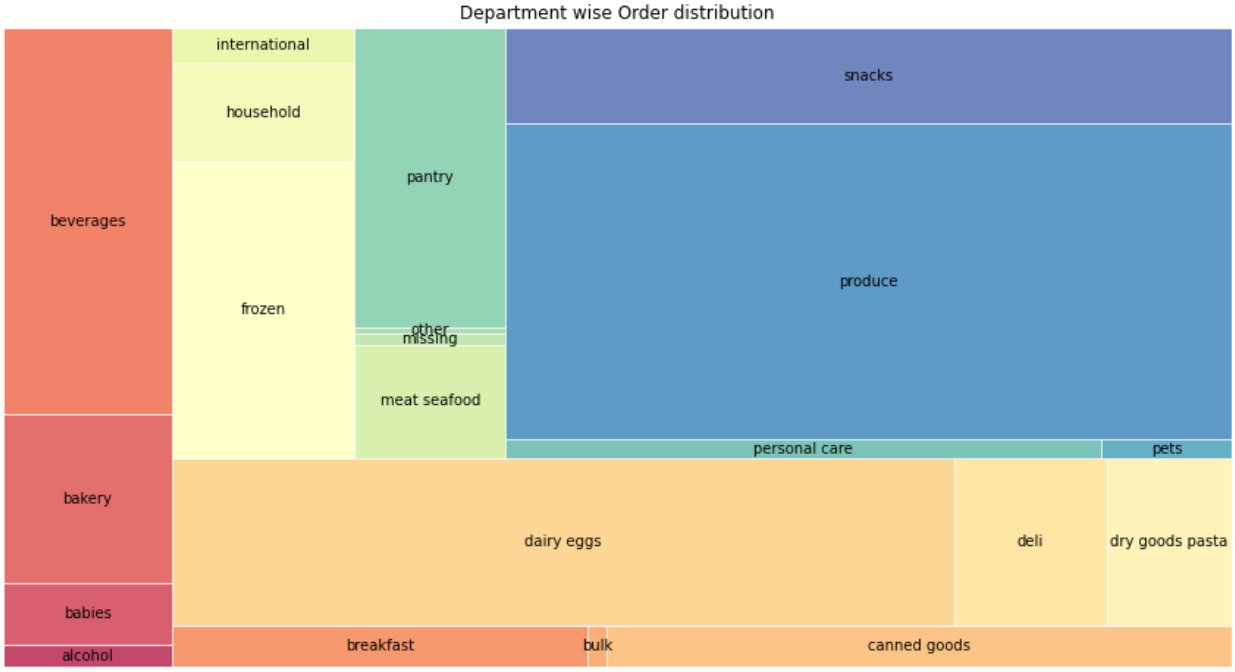


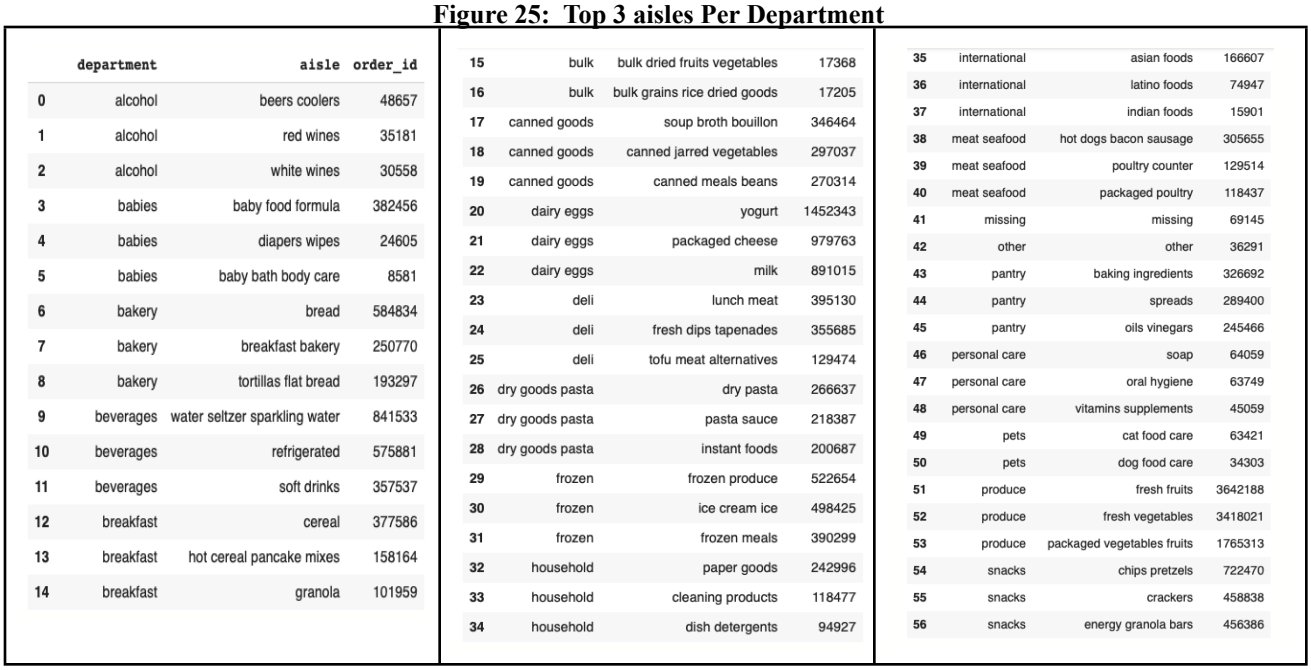

**Figure 24: Number of Orders Per Department**




Finding top 3 aisles in each department

```
[ ] best_selling_aisles = merged_orders.groupby(["department", "aisle"])["order_id"].aggregate('count').reset_index()

    #Top 3 aisles in each department
    best_selling_aisles.groupby(["department"]).apply(lambda x: x.sort_values(["order_id"], ascending = False).head(3)).reset_index(drop=True)
```

**Figure 25: Top 3 aisles Per Department**



| | department | aisle | order_id |
|---|---|---|---|
| 0 | alcohol | beers coolers | 48657 |
| 1 | alcohol | red wines | 35181 |
| 2 | alcohol | white wines | 30558 |
| 3 | babies | baby food formula | 382456 |
| 4 | babies | diapers wipes | 24605 |
| 5 | babies | baby bath body care | 8581 |
| 6 | bakery | bread | 584834 |
| 7 | bakery | breakfast bakery | 250770 |
| 8 | bakery | tortillas flat bread | 193297 |
| 9 | beverages | water seltzer sparkling water | 841533 |
| 10 | beverages | refrigerated | 575881 |
| 11 | beverages | soft drinks | 357537 |
| 12 | breakfast | cereal | 377586 |
| 13 | breakfast | hot cereal pancake mixes | 158164 |
| 14 | breakfast | granola | 101959 |
| 15 | bulk | bulk dried fruits vegetables | 17368 |
| 16 | bulk | bulk grains rice dried goods | 17205 |
| 17 | canned goods | soup broth bouillon | 346464 |
| 18 | canned goods | canned jarred vegetables | 297037 |
| 19 | canned goods | canned meals beans | 270314 |
| 20 | dairy eggs | yogurt | 1452343 |
| 21 | dairy eggs | packaged cheese | 979763 |
| 22 | dairy eggs | milk | 891015 |
| 23 | deli | lunch meat | 395130 |
| 24 | deli | fresh dips tapenades | 355685 |
| 25 | deli | tofu meat alternatives | 129474 |
| 26 | dry goods pasta | dry pasta | 266637 |
| 27 | dry goods pasta | pasta sauce | 218387 |
| 28 | dry goods pasta | instant foods | 200687 |
| 29 | frozen | frozen produce | 522654 |
| 30 | frozen | ice cream ice | 498425 |
| 31 | frozen | frozen meals | 390299 |
| 32 | household | paper goods | 242996 |
| 33 | household | cleaning products | 118477 |
| 34 | household | dish detergents | 94927 |
| 35 | international | asian foods | 166607 |
| 36 | international | latino foods | 74947 |
| 37 | international | indian foods | 15901 |
| 38 | meat seafood | hot dogs bacon sausage | 305655 |
| 39 | meat seafood | poultry counter | 129514 |
| 40 | meat seafood | packaged poultry | 118437 |
| 41 | missing | missing | 69145 |
| 42 | other | other | 36291 |
| 43 | pantry | baking ingredients | 326692 |
| 44 | pantry | spreads | 289400 |
| 45 | pantry | oils vinegars | 245466 |
| 46 | personal care | soap | 64059 |
| 47 | personal care | oral hygiene | 63749 |
| 48 | personal care | vitamins supplements | 45059 |
| 49 | pets | cat food care | 63421 |
| 50 | pets | dog food care | 34303 |
| 51 | produce | fresh fruits | 3642188 |
| 52 | produce | fresh vegetables | 3418021 |
| 53 | produce | packaged vegetables fruits | 1765313 |
| 54 | snacks | chips pretzels | 722470 |
| 55 | snacks | crackers | 458838 |
| 56 | snacks | energy granola bars | 456386 |

Key Findings:

- Most users buy 1-10 numbers of products in an order. Few users buy more than 30 products per order and a minor segment of users buy more than 50 products. The highest number of products in an order is 145.
- Fruit & vegetables are the most bought products by users. Interestingly, 11 out of the top 15 products are organic. People seem to prefer organic produce over normal produce. We will further explore the 'organic' section to determine people's preference for organic products.
- Most ordered products are from fresh produce, dairy eggs, snacks, beverages & frozen department. This is expected as these grocery items are perishable and consumed quickly.
- Interestingly, people consume more cheese than milk. In the international department, Asian food is most famous followed by Latino & Indian.

## Reorder Ratio:

**Figure 26: Reorder ratio of Department**

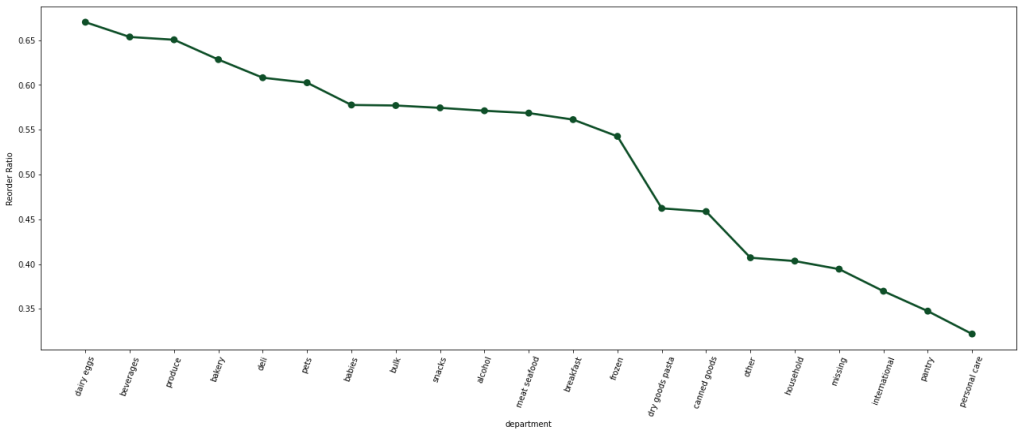



**Figure 27: Reorder ratio of aisle**

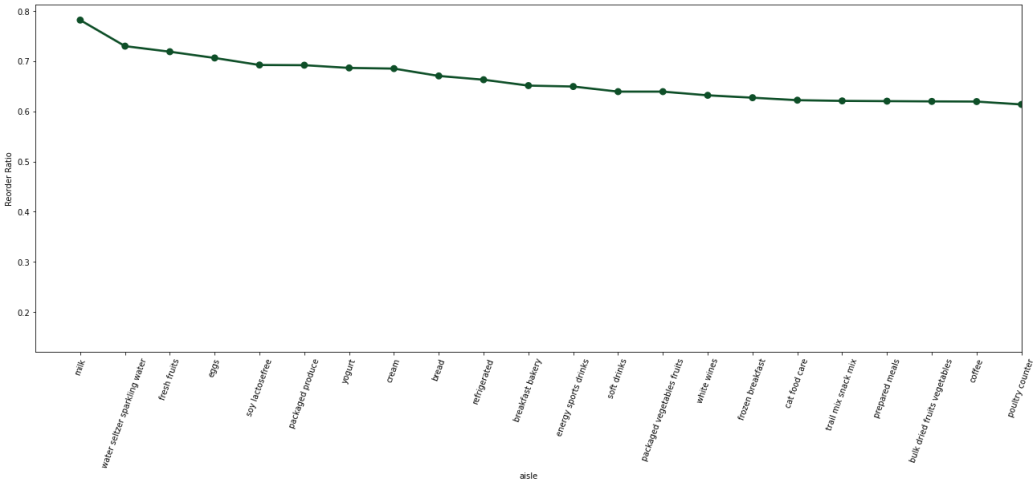



**Figure 28: Reorder ratio of product's 'add_to_cart_order'**

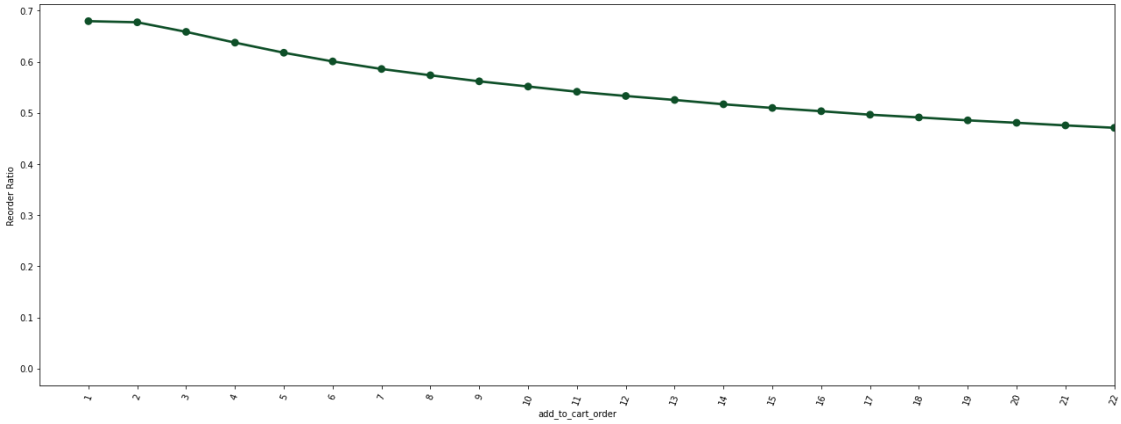


**Figure 29: Reorder ratio of Day of the week**

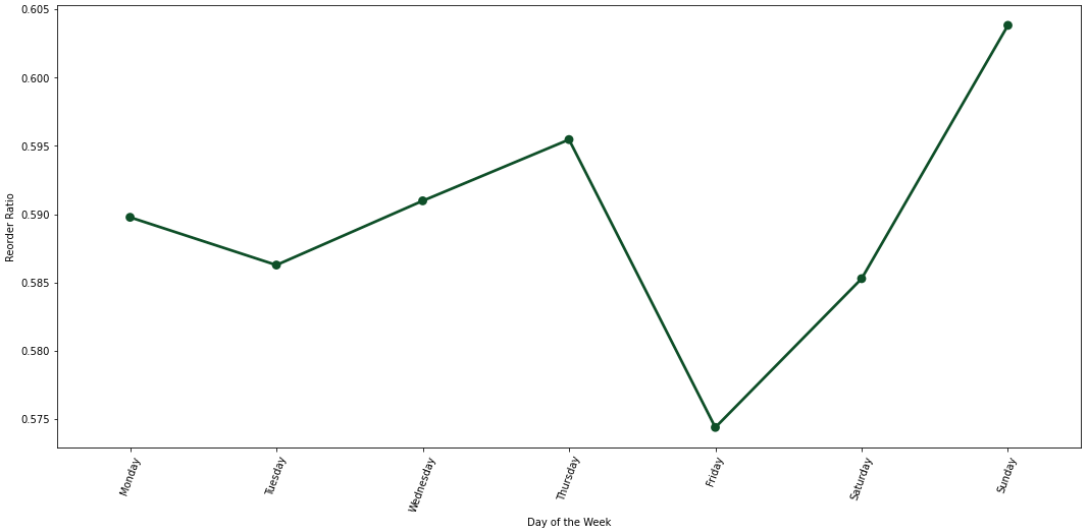
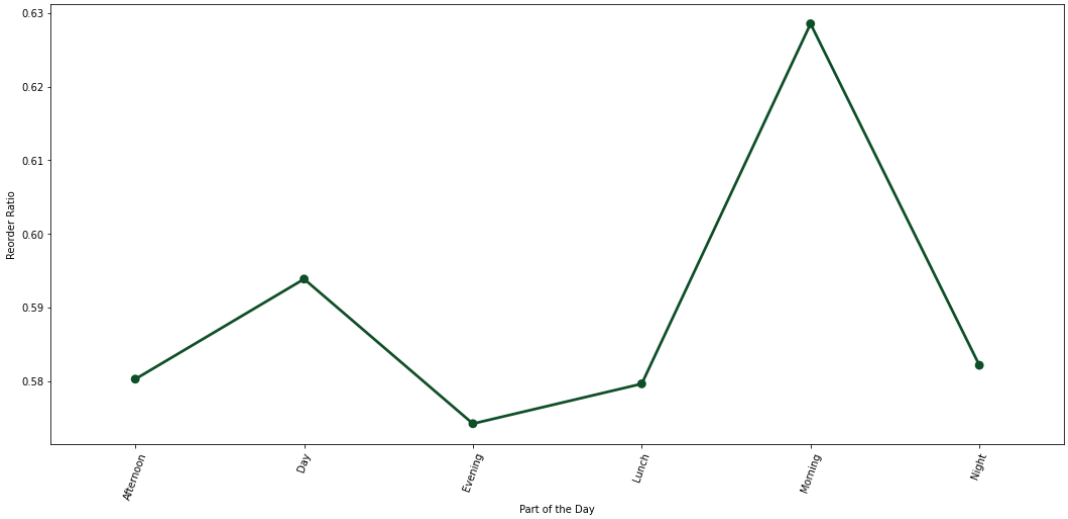



**Figure 30: Reorder ratio of Part of the week**




Key Findings:

- Reorder ratio of dairy-egg, beverage and produce department is more than 60%.
- A user usually adds frequently reordered products at the beginning of the cart. Hence, products added in the beginning of the order are the most reordered products.
- Sunday has the highest reorder ratio, followed by thursday.

6% of the non-first orders have no reordered products and 23% of the non-first orders have all the reordered products.

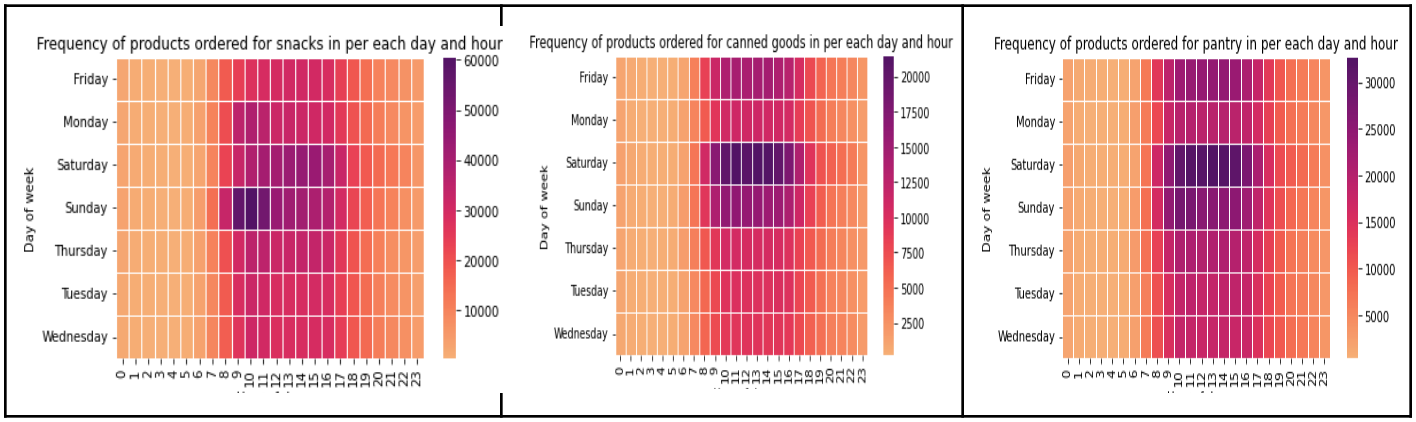
**Figure 31: Reorders in Non-first orders**

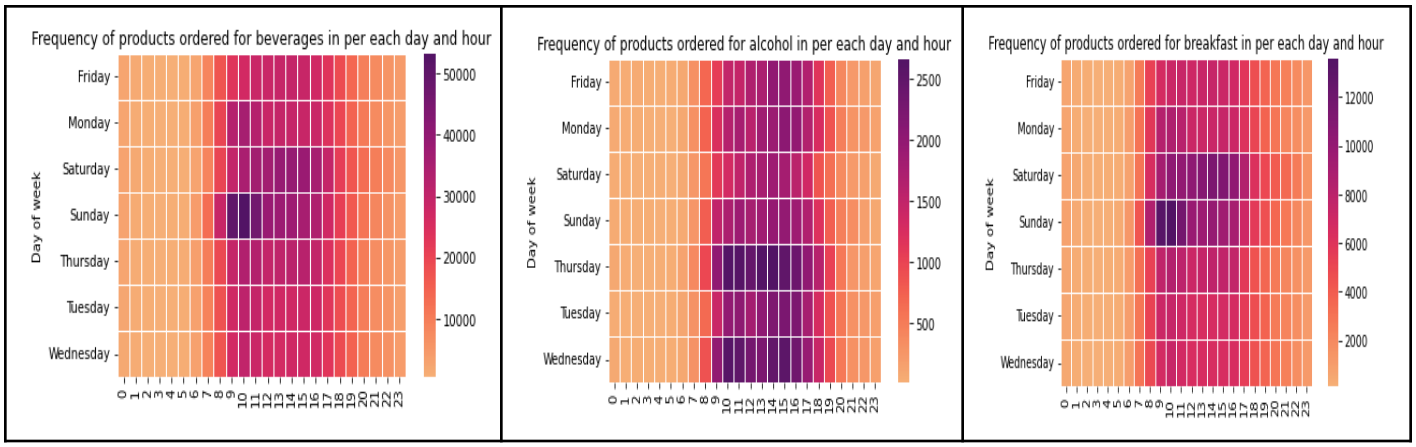
```
[ ]  reordered_prod = dataset[dataset['order_number'] != 1].groupby(['order_id'])['reordered'].mean().round(2).reset_index()

     print('Percentage of non-first order with no reordered product is ',
           round((reordered_prod[reordered_prod['reordered'] == 0.00]['order_id'].count())/reordered_prod['order_id'].count(),2))

     print('Percentage of non-first order with all reordered product is',
           round((reordered_prod[reordered_prod['reordered'] == 1]['order_id'].count())/reordered_prod['order_id'].count(),2))

     Percentage of non-first order with no reordered product is  0.06
     Percentage of non-first order with all reordered product is 0.23
```

Now, we will look at how department wise orders differ on the basis of day and time of the day using the below function.

**Figure 32: Order time and day based on department**

```
[ ]  def dept_name(x):

       ''' Create a subset of department data, groups by order_dow & hour_of_day and
       then visualises a heatmap to reflect the density of orders based on days and hour '''

       order_dpt = dataset[dataset['department']== x]
       group = order_dpt.groupby(['order_dow', 'order_hour_of_day']).agg('count').reset_index()
       heatmap = group.pivot('order_dow', 'order_hour_of_day', 'order_id')

       sns.heatmap(data=heatmap, cmap="flare", linewidths=.8)
       plt.title('Frequency of products ordered for '+x+' in per each day and hour')
       plt.ylabel('Day of week')
       plt.xlabel('Hour of day')
       plt.show()
```






Except for alcohol, which is mostly ordered on weekdays: Tuesday, Wednesday & Thursday, the rest of the products are mostly ordered during the weekend. Staples like canned foods, frozen produce, pantry, etc. are more ordered on Saturday while snacks, beverages, and breakfast are mostly ordered on Sunday.

Organic Products:

**Figure 33: Organic Products**

```
[ ] print('Count of Organic products:', products[products['product_name'].str.contains('Organic')]['product_id'].count())
    print('Percentage of Organic products:',
          (100 * products[products['product_name'].str.contains('Organic')]['product_id'].count())/products['product_id'].count())

    Count of Organic products: 5035
    Percentage of Organic products: 10.133231363709548

[ ] print('Percentage of orders that contains Organic products:',
          (100 * dataset[dataset['product_name'].str.contains('Organic')]['order_id'].count())/dataset['order_id'].count())

    Percentage of orders that contains Organic products: 31.603371337220697

[ ] print('Reorder ratio of organic products is : ',dataset[dataset['product_name'].str.contains('Organic')]['reordered'].mean())

    Reorder ratio of organic products is :  0.6349255716269192
```

On the basis of the above analysis, it seems that people are more inclined towards organics produce. While only 10% of the products are organic, around 31% of the orders contain organic products and the reorder ratio is also quite high for organic products.

### 3.3 Summary

Exploratory analysis of the data provides a detailed analysis of customers' buying patterns. A regular customer shops around 17 times from Instacart. Usually, customers have 10-15 numbers of products in an order. Groceries are purchased on the weekend for the week ahead. Staples have a higher reorder ratio and customers' preference is shifted towards organic produce. While exploratory data analysis gives a global overview of our customer base, there are techniques in machine learning which can determine customer behavior and segment according to the similarity between the group. These techniques will help us create and explore the personas of target customers.

# 5.0 CUSTOMER SEGMENTATION

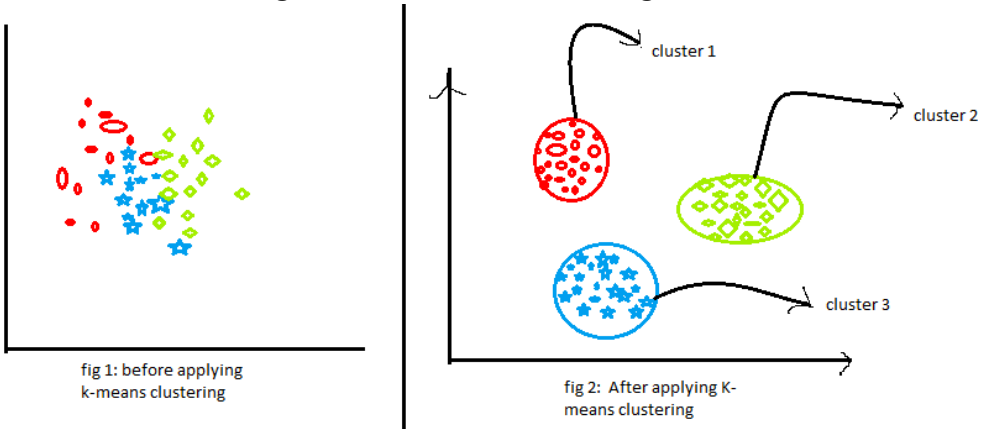
Customer segmentation is the process of clustering or segmenting customers based on shared behavior or traits. Clustering is a method of dividing data points in such a way that data points with the same traits are closer to each other than those in other groups. Some popular methodologies of clustering are:

- Connectivity model: This model is based on the distance between the data points. So, data points closer to each other will exhibit more similarity. It follows two approaches to clustering data points. In the first approach, all data points are classified into separate clustering and then aggregated as the distance decreases. The second approach is opposite it, in which all data points are classified in one cluster and then segregated as the distance increases. Hierarchical clustering is based on this model.
- Centroid model: In this model, the similarity between data points is derived based on the distance between data points and the centroid of clusters. K-mean clustering is based on the centroid model.
- Distribution model: In this model, data points are divided based on the probability of belonging to the same distribution (eg. Normal, Gaussian). This model suffers from overfitting.
- Density model: This model assigns data space based on the density of data points in the data space. DBSCAN and OPTICS are based on density models (Kaushik, 2020).

K-means clustering is one the simplest yet most effective clustering techniques with the capability of handling large datasets better than other algorithms. K-mean clustering is an unsupervised learning technique. In unsupervised learning techniques, there is no target variable to predict only independent

input variables. ‘K’ is the number of centroids and ‘means’ is the centroid or averaging of the data points.

**Figure 34: How K-mean clustering works**




Source:https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/

For k-mean clustering, the number of centroids needs to be predefined. We will use ‘interia’ to determine the optimum number of clusters. Interia measures the distance between data points and centroid, squares that distance, and then aggregates these squares across a cluster. Low interia and the low number of clusters is the sign of a good model, however as the number of clusters increases, interia decreases. In such a scenario, the elbow method is used in which the optimum number of k is the point when the decrease in inertia becomes slow.

**Model**

Libraries used: Scikit Learn (KMeans, PCA)

In the current dataset, We can cluster users based on aisle, department, or products. However, we have more than 40,000 products, due to which our model will suffer from the curse of dimensionality. Department will give us a very broad perspective, so we can narrow down our clusters using aisles.

Data preparation is the first step to creating a suitable dataset as an input. ‘Order_prior’ tables are merged with ‘products’ and ‘aisle’, which is further cross-tabbed to get the number of orders for each aisle. Since some of the aisle have a larger number of orders than the other, we will normalize the data.

**Figure 35: Users dataframe**

```
#merging datasets to get user_id, order_id & aisle columns

users = order_prior.merge(orders, on = 'order_id', how = 'left').merge(products,
            on = 'product_id', how = 'left').merge(aisle, on = 'aisle_id', how = 'left')

users = users[['user_id', 'order_id', 'aisle', 'reordered']]

users
```

| | user_id | order_id | aisle | reordered |
|---|---|---|---|---|
| 0 | 202279 | 2 | eggs | 1 |
| 1 | 202279 | 2 | fresh vegetables | 1 |
| 2 | 202279 | 2 | spices seasonings | 0 |

**Figure 36: Crosstab dataframe**

```
[ ] #creating crosstab dataframe of all users to get number of orders for each aisle and then normalising the data
    df = pd.crosstab(users['user_id'], users['aisle'])
    df  = df.div(df.sum(axis=1), axis=0)
    df.head()
```

| aisle | air fresheners candles | asian foods | baby accessories | baby bath body care | baby food formula | bakery desserts | baking ingredients | baking supplies decor | beauty | beers coolers | ... | spreads | tea |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | |
| 1 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | 0.0 | 0.000000 | 0.0 | 0.0 | 0.0 | ... | 0.016949 | 0.000000 |
| 2 | 0.0 | 0.015385 | 0.0 | 0.0 | 0.0 | 0.0 | 0.010256 | 0.0 | 0.0 | 0.0 | ... | 0.015385 | 0.005128 |

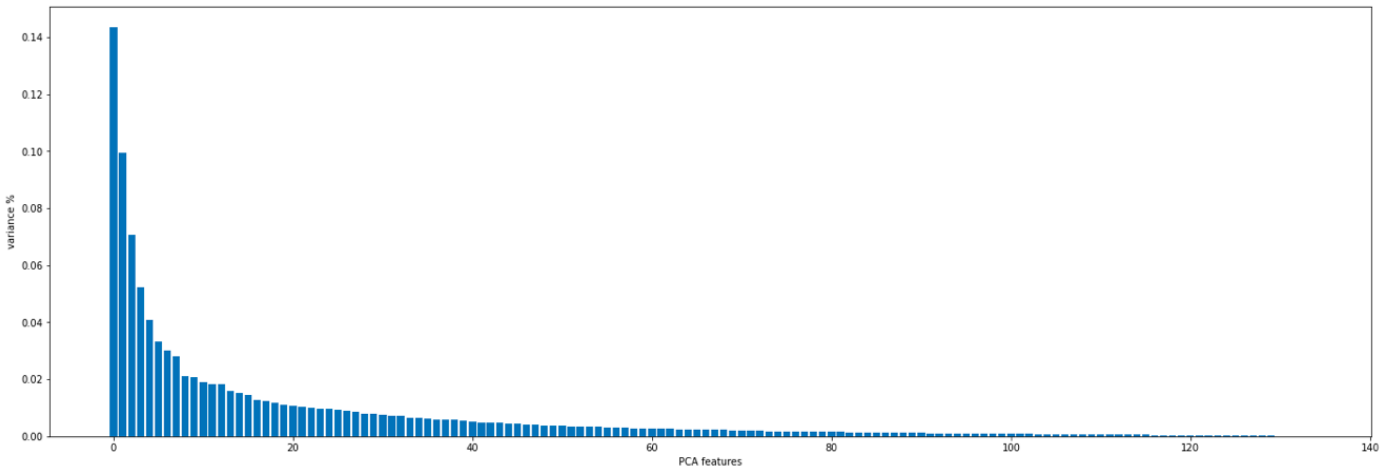
K-mean clustering is based on the distance metric and distance measures do not work well in high-dimensional spaces. Because as the dimensions increase, the distance between any two data points decreases, and the difference between the maximum and minimum distance start converging. Due to this, K-means fails to cluster the data points efficiently. In such cases, Principal component analysis (PCA) is used as a dimensionality reduction technique.

PCA aims to reduce the dimensions while preserving the variance. In PCA, new variables are created which are the weighted average of the original variable

In this dataset, out of 134 components, 10 components can explain 54% of the variance.

**Figure 37: PCA Components**

```
pca = PCA()
pca.fit(df)

plt.figure(figsize=(24,8))
features = range(pca.n_components_)
plt.bar(features, pca.explained_variance_ratio_)
plt.xlabel('PCA features')
plt.ylabel('variance %');
```




```
[12] pca = PCA(n_components= 10)
     pca.fit(df)

     pca.explained_variance_ratio_.sum()

     0.5390335665563243
```

Next, we will use the elbow method to determine the optimum number of clusters. Below plot shows that after 5 numbers of k's the decrease in the inertia is slow. Hence we will use 5 as the optimum number of clusters for K-mean clustering and plot the clusters with the first two components.

**Figure 38: Elbow Plot for number of 'k'**

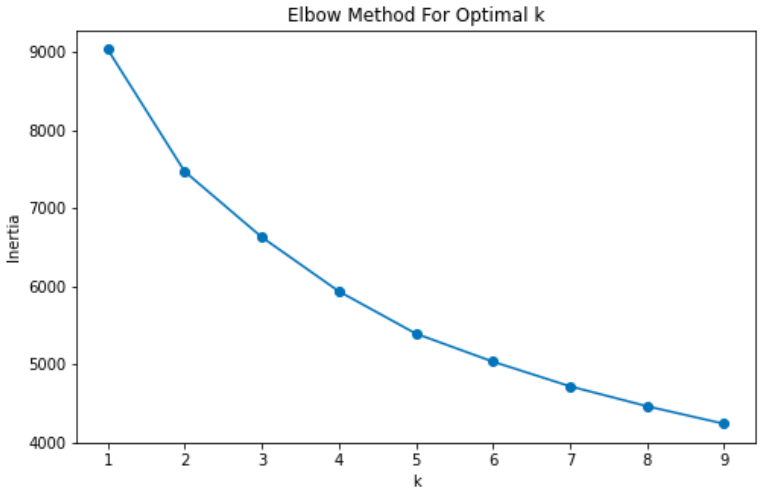
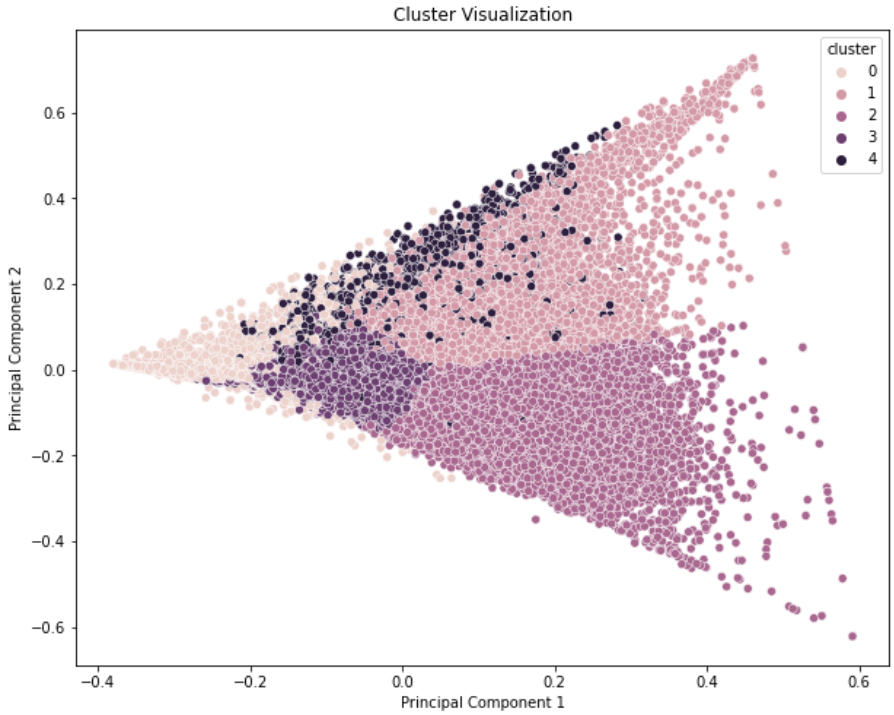
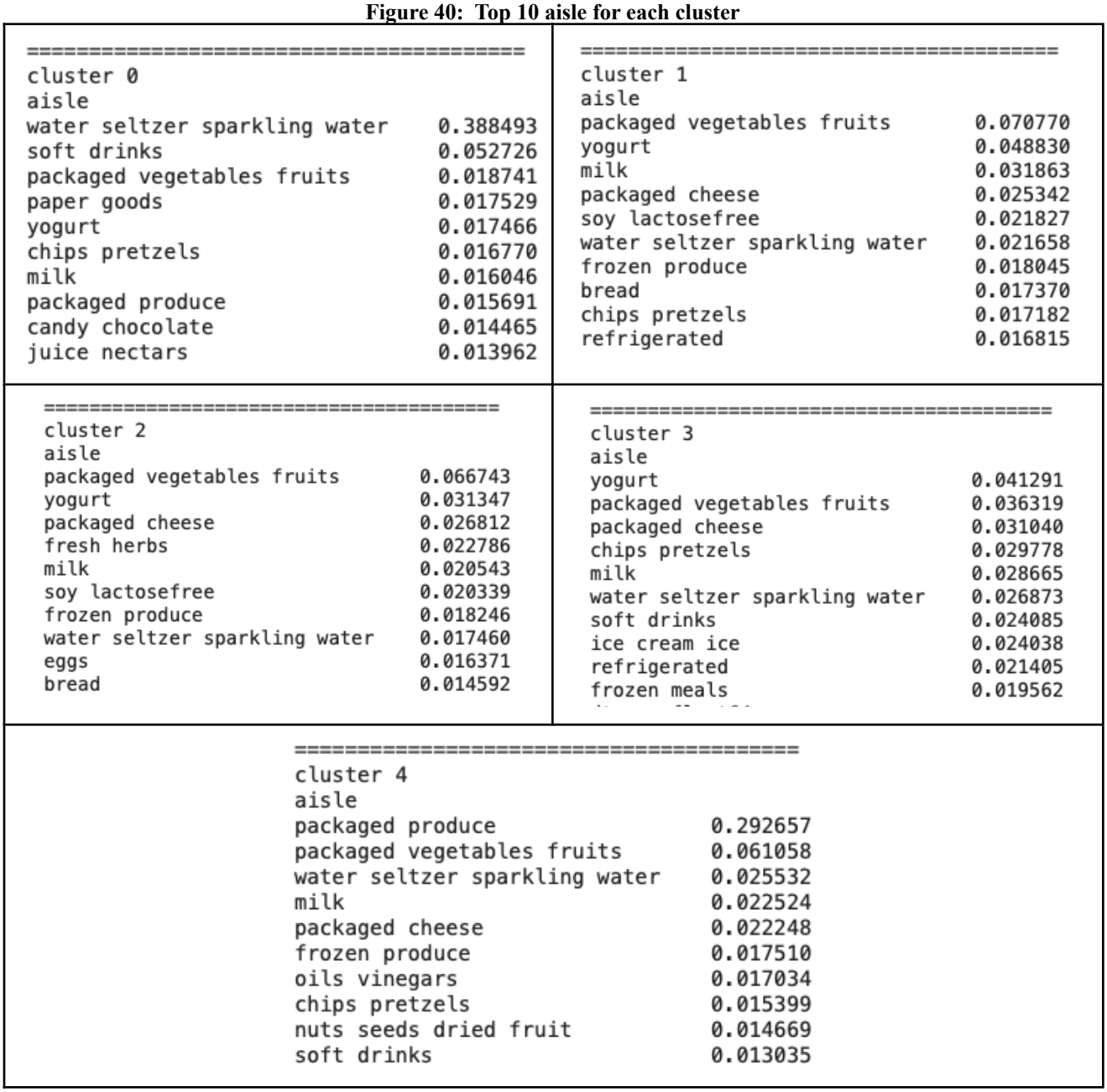



**Figure 39: Clusters Visualization**




In the next step, we have created the top 10 aisles for each cluster. Please note, for this analysis 'fresh fruit' and 'fresh vegetable' aisles are removed since they are uniform across all clusters.

**Figure 40: Top 10 aisle for each cluster**



```
========================================
cluster 0
aisle
water seltzer sparkling water    0.388493
soft drinks                      0.052726
packaged vegetables fruits       0.018741
paper goods                      0.017529
yogurt                           0.017466
chips pretzels                   0.016770
milk                             0.016046
packaged produce                 0.015691
candy chocolate                  0.014465
juice nectars                    0.013962
```

```
========================================
cluster 1
aisle
packaged vegetables fruits       0.070770
yogurt                           0.048830
milk                             0.031863
packaged cheese                  0.025342
soy lactosefree                  0.021827
water seltzer sparkling water    0.021658
frozen produce                   0.018045
bread                            0.017370
chips pretzels                   0.017182
refrigerated                     0.016815
```

```
========================================
cluster 2
aisle
packaged vegetables fruits       0.066743
yogurt                           0.031347
packaged cheese                  0.026812
fresh herbs                      0.022786
milk                             0.020543
soy lactosefree                  0.020339
frozen produce                   0.018246
water seltzer sparkling water    0.017460
eggs                             0.016371
bread                            0.014592
```

```
========================================
cluster 3
aisle
yogurt                           0.041291
packaged vegetables fruits       0.036319
packaged cheese                  0.031040
chips pretzels                   0.029778
milk                             0.028665
water seltzer sparkling water    0.026873
soft drinks                      0.024085
ice cream ice                    0.024038
refrigerated                     0.021405
frozen meals                     0.019562
```

```
========================================
cluster 4
aisle
packaged produce                 0.292657
packaged vegetables fruits       0.061058
water seltzer sparkling water    0.025532
milk                             0.022524
packaged cheese                  0.022248
frozen produce                   0.017510
oils vinegars                    0.017034
chips pretzels                   0.015399
nuts seeds dried fruit           0.014669
soft drinks                      0.013035
```

In the next step, we created a new dataframe by merging the cluster information with the ‘orders’ and ‘order_prior’ table to get number of ‘total_products’, ‘total_orders’, ‘average_cart_size’, ‘average_order_dow’ and ‘average_order_hour_of_day’ for each cluster.

**Figure 41: Order summary for each cluster**

| clusters | total_products | total_orders | average_cart_size | order_dow | order_hour_of_day |
|---|---|---|---|---|---|
| 0 | 62.996493 | 12.815581 | 4.372913 | 2.831457 | 13.047259 |
| 1 | 175.244094 | 17.782196 | 9.553134 | 2.666596 | 13.488057 |
| 2 | 180.987448 | 16.048776 | 11.444275 | 2.653142 | 13.659051 |
| 3 | 149.568301 | 14.755451 | 10.000485 | 2.825759 | 13.568805 |
| 4 | 65.875708 | 14.239275 | 4.553286 | 2.835954 | 13.108945 |

Key Findings:

- Cluster 0 has a strong preference for drinks such as seltzer, sparkling water, soft drinks, juices, and also candies & chocolate. They seem to be in their early 20s, living by themselves, and usually order snacks and drinks, due to which their average number of orders and cart size is less.
- Cluster 1 & Cluster 2 buys regular grocery items such as packaged vegetables, fruits, bread, milk, egg & yogurt. Cluster 2 differs from cluster 1 in terms of its preference for fresh herbs. They seem to be families with high order frequency because they buy weekly staples like fruits, vegetables, eggs, yogurt milk, etc.
- Cluster 3 orders refrigerated items, frozen meals, and ice cream aside from staples. They have a high average cart size but a low average number of orders because they usually buy frozen products due to which they do not have to buy that often but when they do buy, they order a large number of products.
- Cluster 4 mostly orders frozen produce aside from packaged vegetables & fruits. Their basket also contains oils, vinegar, and nuts, seeds & dried fruits. They also have a low average number of orders because they buy long shelf life products such as seltzers, oils, vinegar, nuts, seeds, dried fruits, and frozen produce.

# 6.0 ASSOCIATION RULE

Market basket analysis or MBA is a data mining technique that is used to determine the purchase habits of consumers. The technique aims to uncover the set of products that are frequently brought together.

The Apriori algorithm was proposed by R. Agrawal and R. Srikant in 1994 to find the frequent item sets(“Apriori Algorithm,” 2022). It is called ‘apriori’ because it uses prior knowledge of itemsets and their properties. Apriori algorithm operates on a transaction database. It uses a ‘bottom-up’ approach, in which frequent itemsets are extended one item at a time and when no further extensions are available, the algorithm terminates. It uses an iterative approach to get k+1 itemsets from k itemsets. If the k itemset is not greater than the specified threshold then it is known as the candidate itemset. The frequent itemsets generated can further be used for association rule mining.

Association rules are 'if-then' statements that reflect the probability of relationships among the dataset. It is used in Market basket analysis to determine the relationship between products. This association between products is used by retailers in various marketing decisions. Association rule helps in developing the following strategies:

- Product placement
- Personalized push notifications
- Catalog design
- Cross-selling & Up-selling

Strategic and mindful placement of products in the aisle not only saves consumers time but also encourages consumers to buy related products. For example, consumers buying cereal are more likely to buy milk if the milk aisle is closer to the cereal section.

The 'if' component of the association rule is known as 'antecedent' and the 'then' component is 'consequent'. These components are disjoint, they just reflect co-occurrence and not causality.

The strength of the association rule is measured by support, confidence, and lift ratio.

- Support: Support measures the fraction of the product's occurrence. In simple terms, it reflects the popularity of the product. The number of times a product occurs in transactions. For example, `milk` will occur in more transactions than `shaving cream. Hence `milk` generally has higher support than `shaving cream.
- Confidence: Confidence is the ratio of the number of transactions that contain both antecedent & consequent to the number of transactions that contain all antecedent itemsets. As the name suggests, confidence reflects the reliability of the rule and conditional probability that the consequent will occur given the occurrence of the antecedent.
- Lift: Lift is the ratio of confidence of the rule to the expected confidence. In simple terms, it measures the importance and quality of the rule.

Limitation of Apriori algorithm:

It can be slow, Apriori iterates through the dataset multiple times to find the candidate itemset. It holds a vast number of candidate sets with frequent itemsets, low minimum support, or large itemsets, which results in a costly waste of time. With a large number of transactions and limited memory, Apriori can be very slow and inefficient.

**Model**

Libraries used: MLxtend (apriori, association rule)

Using apriori on 40,000 products would have been very slow due to limited memory. Hence, created a subset of top 200 selling products. Pivoted the dataframe with product names as column and order_id as index and applied apriori algorithm with min_support of 0.03. Algorithm generated 371 sets of frequent itemsets.

**Figure 42: Dataset creation of top 200 products**

```
#calculating top 200 most bought products

top_200_products = order_prior.groupby('product_id')['order_id'].count().sort_values(ascending = False)[:200].reset_index()

#creating a dataframe of only top 200 products

df = order_prior[order_prior.product_id.isin(top_200_products.product_id.to_list())]

#merging df with products data on product_id to get product name
df = df.merge(products, on = 'product_id', how = 'left')

#keeping only the relevant columns
df= df[['order_id','product_name','reordered']].set_index('order_id')
```

```
[ ] x = df.pivot_table(columns='product_name', values='reordered', index='order_id').reset_index().fillna(0).set_index('order_id')

    #converting data type from float to int
    x = x.astype('int32')
```

We will derive the association rule based on the lift. Lift is more important than confidence in the association rule because it reflects the relative strength of association between 2 products. It is the ratio of the change in the probability of item 1 with the knowledge that item 2 is present over the probability of the presence of item 1 without the knowledge of item 2's presence (Nandakumar,2020).

For example:

The probability of milk in the cart with the knowledge of the presence of a toothbrush is = 10/(10+4) = .7

If the probability of milk in the cart without the knowledge of a toothbrush is 80/100 = .8We can see above that the knowledge of toothbrushes reduces the probability of milk from 0.8 to 07. Hence the lift here will be 0.7/.8 = .87 and lift less than 1 shows less association between products (Nandakumar, 2022).

We will set min_threshold to be 1 because a lift more than 1 means that products are likely to be bought together where a lift of 1 means that products have no association between them and lift of fewer than 1 means that products are not likely to be bought together.

**Figure 43: Order summary for each cluster**

```
rules = association_rules(frequent_sets, metric='lift', min_threshold=1)
rules.sort_values(by=['lift'], ascending=False)
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage |
|---|---|---|---|---|---|---|---|---|
| 197 | (Sparkling Water Grapefruit) | (Lime Sparkling Water) | 0.022917 | 0.013527 | 0.003886 | 0.169569 | 12.535775 | 0.003576 |
| 196 | (Lime Sparkling Water) | (Sparkling Water Grapefruit) | 0.013527 | 0.022917 | 0.003886 | 0.287278 | 12.535775 | 0.003576 |
| 275 | (Organic Yellow Onion) | (Organic Garlic) | 0.030969 | 0.029242 | 0.005464 | 0.176421 | 6.033198 | 0.004558 |
| 274 | (Organic Garlic) | (Organic Yellow Onion) | 0.029242 | 0.030969 | 0.005464 | 0.186840 | 6.033198 | 0.004558 |
| 202 | (Limes) | (Organic Cilantro) | 0.037508 | 0.016881 | 0.003669 | 0.097830 | 5.795179 | 0.003036 |

Next, a new column 'length' is created to show the number of items in the set. A function called 'threshold' is defined to filter the rules according to the desired threshold. If a threshold value is not defined, default values: support = .001, confidence = .005 and length = 1 will be taken.

**Figure 44: Length column**

```
[ ]  #adding length column to determine the number of items in the set
     rules['length'] = rules['antecedents'].apply(lambda x: len(x))
```

**Figure 45: Threshold function**

```
def threshold(support = .001,confidence=.005,lift=1.1, length = 1):

  ''' This function will take the given threshold or the default minimum threshold and generate the desired result'''

  return rules[ (rules['support'] >= support) &
       (rules['confidence'] >= confidence) &
       (rules['lift'] >= lift) &
       (rules['length'] == length)].sort_values(by = 'lift', ascending = False)
```

**Figure 46: Rules with 2 items and minimum confidence = .20**

```
threshold(length = 2, confidence =.2).sort_values(by=['confidence','lift'], ascending=[False, False])
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | length |
|---|---|---|---|---|---|---|---|---|---|---|
| 334 | (Organic Raspberries, Organic Hass Avocado) | (Bag of Organic Bananas) | 0.007615 | 0.123727 | 0.003447 | 0.452633 | 3.658309 | 0.002505 | 1.600887 | 2 |
| 340 | (Organic Strawberries, Organic Hass Avocado) | (Bag of Organic Bananas) | 0.011934 | 0.123727 | 0.004470 | 0.374553 | 3.027242 | 0.002993 | 1.401034 | 2 |
| 321 | (Organic Baby Spinach, Organic Hass Avocado) | (Bag of Organic Bananas) | 0.009896 | 0.123727 | 0.003565 | 0.360258 | 2.911707 | 0.002341 | 1.369728 | 2 |
| 346 | (Organic Raspberries, Organic Strawberries) | (Bag of Organic Bananas) | 0.009827 | 0.123727 | 0.003436 | 0.349607 | 2.825626 | 0.002220 | 1.347298 | 2 |
| 327 | (Organic Baby Spinach, Organic Strawberries) | (Bag of Organic Bananas) | 0.010427 | 0.123727 | 0.003056 | 0.293130 | 2.369159 | 0.001766 | 1.239652 | 2 |
| 332 | (Bag of Organic Bananas, Organic Raspberries) | (Organic Hass Avocado) | 0.012306 | 0.066632 | 0.003447 | 0.280099 | 4.203684 | 0.002627 | 1.296523 | 2 |
| 344 | (Bag of Organic Bananas, Organic Raspberries) | (Organic Strawberries) | 0.012306 | 0.080619 | 0.003436 | 0.279176 | 3.462899 | 0.002443 | 1.275459 | 2 |
| 320 | (Organic Baby Spinach, Bag of Organic Bananas) | (Organic Hass Avocado) | 0.014769 | 0.066632 | 0.003565 | 0.241395 | 3.622814 | 0.002581 | 1.230374 | 2 |
| 338 | (Bag of Organic Bananas, Organic Strawberries) | (Organic Hass Avocado) | 0.018520 | 0.066632 | 0.004470 | 0.241361 | 3.622307 | 0.003236 | 1.230319 | 2 |
| 339 | (Bag of Organic Bananas, Organic Hass Avocado) | (Organic Strawberries) | 0.019559 | 0.080619 | 0.004470 | 0.228530 | 2.834678 | 0.002893 | 1.191725 | 2 |
| 326 | (Organic Baby Spinach, Bag of Organic Bananas) | (Organic Strawberries) | 0.014769 | 0.080619 | 0.003056 | 0.206948 | 2.566977 | 0.001866 | 1.159294 | 2 |

**Figure 47: Rules with 1 items and minimum confidence = 30**

```
[ ] threshold(length = 1,  confidence = .30).sort_values(by=['confidence','lift'], ascending=[False, False])
```

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | length |
|---|---|---|---|---|---|---|---|---|---|---|
| **95** | (Bartlett Pears) | (Banana) | 0.009014 | 0.156115 | 0.003630 | 0.402737 | 2.579742 | 0.002223 | 1.412921 | 1 |
| **133** | (Organic Fuji Apple) | (Banana) | 0.024992 | 0.156115 | 0.009735 | 0.389525 | 2.495112 | 0.005833 | 1.382342 | 1 |
| **113** | (Honeycrisp Apple) | (Banana) | 0.022644 | 0.156115 | 0.008058 | 0.355858 | 2.279456 | 0.004523 | 1.310091 | 1 |
| **63** | (Organic Navel Orange) | (Bag of Organic Bananas) | 0.010705 | 0.123727 | 0.003774 | 0.352590 | 2.849735 | 0.002450 | 1.353506 | 1 |
| **109** | (Granny Smith Apples) | (Banana) | 0.009218 | 0.156115 | 0.003221 | 0.349365 | 2.237863 | 0.001781 | 1.297017 | 1 |
| **101** | (Broccoli Crown) | (Banana) | 0.010786 | 0.156115 | 0.003671 | 0.340378 | 2.180296 | 0.001987 | 1.279345 | 1 |
| **105** | (Cucumber Kirby) | (Banana) | 0.026363 | 0.156115 | 0.008748 | 0.331808 | 2.125404 | 0.004632 | 1.262938 | 1 |
| **41** | (Organic D'Anjou Pears) | (Bag of Organic Bananas) | 0.013230 | 0.123727 | 0.004193 | 0.316903 | 2.561303 | 0.002556 | 1.282795 | 1 |
| **59** | (Organic Large Extra Fancy Fuji Apple) | (Bag of Organic Bananas) | 0.022403 | 0.123727 | 0.007055 | 0.314907 | 2.545168 | 0.004283 | 1.279057 | 1 |
| **167** | (Seedless Red Grapes) | (Banana) | 0.021610 | 0.156115 | 0.006603 | 0.305562 | 1.957284 | 0.003230 | 1.215205 | 1 |

Bananas and Bag of Organic Bananas are natural consequent for fruits and vegetables. So, we followed the above steps and ran a new association rule without Bananas and Bag of Organic Bananas to check the association between products excluding these two products from the set.

**Figure 48: Top 20 rules without bananas or organic bag of banana**

| | antecedents | consequents | antecedent support | consequent support | support | confidence | lift | leverage | conviction | length |
|---|---|---|---|---|---|---|---|---|---|---|
| **27** | (Lime Sparkling Water) | (Sparkling Water Grapefruit) | 0.013527 | 0.022917 | 0.003886 | 0.287278 | 12.535775 | 0.003576 | 1.370918 | 1 |
| **112** | (Organic Lemon) | (Organic Hass Avocado) | 0.023709 | 0.066632 | 0.005860 | 0.247175 | 3.709565 | 0.004280 | 1.239821 | 1 |
| **133** | (Organic Raspberries) | (Organic Strawberries) | 0.041283 | 0.080619 | 0.009827 | 0.238035 | 2.952579 | 0.006499 | 1.206591 | 1 |
| **126** | (Organic Kiwi) | (Organic Strawberries) | 0.014007 | 0.080619 | 0.003253 | 0.232210 | 2.880332 | 0.002123 | 1.197438 | 1 |
| **90** | (Organic Blueberries) | (Organic Strawberries) | 0.024643 | 0.080619 | 0.005464 | 0.221719 | 2.750198 | 0.003477 | 1.181297 | 1 |
| **32** | (Organic Cilantro) | (Limes) | 0.016881 | 0.037508 | 0.003669 | 0.217363 | 5.795179 | 0.003036 | 1.229807 | 1 |
| **92** | (Organic Cucumber) | (Organic Hass Avocado) | 0.021952 | 0.066632 | 0.004737 | 0.215803 | 3.238743 | 0.003275 | 1.190222 | 1 |
| **138** | (Organic Whole String Cheese) | (Organic Strawberries) | 0.017875 | 0.080619 | 0.003786 | 0.211836 | 2.627615 | 0.002345 | 1.166485 | 1 |
| **148** | (Raspberries) | (Strawberries) | 0.015473 | 0.039087 | 0.003199 | 0.206748 | 5.289370 | 0.002594 | 1.211359 | 1 |
| **118** | (Organic Tomato Cluster) | (Organic Hass Avocado) | 0.016552 | 0.066632 | 0.003364 | 0.203237 | 3.050148 | 0.002261 | 1.171450 | 1 |
| **134** | (Organic Red Bell Pepper) | (Organic Strawberries) | 0.014870 | 0.080619 | 0.003020 | 0.203092 | 2.519151 | 0.001821 | 1.153685 | 1 |
| **94** | (Organic Cucumber) | (Organic Strawberries) | 0.021952 | 0.080619 | 0.004312 | 0.196428 | 2.436493 | 0.002542 | 1.144118 | 1 |
| **43** | (Michigan Organic Kale) | (Organic Baby Spinach) | 0.018345 | 0.073193 | 0.003589 | 0.195662 | 2.673226 | 0.002247 | 1.152260 | 1 |
| **77** | (Organic Small Bunch Celery) | (Organic Baby Spinach) | 0.016830 | 0.073193 | 0.003262 | 0.193852 | 2.648494 | 0.002031 | 1.149673 | 1 |
| **128** | (Organic Large Extra Fancy Fuji Apple) | (Organic Strawberries) | 0.022403 | 0.080619 | 0.004300 | 0.191920 | 2.380569 | 0.002493 | 1.137734 | 1 |
| **62** | (Organic Cucumber) | (Organic Baby Spinach) | 0.021952 | 0.073193 | 0.004127 | 0.187990 | 2.568402 | 0.002520 | 1.141373 | 1 |
| **105** | (Organic Garlic) | (Organic Yellow Onion) | 0.029242 | 0.030969 | 0.005464 | 0.186840 | 6.033198 | 0.004558 | 1.191685 | 1 |
| **115** | (Organic Raspberries) | (Organic Hass Avocado) | 0.041283 | 0.066632 | 0.007615 | 0.184462 | 2.768382 | 0.004864 | 1.144482 | 1 |
| **84** | (Organic Zucchini) | (Organic Baby Spinach) | 0.028263 | 0.073193 | 0.005189 | 0.183607 | 2.508526 | 0.003121 | 1.135246 | 1 |
| **3** | (Apple Honeycrisp Organic) | (Organic Hass Avocado) | 0.024482 | 0.066632 | 0.004436 | 0.181203 | 2.719465 | 0.002805 | 1.139926 | 1 |

Key Findings: A prominent pattern in the above rule is the lack of association between organic and non-organic products. This shows that customers who buy organic products have stronger preference for organic produce over non-organic produce. Such customers will appreciate and prefer the wide variety & selection of organic products on Instacart.

# 7.0 FEATURE ENGINEERING

In machine learning, algorithms are not the most important tool but the data is(2021). With proper data and features, the simplest algorithm can provide the best solution but in absence of the right features, none will. A feature is a measurable characteristic of the data, and feature engineering is the process of extracting features from the raw data ("Feature (Machine Learning)," 2021).

In this step, we will create features from the raw data by joining related tables and applying mathematical calculations. This step will make training data stronger, and improve the performance of the models.

## 7.1 Feature Expansion

We will use order_prior data which contains information about past orders of the users to create features for the model. A complete dataset is created by joining the order_prior table with the orders table on 'order_id'and further joining the resultant table with the products table on 'product_id'.

The new table contains 32434489 rows and 10 columns.

**Figure 49: Merged dataset**

```
#merging prior dataset with orders
df = prior.merge(orders, on = 'order_id', how = 'inner').merge(products, on = 'product_id', how = 'left')

#dropping unrequired columns
df.drop(['aisle_id', 'department_id', 'eval_set'], axis=1, inplace=True)

#day_since_prior order contain nan value for first order. Imputing nan with 0
df = df.fillna(0)

df.head()
```

| | order_id | product_id | add_to_cart_order | reordered | user_id | order_number | order_dow | order_hour_of_day | days_since_prior_order | product_name |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2 | 33120 | 1 | 1 | 202279 | 3 | 5 | 9 | 8.0 | Organic Egg Whites |
| 1 | 2 | 28985 | 2 | 1 | 202279 | 3 | 5 | 9 | 8.0 | Michigan Organic Kale |

There is potential of reorganizing and creating new features based on the given features. We will derive new features based on users, products and user-product characteristics. The following table presents the information about created features. All newly created features are then merged together on primary keys i.e. user_id and product_id to create a new table named 'features'.

**Figure 50: Created Features**

| Feature Name | Description | Code |
|---|---|---|
| **User specific features** | | |
| 'total_orders_per_user' | Number of orders by a user | df.groupby('user_id')['order_number'].nunique().reset_index(name = 'total_orders_per_user') |
| 'total_products_per_user' | Number of products bought by a user | df.groupby('user_id')['product_id'].count().reset_index(name = 'total_products_per_user') |
| 'unique_products_per_user' | Number of unique products bought by a user | df.groupby('user_id')['product_id'].nunique().reset_index(name = 'unique_products_per_user') |
| 'total_reorders_per_user' | Number of reordered products by a user | df.groupby('user_id')['reordered'].sum().reset_index(name = 'total_reorders_per_user') |
| 'reorder_ratio_per_user' | Average of reorders by a user | df.groupby('user_id')['reordered'].mean().reset_index(name = 'reorder_ratio_per_user') |
| 'avg_gap_in_orders_per_user' | Average number of days since prior order | df.groupby('user_id')['days_since_prior_order'].mean().reset_index(name = 'avg_gap_in_orders_per_user') |
| **Product specific features** | | |
| 'prod_reorder_ratio' | Average reorders for a product | df.groupby('product_id')['reordered'].mean().reset_index(name = 'prod_reorder_ratio') |
| 'num_orders_for_prod' | Total number of orders for a product | df.groupby('product_id')['order_id'].count().reset_index(name = 'num_orders_for_prod') |
| 'avg_cart_order_of_product' | Average sequence in | df.groupby('product_id')['add_to_cart |

| | | |
|---|---|---|
| | which a product is added to the cart | _order'].mean().round().reset_index(name = 'avg_cart_order_of_product') |
| 'is_organic' | 1 if a product is organic, else 0 | np.where(v['product_name'].str.contains('Organic'),1,0) |
| 'is_vegan' | 1 if a product is vegan, else 0 | np.where(v['product_name'].str.contains('Vegan'),1,0) |
| 'is_Gluten-free' | 1 if a product is gluten-free, else 0 | np.where(v['product_name'].str.contains('Gluten'),1,0) |
| **User-product specific** | | |
| 'total_prod_by_user' | Number of orders per product by a user | df.groupby(['user_id', 'product_id'])['order_id'].count().reset_index(name = 'total_prod_by_user') |
| 'prod_reorder_by_user' | Number of reorders per product by a user | df.groupby(['user_id', 'product_id'])['reordered'].sum().reset_index(name= 'prod_reorder_by_user') |
| 'prod_reorder_ratio_by_user' | Average reorders per product by a user | df.groupby(['user_id', 'product_id'])['reordered'].mean().reset_index(name = 'prod_reorder_ratio_by_user') |
| 'days_since_prior_prod_user' | Average days since prior order for a product by a user | df.groupby(['user_id', 'product_id'])['days_since_prior_order'].mean().reset_index(name = 'days_since_prior_prod_user') |

### 7.2 Data Preparation

To create the final data, first, a list of users is created from a merged table of order_train and orders tables. This users list is used to extract features from the feature table and finally merged with the order_train table, which contains the latest order information of the users. In the final_data table, nan values from columns such as order_dow, days_since_prior_order, and order_hour_of_the_day are imputed with the mean of the column. Next, products that were ordered for the first time in the latest order were removed, since the aim of the model is to predict which reordered products will be bought by the user.

Finally, irrelevant columns like ‘order_id’, ‘product_name’, ‘add_to_cart_order’,’user_id’, and ‘product_id’ are removed before training the model and categorical columns like ‘order_dow’ were one-hot encoded. Our Final dataset has 8,474,661 rows and 25 columns.

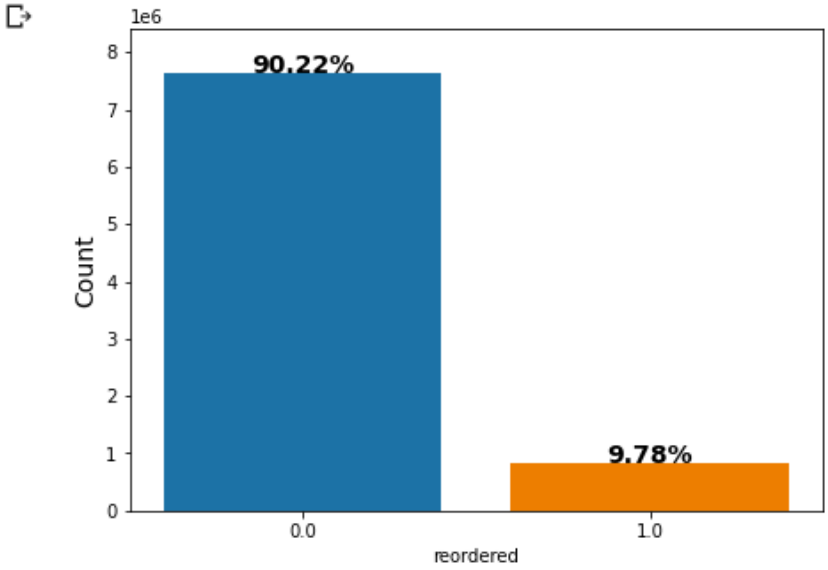
Next, the dataset is divided into train(70%) and test(30%) using the train_test_split from sklearn.model_Selection.

### 7.3 Class Distribution

A dataset with skewed class distribution is known as an imbalanced dataset. The class that has the largest proportion of the data is known as the ‘majority’ class, while a class that has a smaller proportion of the data is known as the ‘minority’ class. Machine learning algorithms assume equal distribution of the class. Hence, an imbalanced dataset results in the poor predictive performance of the model, especially for the minority class because the model is more biased towards the majority class.

**Figure 51: Class Imbalance**

```
total = len(df)
plt.figure(figsize=(7,5))
g = sns.countplot(x='reordered', data=df)
g.set_ylabel('Count', fontsize=14)
for p in g.patches:
    height = p.get_height()
    g.text(p.get_x()+p.get_width()/2.,
            height + 1.5,
            '{:1.2f}%'.format(height/total*100),
            ha="center", fontsize=14, fontweight='bold')
plt.margins(y=0.1)
plt.show()
```




Our dataset has an imbalance class distribution, with the majority to minority ratio of 9:1. This imbalanced class distribution will hamper the accuracy of the model. The common approach to handling the imbalanced problem is over-sampling & under-sampling. Oversampling is a preferred method because all the information remains intact whereas, in the undersampling technique, some potentially important information is discarded.

However, since we have a large dataset, the oversampling technique will further increase the size of the dataset and the computational cost. Hence, undersampling is the preferred choice in the current scenario in which the size of the majority class will be reduced. This will decrease the computational burden on the machine and make the dataset more manageable.

UNDERSAMPLING TECHNIQUES:

- Near Miss Undersampling: This method selects samples based on the distance from the majority class. There are three versions of near-miss undersampling. The first version, NearMiss-1 considers the smallest average distance to the three closest minority classes. NearMiss-2 takes samples from the majority class that has the smallest average distance from the farthest minority class. NearMiss-3 selects the closest majority class samples for each minority class.
    - Advantage: One of the most powerful and balanced ways of undersampling. Version-3 guarantees that every minority class is surrounded by a majority class.
    - Disadvantage: High computational cost on large datasets.
- Random Undersampling: As the name suggests, it randomly removes majority class data points till it matches the number of minority classes
    - Advantage: Easy implementation and faster execution. It has the ability to handle large datasets with limited memory requirements.
    - Disadvantage: May lose some of the important information.
- Condensed Nearest Neighbor: In the method, a subset of samples results in no loss in model performance. Datapoints that can't be correctly classified are placed in 'store'.
    - Advantage: It does not compromise the model performance.
    - Disadvantage: This method can be slow as it passes through training data multiple times (Brownlee, 2021).

Selected technique: Random Undersampling because it is the most efficient technique to handle large datasets even with limited memory. 'RandomUnderSampler' is imported from the 'imblearn' library and applied to the training and test dataset.

**Figure 52: Train & Test data after undersampling**

```
#after randomly undersampling
print(train_x.shape, train_y.shape)
print(test_x.shape, test_y.shape)
```

```
(1161314, 25) (1161314,)
(496334, 25) (496334,)
```

Final training dataset has 1,161,314 rows and 25 columns and the test dataset has 496,334 rows and 25 columns.

# 8.0 MODEL EXPLORATION

## 8.1 Model Introduction

Binary classification is a supervised learning algorithm that classifies data points into one of the two classes. In the current business problem, it would be if a product will be reordered by a user or not. If a model successfully predicts the true class of a datapoint, it is known as 'True positive' (TP). If a model successfully predicts the false class of a datapoint, it is known as 'False positive'(FP). If a model predicts the true class as false, it is known as 'False negative'(FN). Lastly, if a model predicts the false class as true, it is shown as 'False positive'(FP).

These above values are used to calculate the following metrics for binary classification model:

- Accuracy: How accurately model is able to classify the classes

  (TP + TN) / (TP + FP + TN + FN)

- Precision: Also known as Positive predictive value. It is the ratio of relevant instances to the predicted instances. It is a measure of quality.
  - TP/ ( TP + FP)
- Recall: Also known as true positive rate. It is the ratio of positive class prediction to all positive data points in the dataset. It is a measure of quantity.
  - TP/ (TP + FN)
- F-1 Score: It is a harmonic mean of precision and recall.
  - 2 x (precision x recall / precision + recall)
- AUC-ROC (Area under Receiver Operating Curve): It is a measure of the model's ability to distinguish between classes by plotting true positive rate against false negative rate. Y-axis contains true positive rate values while the x-axis contains false negative values. Hence, a curve closer to the top-left corner indicates better accuracy of the model.

For model evaluation higher these metrics are, better the model is. For balanced class dataset accuracy is the dominant metrics, which measures how accurately a model classifies the correct classes.

Following are the codes that will calculate evaluation metrics and plot confusion matrix and ROC curve.

**Figure 53: Metrics function**

```
def metrics(prediction):

  ''' This function will return accuracy,precision, recall, F1 score and ROC AUC'''

  accuracy = accuracy_score(test_y, prediction)
  f1 = f1_score(prediction,test_y)
  precision = precision_score(prediction,test_y)
  recall = recall_score(prediction,test_y)
  auc = roc_auc_score(prediction,test_y)
  print("Accuracy: {:.4%}".format(accuracy))
  print("Precision: {:.4%}".format(precision))
  print("Recall: {:.4%}".format(recall))
  print("F1 Score: {:.4%}".format(f1))
  print("ROC AUC: {:.4%}".format(auc))
```

**Figure 54: Confusion matrix plot function**

```
def confusion_matrix_plot(model):

  ''' This function will create a Confusion Matrix plot '''

  fig, ax = plt.subplots(figsize=(8, 8))
  plot_confusion_matrix(model, test_x, test_y, display_labels=["Not Reordered","Reordered"],
                        cmap=plt.cm.Blues,values_format = '.6g', ax = ax)
  plt.title('Confusion Matrix')
  plt.xlabel('\nPredicted Values')
  plt.ylabel('Actual Values ')
  plt.show();
```

**Figure 55: plot_roc_curve function**

```
def plot_roc_curve(fper, tper):

  ''' This function will create a ROC curve'''

  plt.figure(figsize=(8,7))
  plt.plot(fper, tper, color='red', label='ROC')
  plt.plot([0, 1], [0, 1], color='green', linestyle='--')
  plt.xlabel('False Positive Rate')
  plt.ylabel('True Positive Rate')
  plt.title('Receiver Operating Characteristic Curve')
  plt.legend()
  plt.show();
```

## 8.2 Model Techniques

### 8.2.1 DECISION TREE

A decision tree is a supervised machine learning algorithm. As the name suggests, in this algorithm data is continuously split at each decision point, so it grows like a tree where decision nodes or splitting nodes are where data is split and a terminal node or leaf node is the decision. The decision to split at each node is based on 'purity', which measures the extent to which all data points belong to the same class. If a node is split 50-50, a node is 100% impure whereas it is a pure node if all data points belong to the same class(Chauhan, 2022).

Gini impurity is used as a measure of the probability of misclassification of a data point. The range of Gini impurity is between 0-0.5. The default criteria to measure impurity is gini. Another metric is 'entropy', which also measures the disorder of the input variables with the target variable. The range of entropy is between 0-1. Computationally, entropy is more complex than gini because entropy uses logarithms. Hence, gini impurity is faster and more efficient (2022).

Advantage: A decision tree is simple to create and easy to interpret. Decision trees can be visualized and can be easily interpreted. It requires less data preparation and can handle both numeric and categorical data.

Disadvantage: Decision trees often suffer from overfitting. They are unstable, and a minor change in data results in different trees. Does not handle imbalanced data very well and results in biasness.

OUTPUT:

We first ran a DecisionTreeClassifier() with default parameters and we got a low accuracy score of 63.5%. A low accuracy score is a sign of overfitting. We will use hyperparameter tuning to optimize the model.

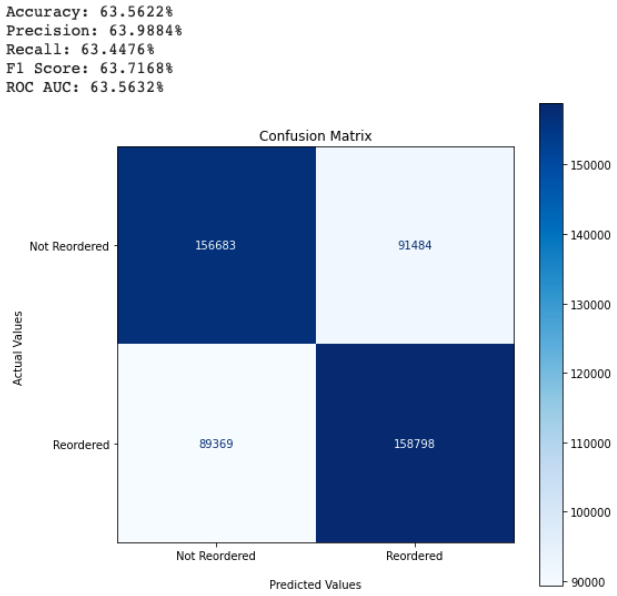
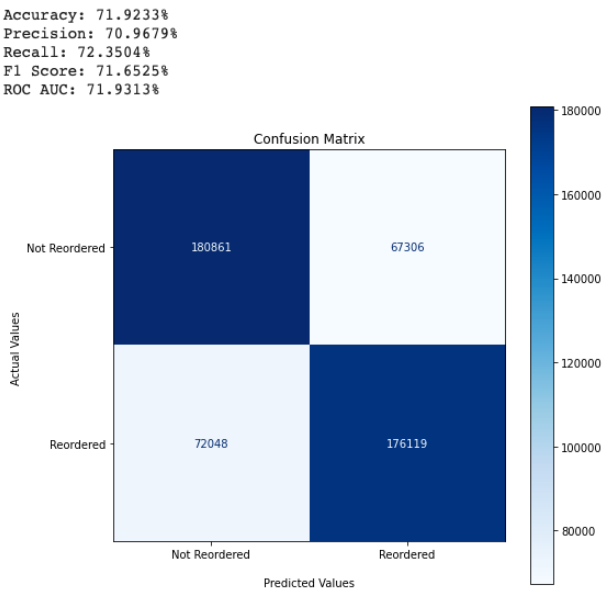
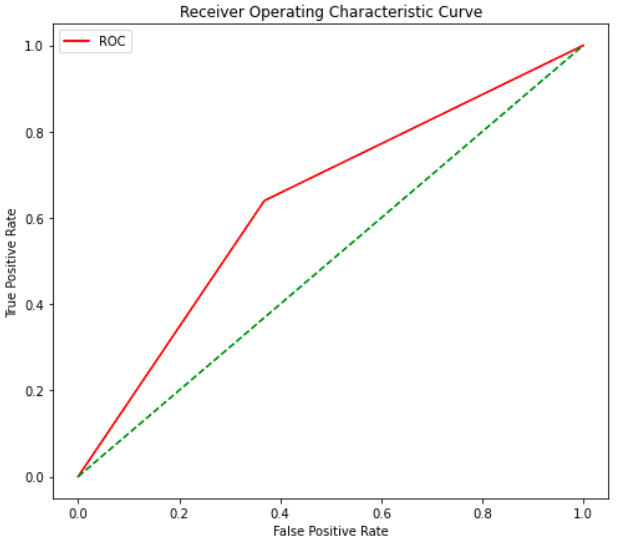
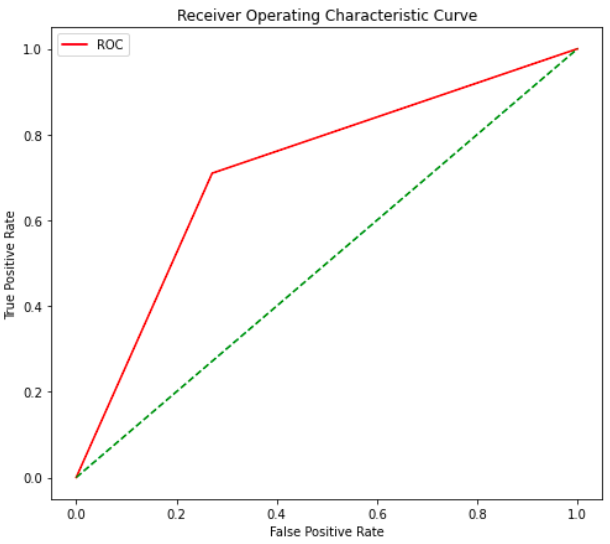
Hyperparameters are the variables whose values are set before model training to maximize the performance of the model. We will use RandomizedSearchCV(), which randomly selects the combination of the hyperparameters from the given range. It will run on all selected combinations and determine the best combination with the highest accuracy score. We then trained our model on the best parameters.

**Figure 56: Hyperparameter tuning Decision tree**

```
#Hyperparameter optimization with RandomizedsearchCV
param_grid = {
    'max_depth': list(range(2, 10)),
    'min_samples_split': list(range(5, 20)),
    'min_impurity_decrease': [0, 0.001, 0.0011],
}

gridSearch = RandomizedSearchCV(DecisionTreeClassifier(), param_grid, cv=5,
                           n_jobs=-1)

gridSearch.fit(train_x, train_y)

print('Initial score: ', gridSearch.best_score_)
print('Initial parameters: ', gridSearch.best_params_)
```

```
Initial score:  0.720252230681532
Initial parameters:  {'min_samples_split': 9, 'min_impurity_decrease': 0, 'max_depth': 9}
```

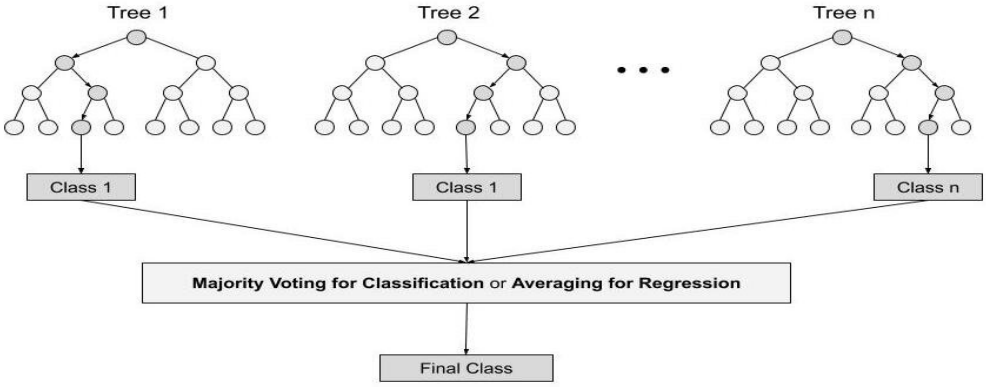
**Figure 57: Decision Tree performance report**

| Default Decision Tree | Decision tree after hyperparameter tuning |
|---|---|
|  |  |
|  |  |

### 8.2.2 RANDOM FOREST

Random forest also known as random decision forest is an ensemble learning technique. It is an ensemble of multiple decision trees created on random samples and the final output is selected on the basis of the majority vote. This method of ensemble is known as ‘bagging’.

**Figure 58: How Random Forest works**



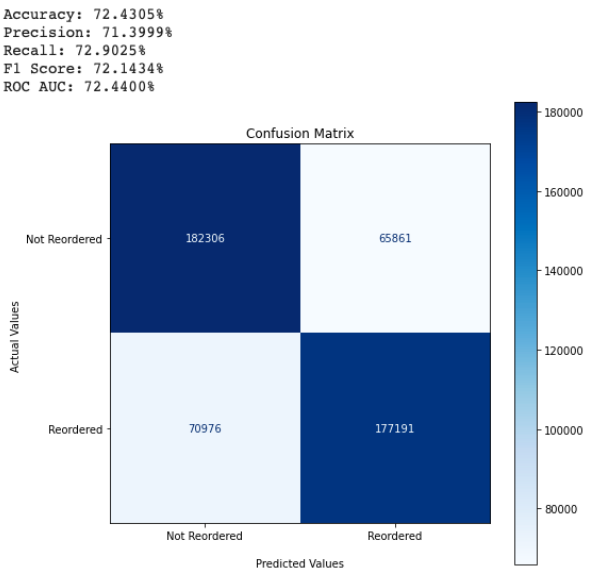
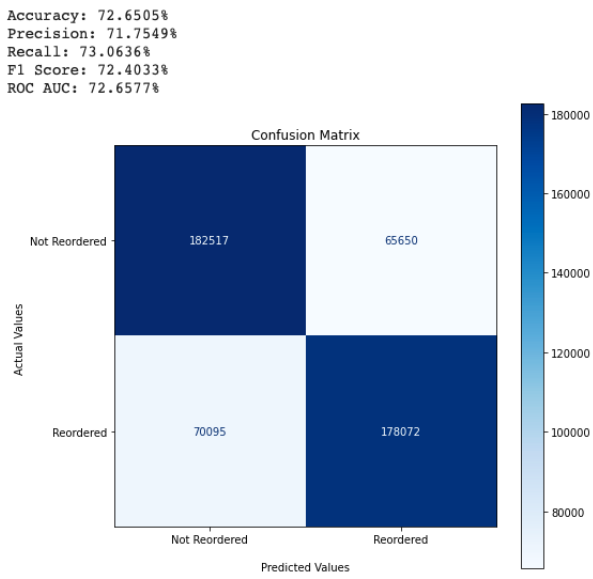
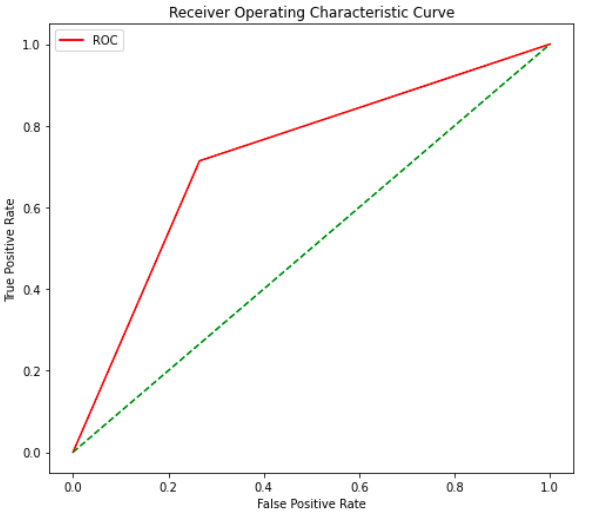
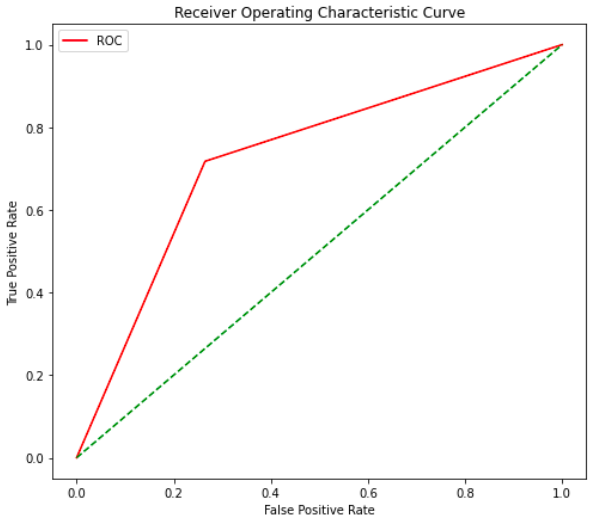

Source: https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

**OUTPUT:**

We first ran with default parameters and then increased the number of estimators to 150.

N_estimators is the number of trees to be built. High number of trees results in better accuracy and stronger model but it increases the computational time. Increase in metrics with increase in the number of estimators shows that there is potential of model performance with a high number of estimator values. However, due to limited computational capacity we ran only 150 trees but the model will be improved with large value in future work.

**Figure 59: Random Forest Performance Report**

| Random forest with n_estimators = 100 | Random forest with n_estimators = 150 |
|---|---|
|  |  |
|  |  |

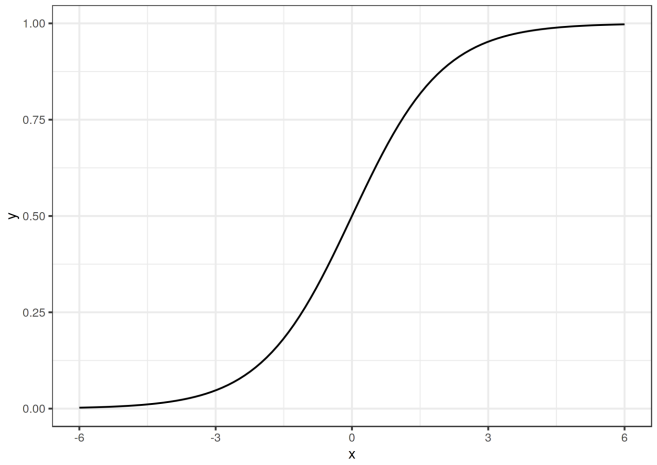
### 8.2.3 LOGISTIC REGRESSION

Contrary to the name. Logistic regression is a linear classification model rather than a regression. It is a supervised machine learning algorithm that calculates the probability of the class using a logistic function. It is an extension of the linear regression and its formula is also derived from linear regression (Molnar, 2022).

$$\text{logistic}(\eta) = 1/ (1 + \exp(-\eta)$$

Alternative to linear regression which fits a straight line, logistic regression uses a logistic function which forces the output to be in one of the two classes (0,1).

**Figure 60: Sigmoid function**



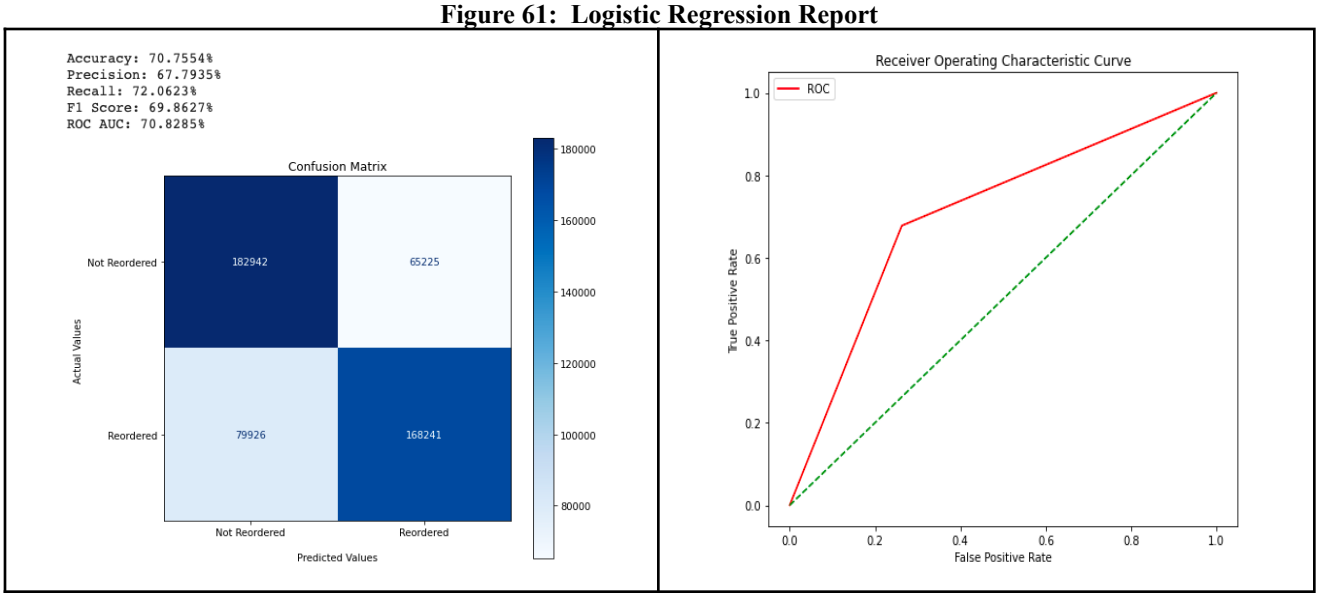

Source: https://christophm.github.io/interpretable-ml-book/logistic.html

Advantage: Logistic regression generates the probability of the outcome, so an outcome with 99% probability and an outcome with 51% probability shows the strength of the predictive class. It can easily be extended to multiple classes.

Disadvantage: It assumes a linear relationship between dependent and independent variables. It tends to overfit on high-dimensional datasets. It is not powerful to compute complex relationships.

Output: Logistic regression performs poorly with just a baseline accuracy of 70%. While Recall is fair, the F1 score is poor and the reason behind the poor performance is the lack of linear relationship between target and input variables. Parameter tuning also did not improve the result of logistic regression.

**Figure 61: Logistic Regression Report**




### 8.2.4 EXTREME GRADIENT BOOSTING (XGBoost)

XGBoost is a supervised machine learning algorithm. It is based on decision trees and improves on random forest and gradient boost. Boosting is an ensemble technique in which new models are added to rectify the error of current models. This step continues until there is no further scope for improvement in the model.

XGBoost is known for its computational power & performance in handling complex datasets. In this algorithm, initial predictions are made and the residual value is calculated. New models are created that calculate the residual or error of the existing model. This process is repeated until there is no further improvement or the residual value stops reducing.

The following methods are used to optimize the algorithm:

- Regularization: It is used to avoid overfitting by calculating the similarity score to reduce the sensitivity.
- Pruning: Gamma, tree complexity parameter is used to compare the gains. A branch is removed if the gain is smaller than gamma. This prevents overfitting.
- Weighted quantile sketch: Weighted quantile is used as a threshold to split the data.
- Parallel learning: In this method, data is divided into blocks that are used to create trees.
- Sparsity: XGBoost works around sparsity by trying in both directions and finding the default direction by calculating the gain.
- Cache: Cache memory of the system is used to calculate the similarity score and output values. This method is faster and hence improves the performance of the model.
- Blocks for Out-of-core Computation: When a dataset is too large to fit in the memory, it is divided into blocks and compressed. Uncompressing is faster than reading from the hard drive (ArcGIS Pro, n.d.)

Advantage: It is an easy-to-use algorithm, with high performance, better accuracy, and faster processing. It easily handles large complex datasets.

Disadvantage: It does not perform very well on a very sparse dataset and unstructured dataset. It is sensitive to outliers.

**Output:**

We first ran XGBoost with the default parameter and then chose the following hyperparameter on the basis of tuning result.

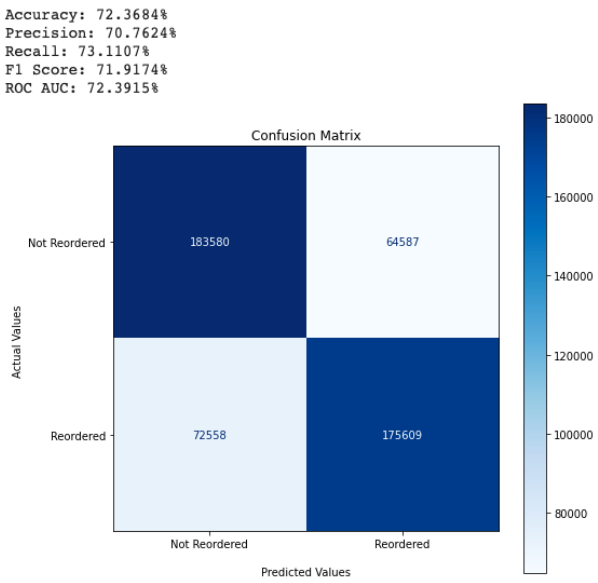
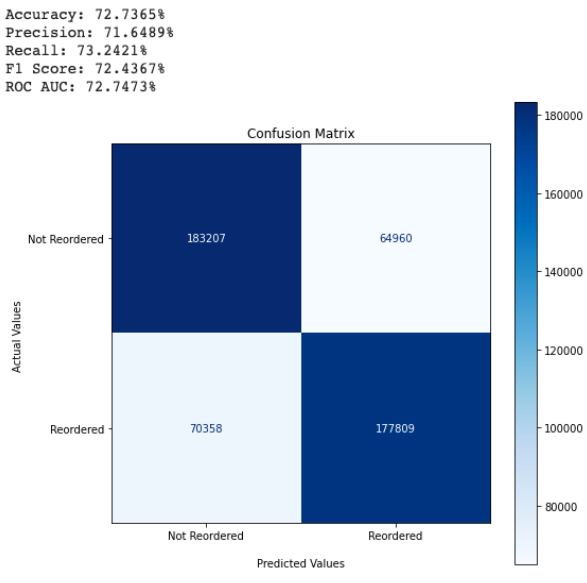
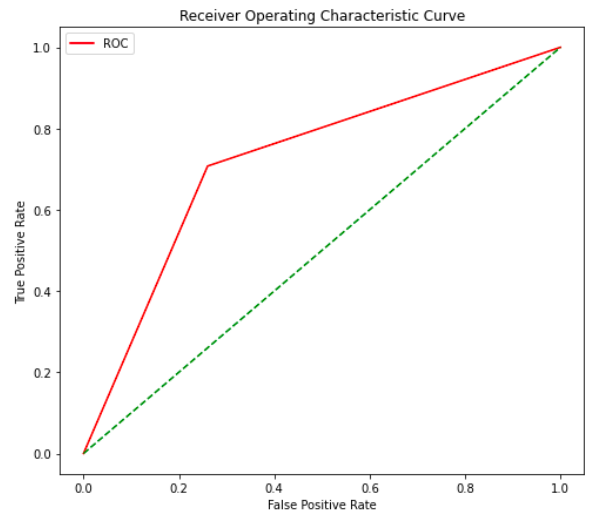
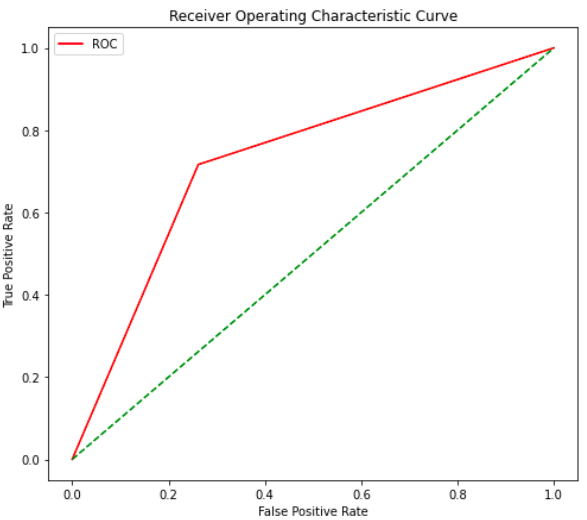
**Figure 62: XGBoost model**

```
clf= xgb.XGBClassifier(objective='binary:logistic', eval_metric = 'logloss', colsample_bytree = 0.7, learning_rate = 0.1,
                       max_depth = 6, subsample = 0.7, gamma = 0, reg_lambda = 5.0, n_estimators = 400)
```

Similar to random forest, accuracy improved with increase in the number of estimators.

Current RAM limited the number of estimators to 400.

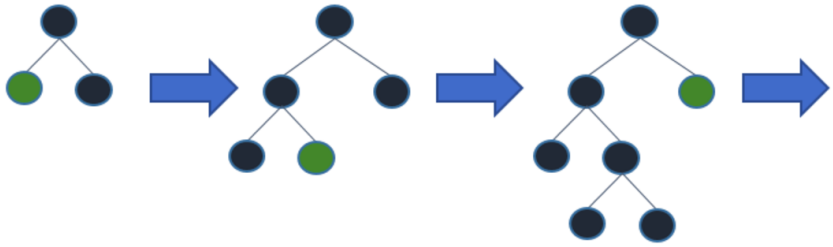
**Figure 63: XGBoost Performance report**

| With n_estimators = 100 | With n_estimators = 400 |
|---|---|
|  |  |
|  |  |

### 8.2.5 LIGHTGBM

LightGBM is a gradient boosting technique based on decision trees. LightGBM grows vertically as it splits the tree leaf-wise whereas other boosting techniques grow horizontally by splitting the tree depth-wise.

**Figure 64 : Leaf-wise tree growth**



Source:https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters

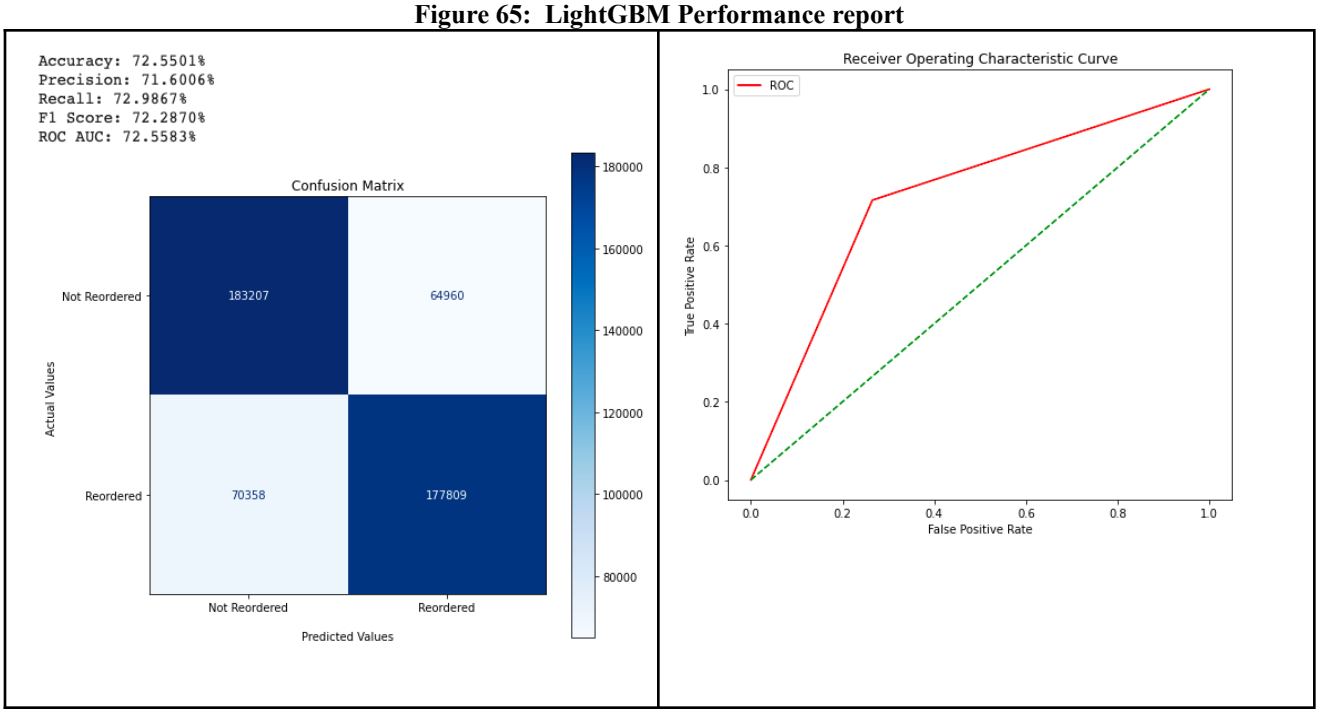
It uses Gradient-based one-side sampling and Exclusive feature bundling to counteract the limitation of the histogram-based algorithm. In the Gradient-based one-side sampling technique, instances with large gradients are kept as they result in more information gain and randomly drop those instances with low gradients. This results in a more accurate gain estimation. Exclusive feature bundling bundles the mutually exclusive features into a single feature, which improves the computation speed without compromising accuracy (GeeksforGeeks, 2021).

Advantage: LightGBM has lower memory usage, higher efficiency, and better accuracy. It is also capable of handling large-scale data efficiently.

Disadvantage: It suffers from overfitting as it grows leaf-wise. LightGBM is also not suitable for small datasets, as it will more likely overfit on small datasets.

OUTPUT:

**Figure 65: LightGBM Performance report**




### 8.2.6 CATBoost

CATBoost is a gradient boosting algorithm based on a decision tree. It is an acronym for category boosting, which means CATBoost handles categorical variables effectively. It is a robust technique that also works with small datasets, unlike other boosting algorithms. It used the Minimal variance sampling technique to split features, which is a weighted sampling technique. The weighted sampling technique improves the quality of the model by decreasing the number of required examples for each iteration.

Advantage: It is a robust technique that provides better results without extensive hyperparameter tuning. It is the fastest among other ensemble techniques. It outperforms other techniques in terms of categorical variables.

Disadvantage: Parameter tuning of categorical variables can be difficult.

**OUTPUT:**

Algorithm was first trained with default parameters and then hyperparameter tuning was performed using RandomizedSearchCV to find the best parameter.

**Figure 66: Hyperparameter tuning**

```
[ ]  grid = {'learning_rate': [0.2, 0.1, 0.3, 0.4],
             'depth': [4, 6, 10],
             'l2_leaf_reg': [1, 3, 5, 7, 9],
             'loss_function' : ['Logloss' , 'CrossEntropy']}

     model = CatBoostClassifier()
     randm = RandomizedSearchCV(estimator=model, param_distributions = grid, cv = 2, n_iter = 10, n_jobs=-1)
     randm.fit(train_x, train_y)
```

```
============================================================
 Results from Random Search
============================================================

 The best estimator across ALL searched params:
 <catboost.core.CatBoostClassifier object at 0x7f182ae25650>

 The best score across ALL searched params:
 0.7268464859633139

 The best parameters across ALL searched params:
 {'loss_function': 'Logloss', 'learning_rate': 0.2, 'l2_leaf_reg': 5, 'depth': 4}

 ============================================================
```

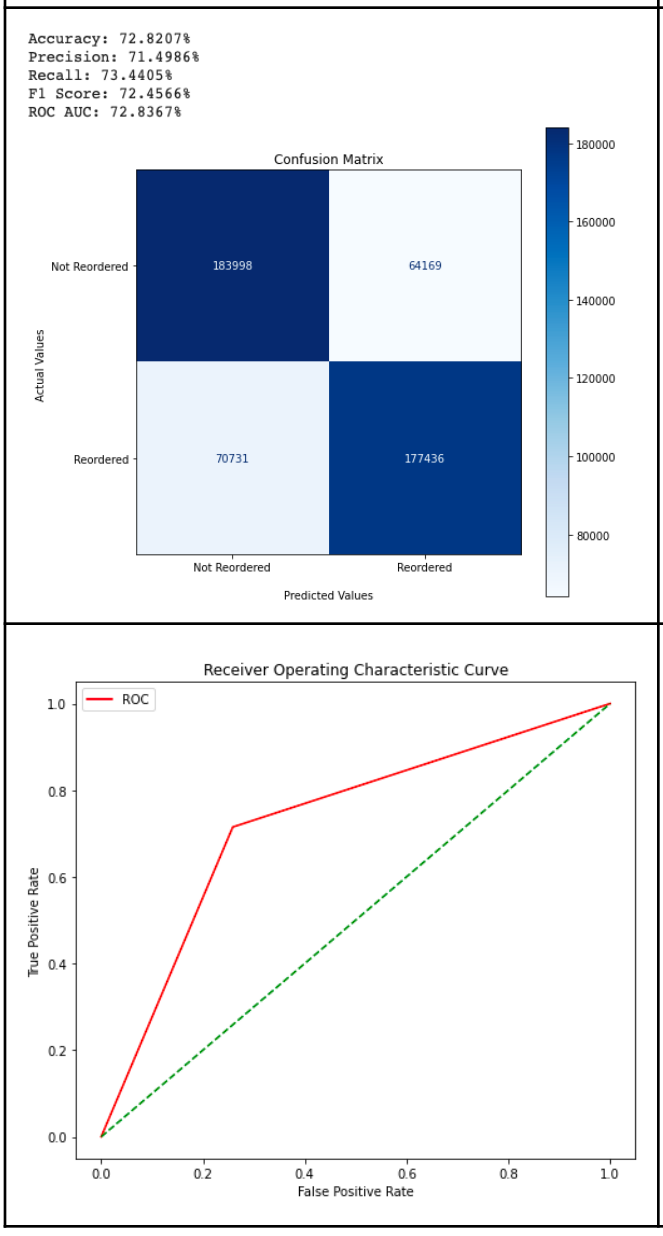
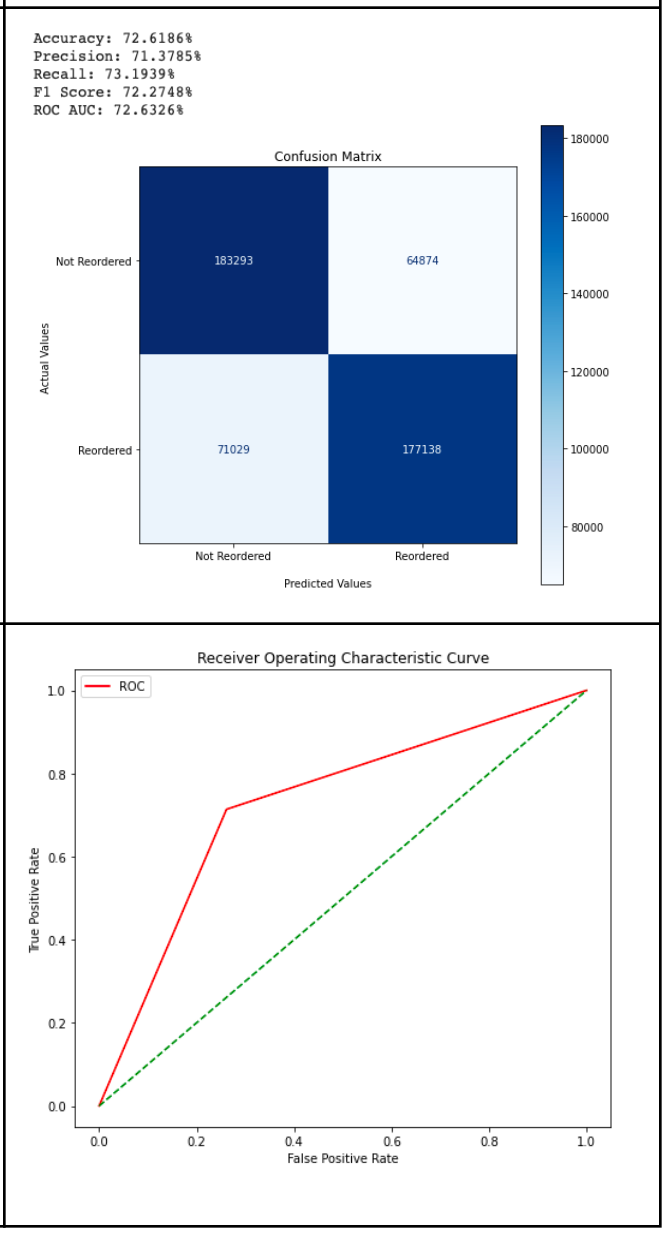
However, hyperparameter tuning decreases the model's performance. Following default parameters performs better.

**Figure 67: Default CATBoost parameters**

```
cb.get_all_params()
```

```
{'auto_class_weights': 'None',
 'bayesian_matrix_reg': 0.10000000149011612,
 'best_model_min_trees': 1,
 'boost_from_average': False,
 'boosting_type': 'Plain',
 'bootstrap_type': 'MVS',
 'border_count': 254,
 'class_names': [0, 1],
 'classes_count': 0,
 'depth': 6,
 'eval_metric': 'Logloss',
 'feature_border_type': 'GreedyLogSum',
 'force_unit_auto_pair_weights': False,
 'grow_policy': 'SymmetricTree',
 'iterations': 1000,
 'l2_leaf_reg': 3,
 'leaf_estimation_backtracking': 'AnyImprovement',
 'leaf_estimation_iterations': 10,
 'leaf_estimation_method': 'Newton',
 'learning_rate': 0.2097329944372177,
 'loss_function': 'Logloss',
 'max_leaves': 64,
 'min_data_in_leaf': 1,
 'model_shrink_mode': 'Constant',
 'model_shrink_rate': 0,
 'model_size_reg': 0.5,
 'nan_mode': 'Min',
 'penalties_coefficient': 1,
 'pool_metainfo_options': {'tags': {}},
 'posterior_sampling': False,
 'random_seed': 0,
 'random_strength': 1,
 'rsm': 1,
 'sampling_frequency': 'PerTree',
 'score_function': 'Cosine',
 'sparse_features_conflict_fraction': 0,
 'subsample': 0.800000011920929,
```

**Figure 68: CATBoost Performance report**

| Default CatBoost classifier | After Hyperparameter tuning |
|---|---|
|  |  |
|  |  |



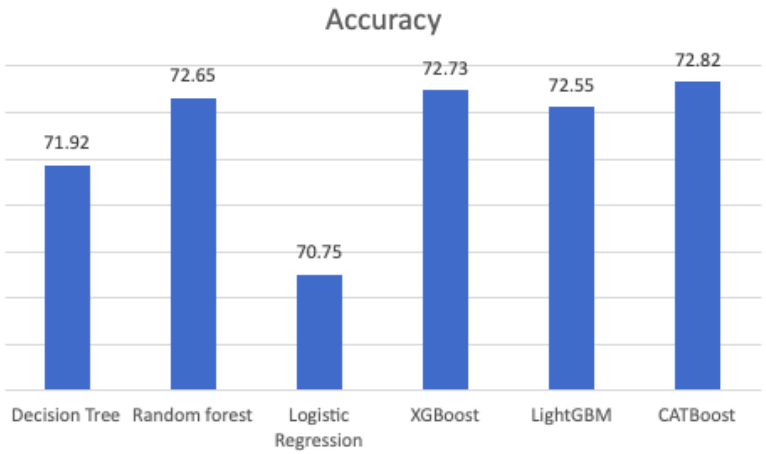


## 8.3 Model Performance

In this section, performance of models will be compared on the basis of evaluation metrics. Overall, ensemble models provide the best results. Accuracy measures how accurately a model is able to classify classes, this metric is used when the class is balanced. Since, in data preparation stage class was balanced using undersampling technique, accuracy metrics can be used as an evaluation metric.

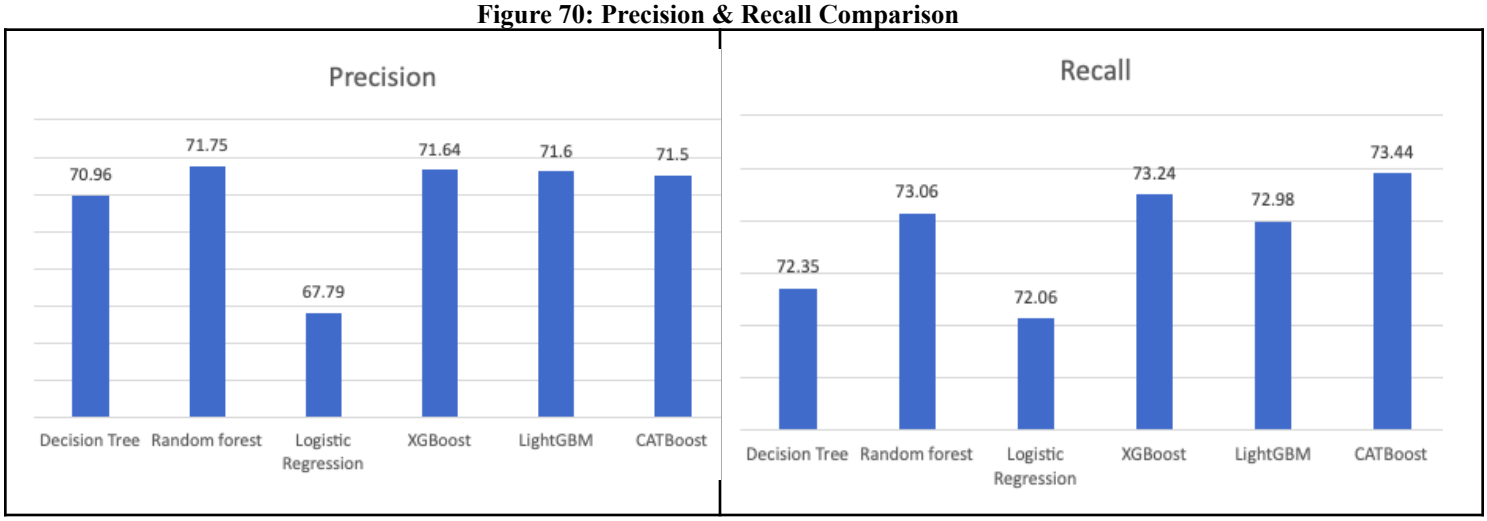
XGBoost(72.73) and CatBoost(72.82) have almost equal accuracy scores, with CATBoost obtaining the better accuracy.

**Figure 69: Accuracy Comparison**



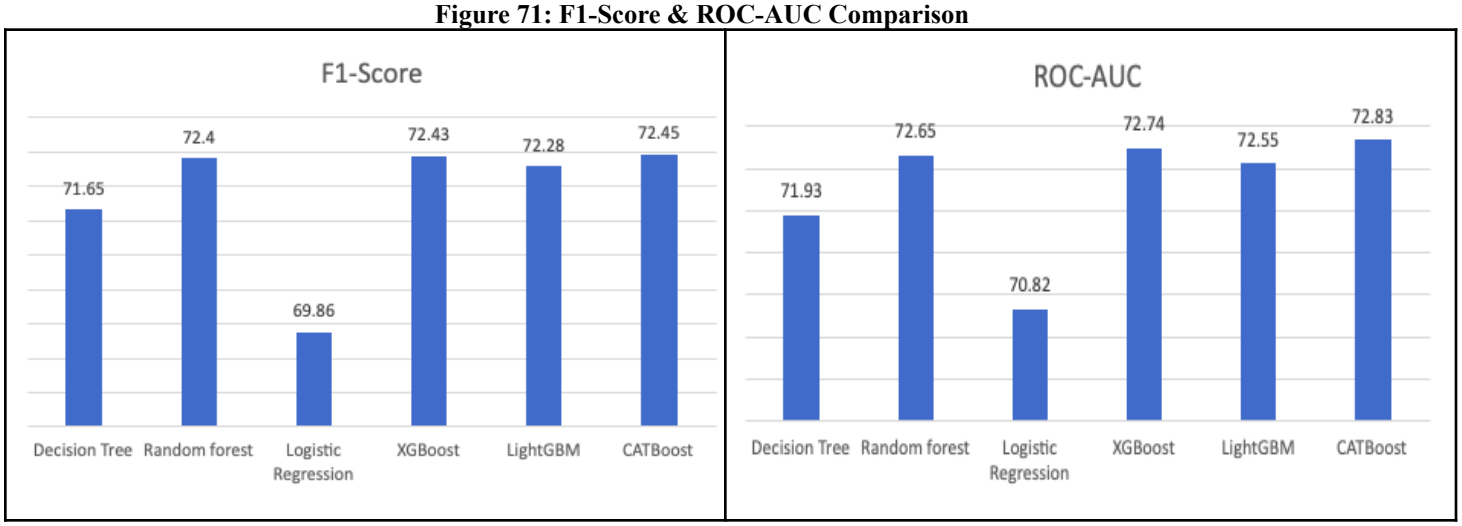

Precision calculates the number of positive class predictions that actually belong to the positive class and recall measures the number of positive class predictions out of all positive classes in the dataset. For our current business problem, recall will be a superior metric than precision because it is important to detect as many 1 as possible. In terms of recall, CATBoost(73.32) outperforms other models.

**Figure 70: Precision & Recall Comparison**



F1-score combines both precision and recall, as a result it is an important metric when the dataset is imbalanced. ROC-AUC score determines the strength of the model to classify the classes. Again, XGBoost(72.74) and CATBoost(72.76) have similar results.

**Figure 71: F1-Score & ROC-AUC Comparison**




All three of our ensemble techniques i.e. XGBoost, LightGBM, and CATBoost perform well, with almost similar performance metrics. Hence, in this case, model selection will be based on other factors like speed, computational effectiveness, and technical limitations.

These algorithms differ from each other based on the boosted tree algorithm implementation and their technical capabilities (Nahon, 2021). In the following table, we will compare between key features of the algorithms.

| Feature | XGBoost | LightGBM | CATBoost |
|---|---|---|---|
| Split | It looks for features and split-point that maximizes the gain. The splitting process is slower as it does not use weighted technique. | It used Gradient based one-side sampling, in which a split is made on all instances with large error and random instances with small error. This maintains a balance between high speed and accuracy. | It used minimal variance sampling, in which weighted sampling is done on tree level instead of split level. The instances are sampled in a way that maximizes the accuracy of the split score. |

| | | | |
|---|---|---|---|
| Leaf Growth | It splits according to the max_depth hyperparameter and then starts pruning until there is no more gain. | It grows tree leaf-wise i.e. leaf with minimum loss is chosen to grow. This results in unbalanced trees and overfitting in small datasets. | It grows a balanced tree, feature-split pair with lowest loss is used for level nodes. |
| Feature importance | It used tow method Gain (contribution in improvement in accuracy by a feature) and Split (number of time a feature occurs in each split) | LightGBM uses the same methods as XGBoost to calculate feature importance. | It uses PredictionValuesChange (average effect of change in feature value on prediction value) & LossFunctionChange (difference between loss of value with and without the feature) |
| Categorical variables | No built method to hand categorical variables | It splits features by partitioning categories into two subset. This method sometimes degrades the performance. | It uses a combination of one-hot encoding and an advanced mean encoding |

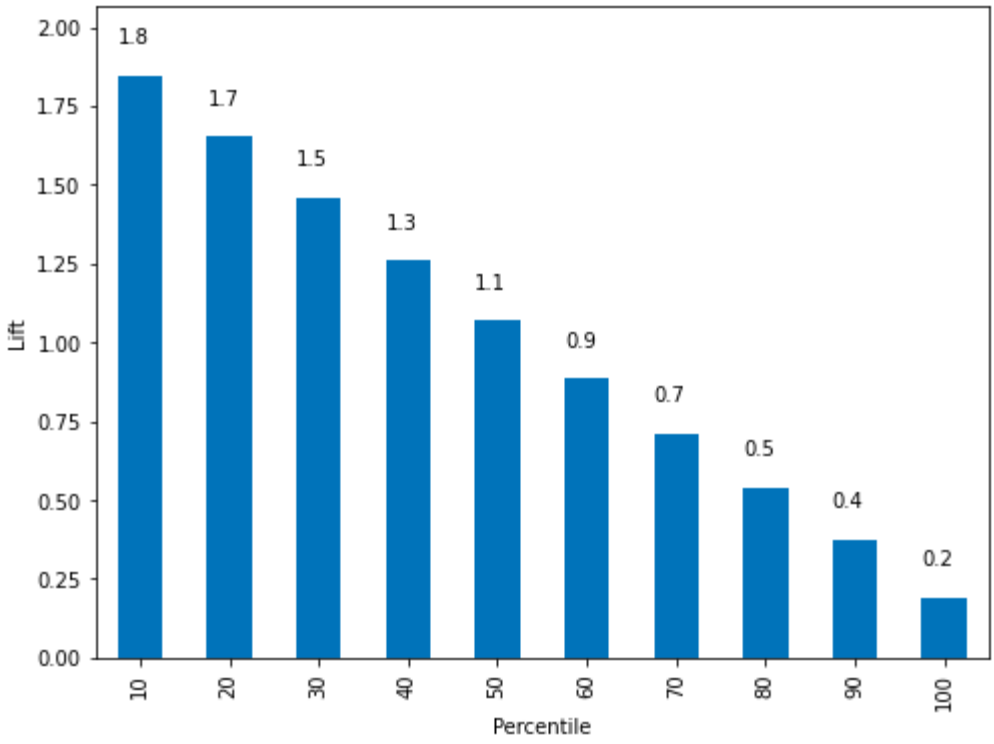
CATBoost is preferred because it outperforms other models in terms of performance metrics even though not with a meaningful difference. In terms of other criteria like speed and computational power, CATBoost is still a better choice and also its inbuilt ability to handle categorical data is also a bonus factor.

## 8.4 Model Selection

### 8.4.1 Model Effectiveness

Lift score is the ratio between the result obtained with a model and the result obtained without a model. The ‘staircase’ effect in the decile chart shows the viability of the model by presenting from most likely to least likely.

**Figure 72: Lift decile chart**



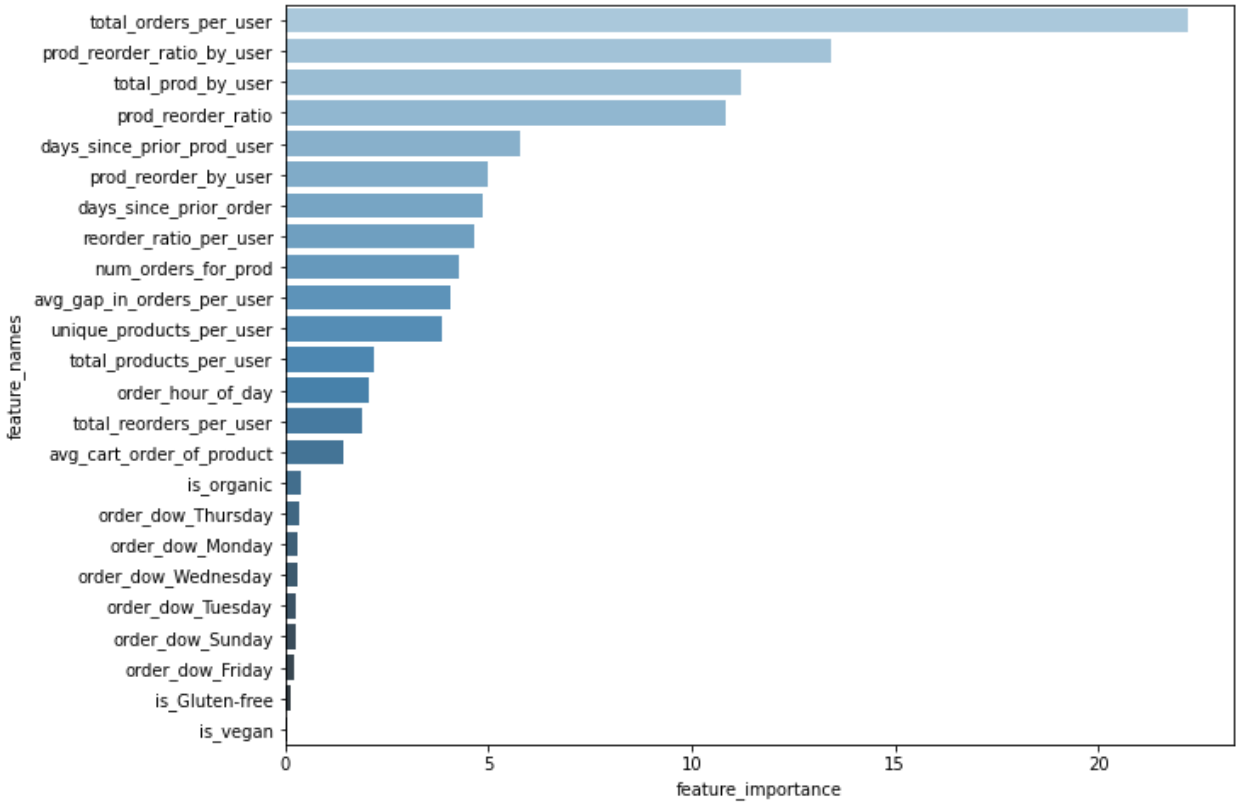

10% of the records that are ranked by the model as "the most probable 1's" yields 1.8 times as many 1's as would a random selection of 10% of the records. Lift value greater than 1 means that the model is better at identifying the reordered products than selecting arbitrarily.

### 8.4.2 Feature Importance

Feature importance is a representation of the importance of the input features in the model. It calculates the effect of the feature on the model. It is one of the most important metrics for model interpretability. CatBoost uses the difference between the metric (Loss function) obtained using the model in normal scenarios (when we include the feature) and the model without this feature.

Below figure represents the important features in the CATBoost model.

**Figure 73: Feature Importance**




Top five features in the model that are predictive of reorders are:

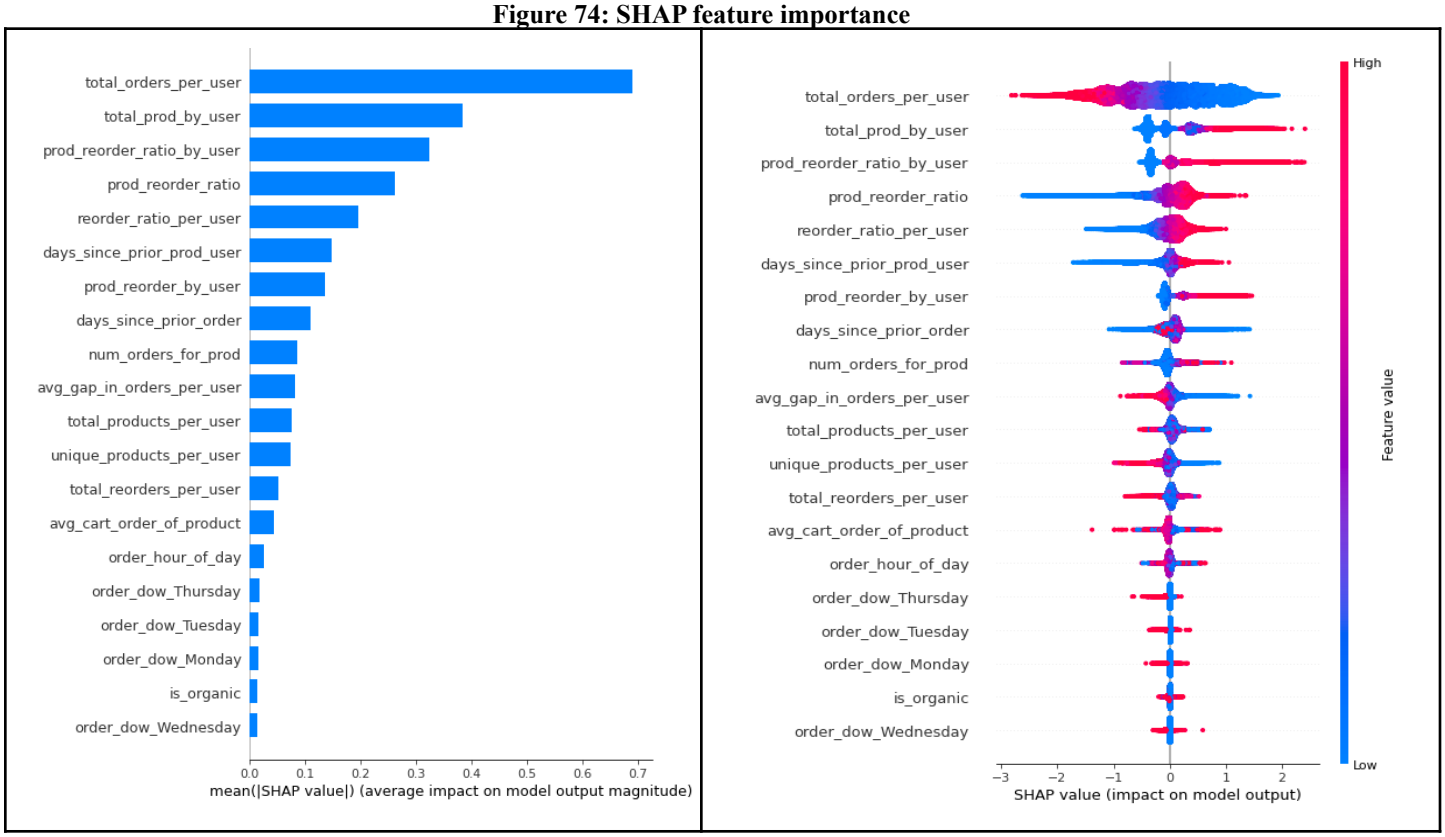
- Total number of orders by a user
- Reorder ratio of user for a product
- Total number of products ordered by user
- Reorder ratio of a product
- Number of days since prior order for a product by a user

CATBoost calculates global feature importance i.e. impact of the feature on the model. This method is biased towards features that have more potential split points. Hence it is not the most reliable method of feature selection. As proposed by Lundberg and Lee, feature importance attribution methods should have:

- Consistency – if a model is changed so that it relies more on a particular feature, then the method must not attribute less importance to that feature; and
- Accuracy – the total contribution of each feature must sum up to the total contribution in the whole model

A model that satisfies both these conditions is SHAP (SHapley Additive exPlanation). SHAP is the average marginal contribution of a feature value across all the possible combinations of features. SHAP accounts for the local importance of the feature. It uses a game-theoretic approach to derive the importance of each feature in the final prediction. It gives local feature importance i.e. the importance of a feature for each data point.

**Figure 74: SHAP feature importance**




Interpretation:

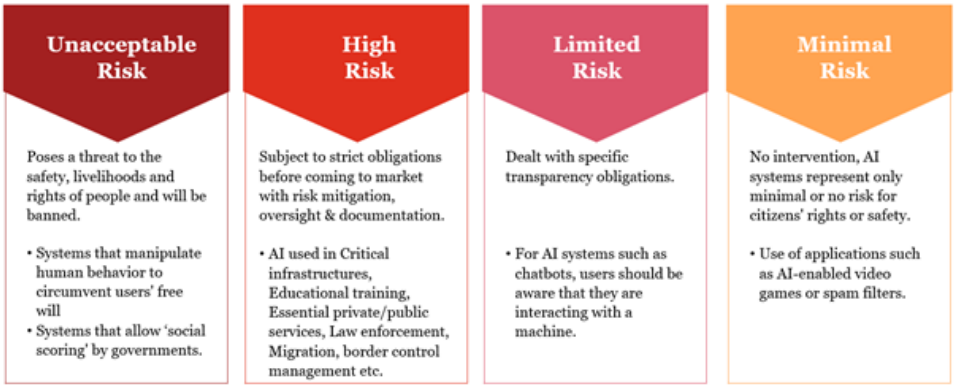
- 'Total_orders_per_user' is the most important feature but it is somewhat equally split, which means that its interaction is dependent on other variables.
- High values of 'total_product_per_user' & 'product_reorder_ ratio_by_user' have a positive impact on the model. While they are not the top important features, they have a huge impact on a subset of the users.
- Lower value of 'prod_reorder_ratio' and 'reorder_ratio_per_user' will have greater negative impact on model.
- Lower value of 'days_since_prior_prod_user' will have a negative impact. So, a product is less likely to be reordered if the gap between two orders is very small. This will be an important factor in determining the best day to send push notification to increase call to action.

# 9.0 GOVERNANCE

Model risk is the potential loss suffered by an organization as a result of decisions based on the internal model's output. Overtime performance of the model degrades and it fails to perform adequately or as previously set standards. This results in faulty business decisions and loss to the organization. To ensure consistent model performance, certain measures such as variable drift monitoring, drift tolerance, variable value range, the addition of new variables, and future market conditions are set before deployment of the model.

Risk tiering is the process of quantifying the potential harm of the model or solution. The European Commission has created the following four tiers to monitor the potential harm of AI/ML learning models (Biswas, 2022).

**Figure 70 : Risk Tier of AI/ML Model**




Source:https://towardsdatascience.com/ai-ml-model-validation-framework-13dd3f10e824

Our model falls under the 'Minimal Risk' tier as it will not project any risk to citizens' rights or safety. Data will be used by the internal teams as per our data privacy policies. In no way any data privacy policy will be compromised at any stage of the model.

The following model verification framework will be followed to monitor the continuous stability of the model:

- Data appropriateness: In data appropriateness, data protection, completeness, integrity, and appropriateness, and pre-processing of the data will be done according to our data privacy guidelines and current process.
- Model testing: Model will be continuously tested on the set hyperparameters, performance metrics, and set benchmarks.
- Soundness and Interpretability: The transparency and explainability of the model will be continuously measured by the model's feature importance and SHAP values.
- Model documentation & version control: An extensive and self-explanatory documentation describing the complete modeling process from data extraction to model selection and interpretation will be maintained. Any modification or change in the model will be saved with a new version number to ensure the whole process and steps are documented.
- Ongoing monitoring: Validators will continuously monitor that all components are in place and as per the standards. This will help in managing and assessing the risk of the model.

**9.1 Data Drift**

Model drift or data drift is the phenomenon in which the data on which the model was trained changes over time. In the current dataset variables that are likely to drift over time are:

- is_organic: Preference for organic produce is rising, and so will the number of orders. As the number of data points will increase, so will the importance of the feature in the model.
- Similar will be the result for the 'is_vegan' and 'is_gluten_free' variables.
- days_since_prior_order: The number of days since the prior order will remain capped at 30 days. Users whose number of days between two orders is greater than 30 days is not our target audience for the current model. These are not our regular customers and they most likely shop only during promotions or heavy discounts.

### 9.2 Model Stability Measure

Following metrics will be used to determine the stability of the model:

| Metric | Current value | Minimum Acceptable value |
|---|---|---|
| Accuracy | 72.82 | 71 |
| F1-Score | 72.45 | 70.5 |
| ROC-AUC | 72.83 | 71 |

### 9.3 Action

- Drift within an acceptable range of 2-3% - No action but will be reported.
- If we observe more than 2-3% of the drift in the current model, the hyperparameters will be fine-tuned to achieve the desired performance.
- If the tweaks in the hyperparameters do not result in performance improvement or the drift is beyond 5-6%, the following next step will be adopted.
    - Variables will be inspected for any out-of-range values or addition of any new variables.
    - Out-of-range values will be transformed within the range and new variables will be used for feature engineering and refitting of the model.
- On the other hand, if the observed drift in performance metrics is beyond 85% then class distribution will be inspected. Class imbalance results in biases in the model, is likely to be the cause in overfitting.

# 10.0 CONCLUSION & RECOMMENDATION

## 10.1 Conclusion & Impact on business Problem

This project consists of two major parts: Order trend and consumer behavior analysis and product purchase prediction. First, by researching transactional data using exploratory data analysis, segmenting customers using cluster analysis, and discovering frequent sets of products using the association rule, we understand the customer personas and their buying behavior like frequency of order, an average number of products in an order, and preferred products. This information will be used in creating personalized push notifications for each cluster of customers. Then, the data knowledge was used to develop and train a predictive model which attains the accuracy of 72.82% in predicting which products will be the user's next order.

The resulting knowledge will help businesses in updating the current product portfolio and in creating strategic promotions for the most reordered products to trigger new sales and keep customers engaged. Additionally, it will help inculcate the top two most preferred interfaces by customers (Figure 3), as discussed in the ‘Introduction’ section of the project:

- Quick selection & checkout
- Easy access and reordering of past purchases.

This addition of a pre-fill feature will increase the customer service value and enhance the shopping experience. This project is a marketing mix model with strategic use of data to improve the current marketing inputs. It will help in quantifying the impact of proposed marketing inputs on sales and increase return on investment by increasing sales.

### 10.2 Future Work

Despite achieving a decent performance metric, our current model has a huge scope for improvement and extension in the current model.

Due to memory limitation, the current model is trained on undersampled data, which resulted in discarding potentially useful information. The design of the current model has the potential to leverage complete and recent data to improve performance and draw better predictions. Customer preferences are evolving and so is the grocery market, so the input of real-time data will provide accurate features that are more relevant to the current market conditions.

Secondly, adding demographic data to the clustering technique will increase the power of the model and better understand customer behavior and develop uniquely appealing products and services for each customer.

# REFERENCES

1010Data. (2021, March). *State of grocery report*. https://1010data.com/media/3019/2021_1010data_stateofgroceryreport.pdf

Apriori algorithm. (2022, July 6). In *Wikipedia*. https://en.wikipedia.org/wiki/Apriori_algorithm

ArcGIS Pro. (n.d.). *How XGBoost algorithm works*. https://pro.arcgis.com/en/pro-app/latest/tool-reference/geoai/how-xgboost-works.html

Aull, B., Begley, S., Chandra, V., & Mathur, V. (2021, July 2). Making online grocery a winning proposition. McKinsey & Company. https://www.mckinsey.com/industries/consumer-packaged-goods/our-insights/making-online-grocery-a-winning-proposition

Biswas, P. (2022, May 24). *AI/ML Model Validation Framework - Towards Data Science*. Medium. https://towardsdatascience.com/ai-ml-model-validation-framework-13dd3f10e824

Brownlee, J. (2021, January 26). *Undersampling algorithms for imbalanced classification*. Machine Learning Mastery. https://machinelearningmastery.com/undersampling-algorithms-for-imbalanced-classification/

Chandra, V., Gill, P., Kohl, S., Venkataraman, K., Yoshimura, J., & Mathur, V. (2022, August 9). *The next horizon for grocery e-commerce: Beyond the pandemic bump*. McKinsey & Company. https://www.mckinsey.com/industries/retail/our-insights/the-next-horizon-for-grocery-ecommerce-beyond-the-pandemic-bump

Chauhan, N. S. (2022). *Decision tree algorithm, explained*. KDnuggets. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html

Ecosystem, E. (2022, May 17). *Understanding k-means clustering in machine learning*. Medium. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

Feature (machine learning). (2021, October 25). In *Wikipedia*. https://en.wikipedia.org/wiki/Feature_(machine_learning)

GeeksforGeeks. (2021, December 22). *LightGBM (light gradient boosting machine)*. https://www.geeksforgeeks.org/lightgbm-light-gradient-boosting-machine/

GeeksforGeeks. (2022, January 13). *Apriori Algorithm*. https://www.geeksforgeeks.org/apriori-algorithm/

Instacart. (2018). *The instacart online grocery shopping dataset 2017* [Dataset]. https://www.instacart.com/datasets/grocery-shopping-2017

Instacart. (2022). *Instacart Company | About Us*. https://www.instacart.com/company/about-us#company-history

*The instacart online grocery shopping dataset 2017 data descriptions*. (2017, January 1). [Github]. Gist. https://gist.github.com/jeremystan/c3b39d947d9b88b3ccff3147dbcf6c6b

J. (2018, May 22). *How Instacart Works: - Uber for X*. Medium. https://medium.com/uber-for-x/how-instacart-works-comprehensive-business-revenue-model-94755495e23e

J. (2022a, April 14). *How Instacart Works: Comprehensive Business & Revenue Model*. JungleWorks. https://jungleworks.com/how-instacart-works-makes-money-revenue-business-model/

Kaushik, S. (2020, October 18). *An introduction to clustering and different methods of clustering*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

M. (2021, September 30). *It's all about the features*. Reality AI. https://reality.ai/it-is-all-about-the-features/

Mandot, P. (2017). *Lead wise tree growth* [Illustration]. Medium.

https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc

Martin, G. (2018, May 20). *load data (reduce memory usage)* [Kaggle]. Kaggle. https://www.kaggle.com/code/gemartin/load-data-reduce-memory-usage/notebook

Molnar, C. (2022, July 12). *5.2 logistic regression | interpretable machine learning*. Github. https://christophm.github.io/interpretable-ml-book/logistic.html

Nahon, A. (2021, December 13). *XGBoost, LightGBM or CatBoost — which boosting algorithm should I use?* Medium. https://medium.com/riskified-technology/xgboost-lightgbm-or-catboost-which-boosting-algorithm-should-i-use-e7fda7bb36bc

Nandakumar, S. (2022, May 13). *Why does lift have a bigger role than confidence in association rules?* Medium. https://blog.devgenius.io/why-lift-has-bigger-role-than-confidence-in-association-rules-619324fc21ab

Pandey, A. (2020, October 5). *How k-means clustering works* [Illustration]. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/10/a-simple-explanation-of-k-means-clustering/

R. (2022b, April 7). *Simple explanation on how decision tree algorithm makes decisions*. Regenerative. https://regenerativetoday.com/simple-explanation-on-how-decision-tree-algorithm-makes-decisions/

R, S. E. (2022, June 21). *Understanding random forest*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2021/06/understanding-random-forest/

Redman 1, R. (2022, February 15). *Study: Instacart fueling online grocery market*. Supermarket News.

https://www.supermarketnews.com/online-retail/study-instacart-fueling-online-grocery-marke
t

Stanley, J. (2018, June 20). *3 Million Instacart Orders, Open Sourced - tech-at-instacart*. Medium.
https://tech.instacart.com/3-million-instacart-orders-open-sourced-d40d29ead6f2