National University of Sciences and Technology (NUST)
School of Electrical Engineering and Computer Science

Department of Computing

CS-404 Big Data Analytics

Project Report

Submitted by:

Sr. no.	Name	Registration no.
1	Shehryar Saqib	347703
2	Aiza Islam	343201

Date: 12th May 2024

Instructor: Dr. Daud Abdullah Asif

Contents

Table of Figures

1. Introduction

In this report, details are provided for the Big Data Analytics Semester Project. It involves Exploratory Data Analysis (EDA) and Big Data techniques being applied to the dataset by Instacart, which is a prominent grocery delivery service. The dataset includes historical purchasing data from numerous users, offering a unique opportunity to analyse consumer behavior and shopping patterns. The primary objectives of this analysis are to uncover common item associations, understand customer buying habits, and predict future purchase behaviours. Such insights are crucial and important for enhancing customer engagement, and optimizing stock management.

To achieve these objectives, the analysis will employ several sophisticated data analytics techniques. It begins with EDA to gain preliminary insights and understand the structure of the data and important patterns. This is followed by applying the Frequent Pattern (FP) growth algorithm and association rule mining to identify commonly purchased items. Additionally, a network graph is constructed to visualize relationships between products. Moreover, a recommender system is developed to suggest products, Principal Component Analysis (PCA) is performed to reduce dimensionality, and implement clustering to segment customers based on purchasing patterns. Additionally, a Page Rank algorithm was also applied to rank the items.

2. Dataset Details

The dataset belongs to Instacart, an online grocery delivery service, and was made publicly available as part of a Kaggle competition. The mentioned dataset can be found on the following link https://www.kaggle.com/c/instacart-market-basket-analysis/overview.

2.1 Dataset Composition:

The dataset has multiple files or subsets. The detailed overview and description of these subsets are given below:

2.1.1 Orders:

This subset contains over 3 million grocery orders from more than 200,000 Instacart users. Each order is anonymized for privacy concerns and linked to a specific user ID. It includes information such as the order sequence number, day of the week, and hour of the day the order was placed.

2.1.2 Products:

Detailed information on 50,000 products grouped into 134 aisles and 21 departments. Each product is identified by a unique product ID, along with its name, aisle, and department.

2.1.3 Order Products:

This is a comprehensive link between the orders and products, detailing which products were purchased in each order. It also indicates whether a product was reordered and the sequence in which products were added to the cart.

2.1.4 Aisles and Departments:

These files provide mappings from aisles and department IDs to their corresponding names, offering a clearer understanding of product categorization within the store.

2.2 Data Characteristics:

2.2.1 Temporal Features:

The dataset includes features such as the time and day of the purchase, providing insights into customer shopping patterns over time. The timestamp is included as a separate column in the concerned subsets of the data.

2.2.2 Categorical Data:

Product, aisle, and department information are primarily categorical, requiring specific analytical approaches suitable for non-numeric data.

2.2.3 High Dimensionality:
With tens of thousands of products and millions of orders, the dataset presents challenges in managing high-dimensional data, necessitating efficient data handling and processing techniques.

3. Use Case Definition

The insights derived from the Instacart dataset can be applied in real-world scenarios, particularly benefiting Instacart or similar businesses in the grocery delivery sector.

3.1 Improved Product Recommendation

Description: By analysing patterns in historical purchasing data, including frequent item associations and customer segmentation, recommendation algorithms can be enhanced. This can lead to more personalized shopping experiences for users.

Business Benefits: Improved recommendations can increase customer satisfaction and loyalty, leading to higher repeat purchase rates and increased sales.

3.2 Efficient Inventory Management

Description: Using insights from product clustering and purchase frequency analysis, stores can optimize their inventory levels. Predictive analytics can forecast demand for products, helping to adjust stock levels more accurately.

Business Benefits: Efficient inventory management reduces the risk of overstocking or stockouts, minimizing costs and ensuring that popular items are always available for customers.

3.3 Marketing Campaigns

Description: By understanding customer buying habits and the relationships between different products, marketing teams can create more effective targeted advertising campaigns. For instance, customers who frequently buy certain items can receive promotions specifically for related products or departments.

Business Benefits: Targeted marketing increases the relevance of advertisements to customers, improving conversion rates and boosting marketing Return on Investment.

3.4 Strategic Decisions for Customer Retention

Description: Clustering analysis can identify specific customer segments based on purchasing patterns, which can be targeted with tailored retention strategies, such as personalized discounts or loyalty programs.

Business Benefits: Effective retention strategies can increase the lifetime value of customers by enhancing their engagement and loyalty.

Each of these use cases demonstrates the practical application of data-driven decision-making in improving various aspects of a grocery delivery service.

4. Methodology

4.1 Used Technologies

Several advanced technologies to handle the various aspects of data processing, analysis, and computation were used. Below is a detailed overview of how each technology was utilized:

4.1.1 Python

Python served as the primary programming language for this project. It was used extensively for data preprocessing and analysis tasks. Python's robust libraries, such as Pandas for data manipulation, Matplotlib and Seaborn for data visualization, facilitated efficient handling and processing of data.

4.1.2 Hadoop

Apache Hadoop was utilized specifically for its capability to process large dataset. In this project, Hadoop was used to implement the PageRank algorithm, to rank products based on their centrality and importance in the purchase network. Hadoop's distributed processing power enabled handling of extensive computations, ensuring scalability and efficiency in processing large-scale data.

4.1.3 Google Colab

Google Colab provided a cloud-based platform that allowed us to leverage high-performance computing resources. By using Google Colab, we were able to run complex scripts and models that required substantial computational power, which would be resource-intensive on local machines. The use of Google Colab enabled access to powerful Google servers, facilitating faster computations and the ability to handle larger datasets seamlessly. It also supported collaboration, allowing team members to work on the project simultaneously in a shared environment.

4.2 Data Preprocessing

The data preprocessing stage is pivotal for ensuring the quality and usability of the dataset for our analysis. This process involved several steps using Python and its the Pandas library, which facilitated efficient data cleaning.

4.2.1 Loading Data:

Three key datasets from CSV files: orders.csv, products.csv, and order_products_train.csv were first loaded. Each of these files plays a crucial role in the analysis by providing detailed information about orders, products, and the linkage between the two.

4.2.2 Merging Data:

To create a comprehensive view of the transactions, these datasets were merged. The order_products_train.csv dataset, which links orders with specific products, was first merged with products.csv to append product details like product name, aisle, and department. Subsequently, this merged dataset was combined with orders.csv to include additional order details such as order number, order day of the week, and hour of the day. This merging was performed using Pandas' merge function, ensuring that data from different tables was accurately aligned by product_id and order_id.

4.2.3 Cleaning Data:

The next step involved cleaning the data, which included checking for missing values and removing duplicates. Using the isnull().sum() function, it was verified that there were no missing values in the merged dataset. Duplicates were removed using drop_duplicates(), ensuring that the dataset was concise and accurate.

4.3 EDA

4.3.1 Top 10 Purchased Products

The bar chart illustrates the most frequently bought items by customers using the Instacart service. It ranks products from the highest to the lowest based on the number of times they were purchased. The visualization shows that bananas and organic bananas are the top two most popular items,

followed by other fresh fruits and vegetables like strawberries and avocados. This chart helps us understand consumer preferences, indicating a strong inclination towards purchasing fresh produce.

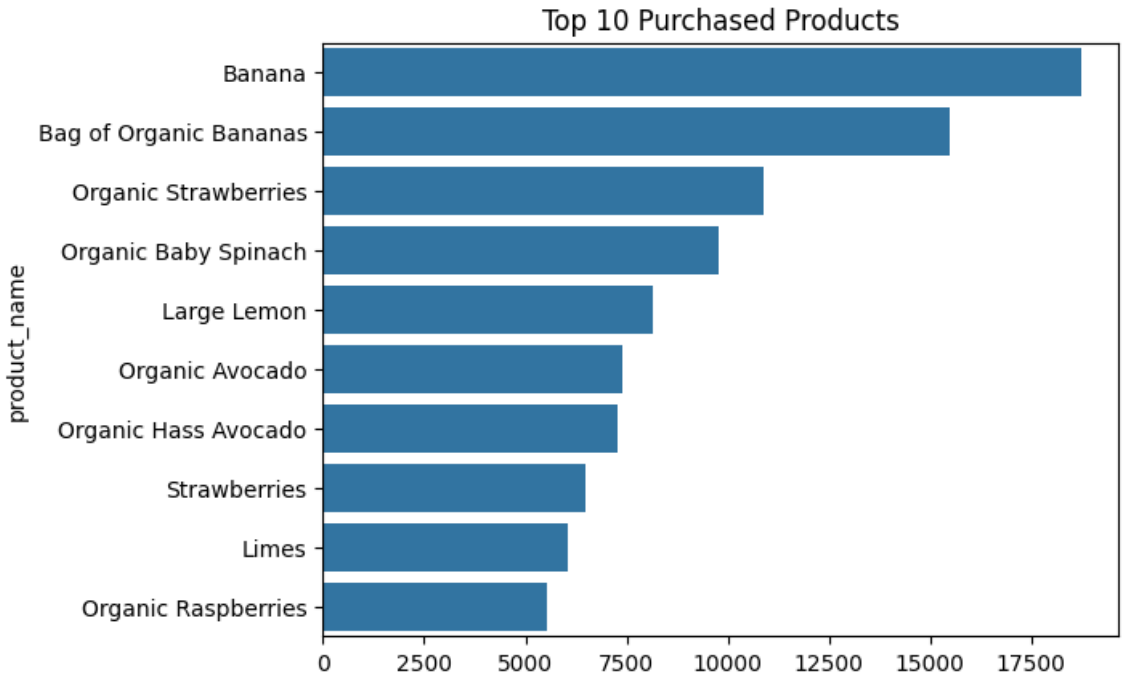

Figure 1: Top 10 purchased products

4.3.2 Frequency of Orders by Day of Week

The bar chart displays the number of orders placed on each day of the week using the Instacart service. From the chart, it is apparent that on Sunday and Monday the highest number of orders placed. The number of orders then tends to decrease slightly and remains relatively steady from Tuesday through Friday. This suggests that customers prefer to do their grocery shopping at the beginning of the week.

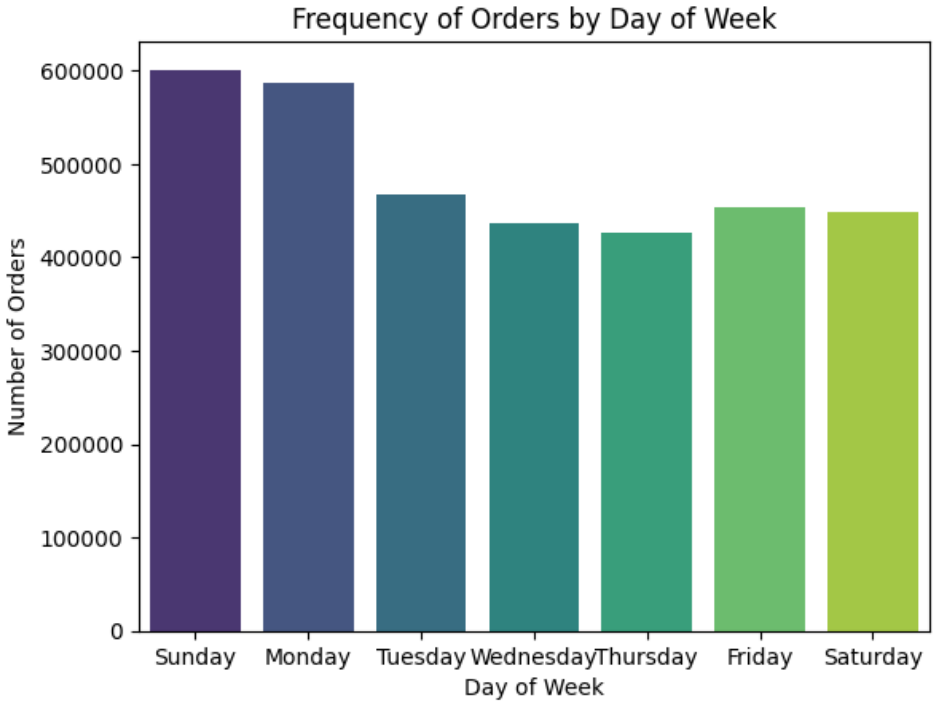

Figure 2: Orders against Day of the Week

4.3.3 Distribution of Add to Cart Order

The histogram illustrates the frequency at which products are added to the shopping cart across different order positions. The chart shows a high frequency for lower numbers, indicating that most products are added to the cart as one of the first few items. The frequency decreases sharply as the add-to-cart order number increases.

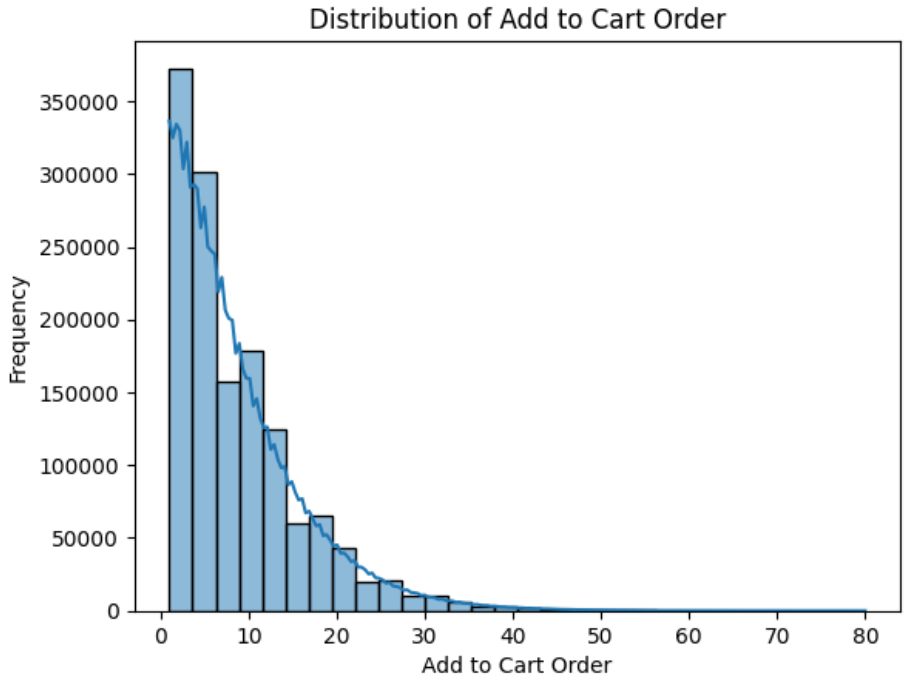

Figure 3: Distribution of Add to Cart Order

4.3.4 Distribution of Orders by Hour of the Day

The histogram shows the number of orders placed at different hours throughout the day. The chart shows increase in order frequency starting from the morning, with a peak during the midday hours, followed by a gradual decrease in the number of orders into the evening. This suggests that the majority of customers prefer to do their grocery shopping during the late morning to early afternoon.

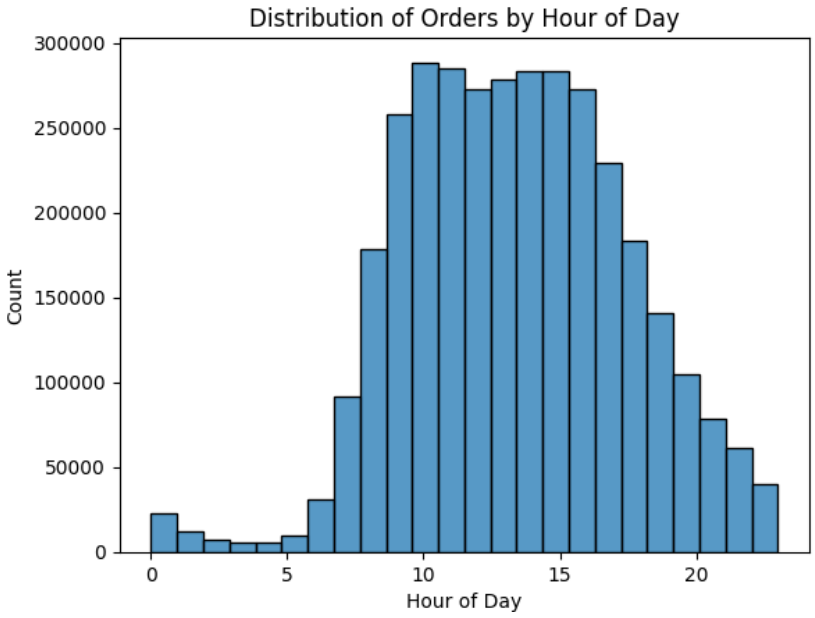

Figure 4: Orders by hour of the day

4.3.5 Correlation Matrix of Numerical Variables

The heatmap indicates minimal correlation between the variables in the Instacart dataset, such as order day, order hour, days since prior order, and add-to-cart sequence. The coefficients suggest that these variables largely work independently.

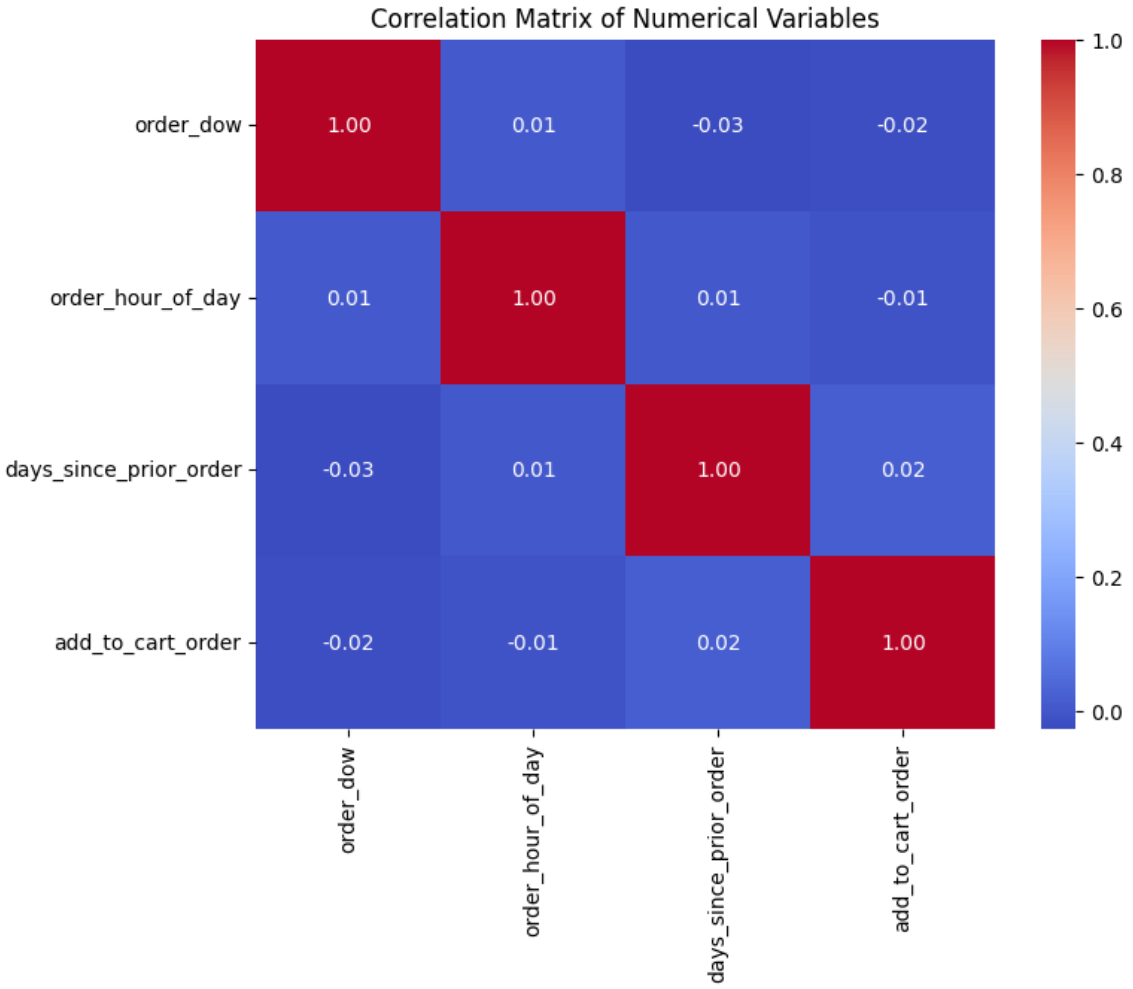

Figure 5:Correlation Matrix

4.3.6 Average Basket Size by Day of the Week

The line graph shows variations in the number of items per order throughout the week. The graph indicates that the average basket size starts relatively high on Sunday, decreases to its lowest point on Wednesday, and then sharply increases again towards Saturday.

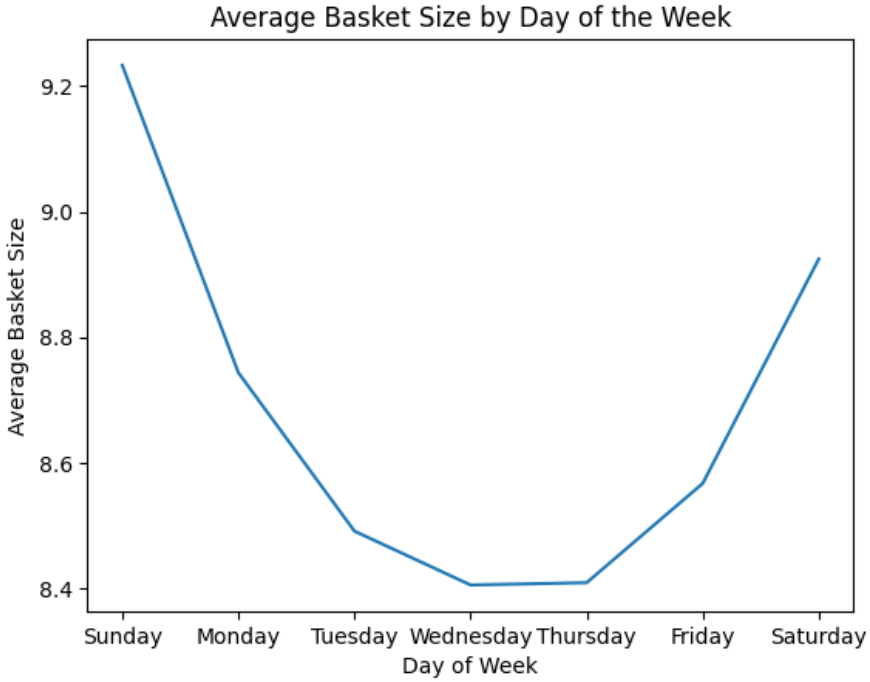

Figure 6: Average basket size by Day of the Week

4.3.7 Average Basket Size by Hour of the Day

The line graph illustrates the fluctuations in the number of items per order at different times throughout the day. The graph shows a significant peak early in the morning around 5 AM, followed by a sharp decline and another peak around 8 PM. This pattern suggests that early morning and late evening shoppers tend to purchase more items per order.

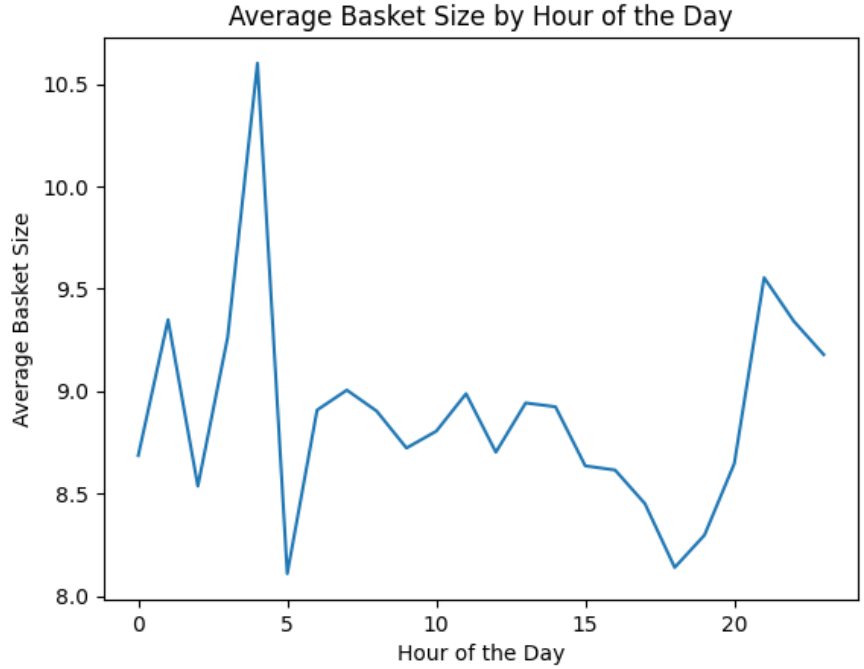

Figure 7: Average Basket Size by Hour of the Day

4.3.8 Average Add to Cart Order by Hour of the Day

The scatter illustrates how the average position at which items are added to shopping carts varies throughout the day. Notably, early morning and late evening show higher positions, indicating items are added later in the shopping process.

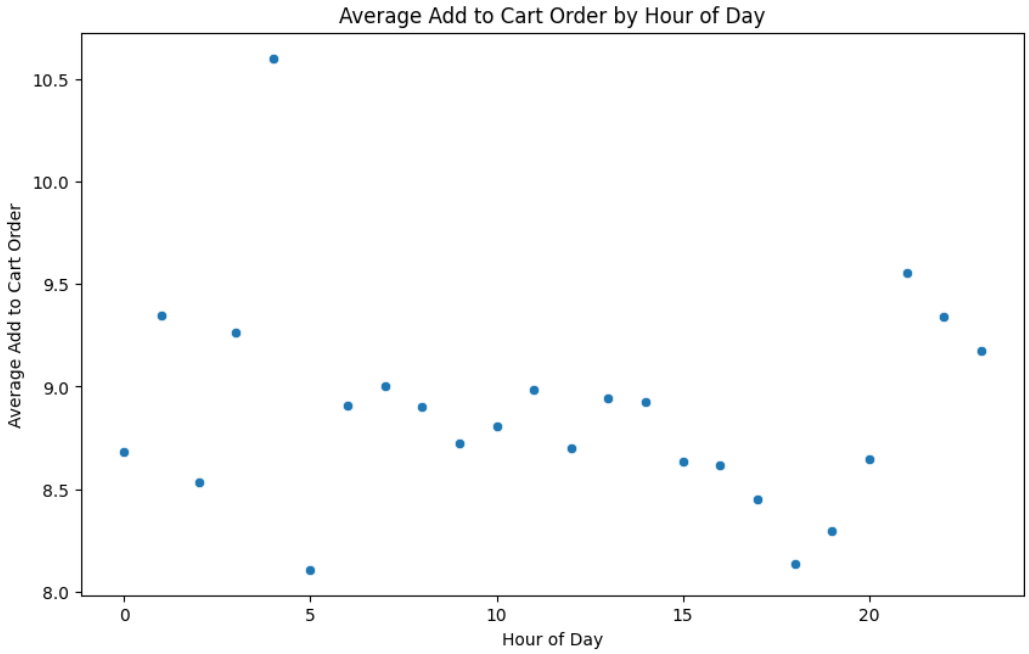

Figure 8: Average Add to Cart Order by Hour of the Day

4.3.9 Days Since Prior Order by Hour of the Day

The plot shows no variation in the days since the last order across different hours of the day. Each hour of the day is consistent across all ranges of days since the last order indicating that the timing of orders throughout the day does not depend on how long it has been since the customer's previous order.

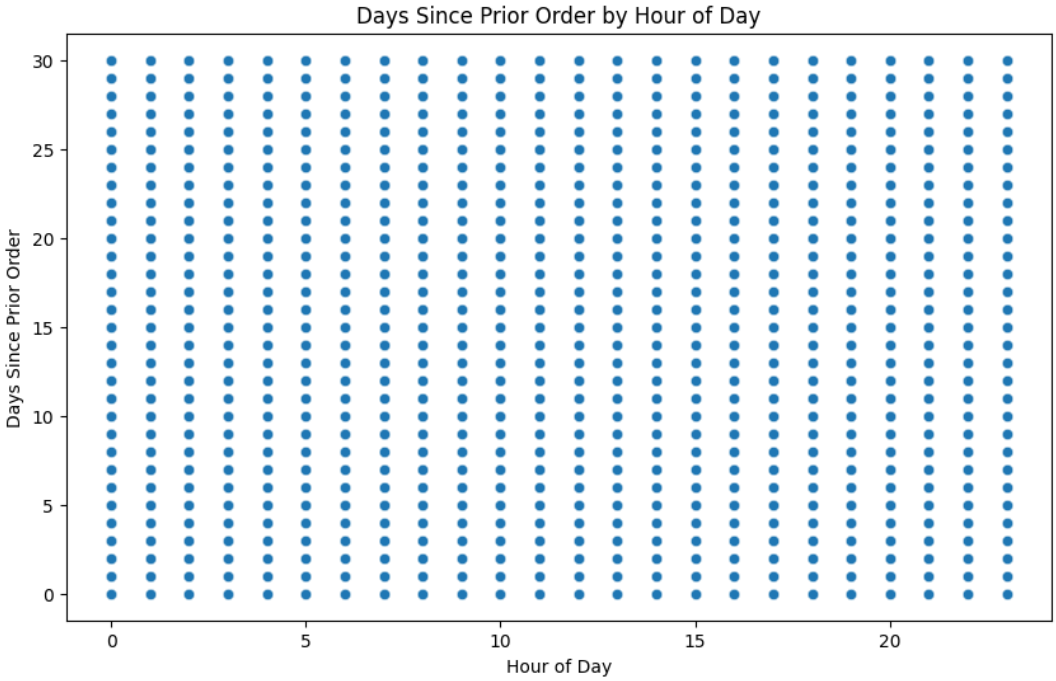

Figure 9: Days Since Prior Order by Hour of the Day

4.4 Technique 1| Apriori Algorithm

The Apriori algorithm is a classical approach used in data mining for identifying the itemsets that appear frequently together. It is prominently used in Market Basket Analysis.

4.4.1 Steps:

1. **Data Preparation:** A list of transactions is created where each list contains the items purchased together, grouped by order_id. The list is transformed using a TransactionEncoder, which converts the list into a one-hot encoded DataFrame suitable for analysis.
2. **Applying the Algorithm:** The apriori function is used with a defined minimum support threshold to find all itemsets that appear frequently. This function considers only those itemsets that appear in at least a specified percentage (e.g., 1%) of all transactions.
3. **Rule Generation:** Using the association_rules function, generate rules from these itemsets based on 'confidence'. This step identifies which items are likely to be bought together.

4.4.2 Visualizations: Visualize The distribution of 'confidence' and 'lift' for the rules using histograms is visualized. This helps in understanding the strength of the associations.

Figure 10: Distribution of Confidence for Apriori

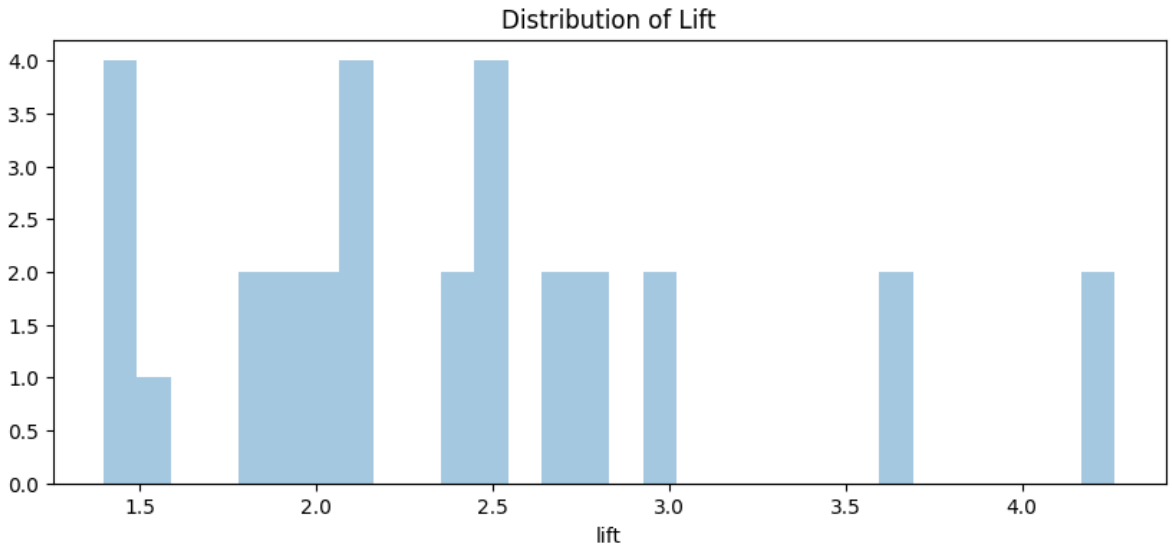

Figure 11:Distribution of Lift for Apriori

4.5 Technique 2| FP Growth

FP-Growth is an efficient alternative to Apriori that uses a tree structure to store compressed information about itemsets, allowing it to find frequent itemsets without candidate generation.

4.5.1 Steps

1. **Data Preparation:** The same transaction list and encoding are used as with the Apriori, prepared to fit the FP-Growth requirements.
2. **Applying the Algorithm**: The fpgrowth function finds frequent itemsets directly from the encoded DataFrame using a minimum support threshold.
3. **Rule Generation:** Rules are generated from these frequent itemsets using the association_rules function with the 'confidence' metric. It identifies strong rules without the exhaustive searching required by Apriori.

4.5.2 Visualizations

1. **Scatter Plot:** Association rules derived from FP-Growth are plotted to show 'support' versus 'confidence'. This scatter plot helps visualize the strength and prevalence of rules.

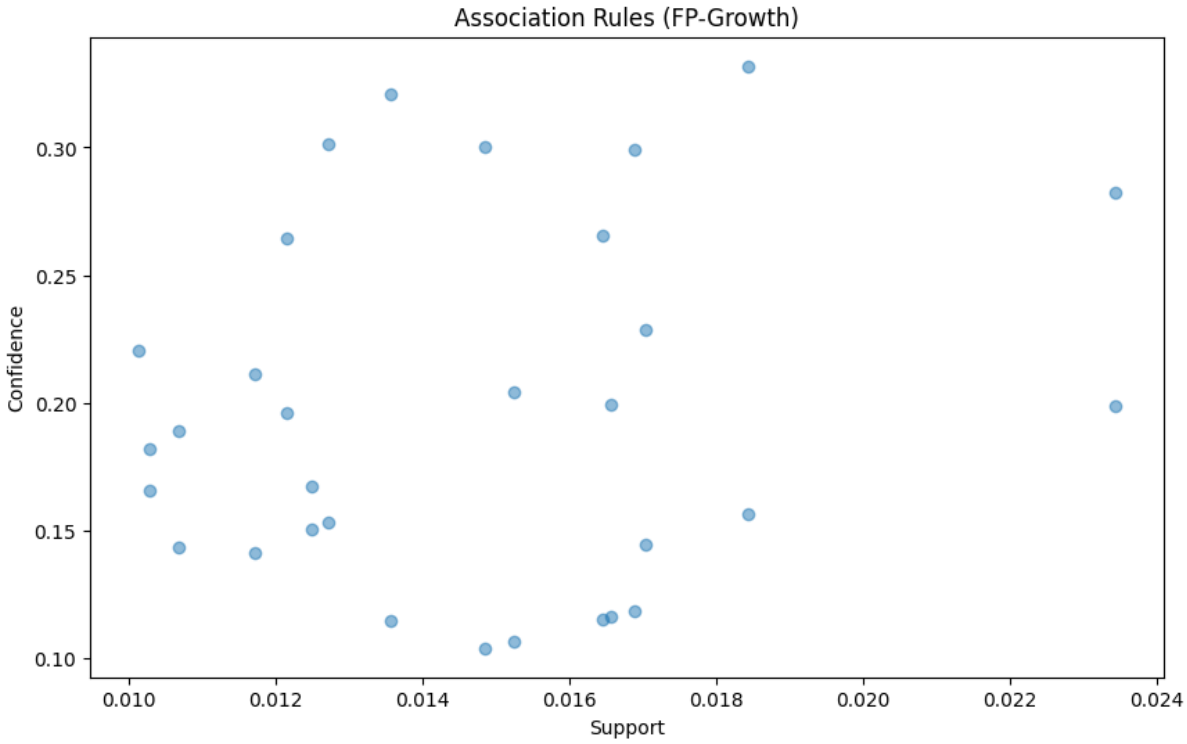

Figure 12: Scatter plot for FP growth

2. **Network Graph:** A directed graph is made to visualize how items are connected based on the association rules. This network graph helps in visually understanding the relationships between items.

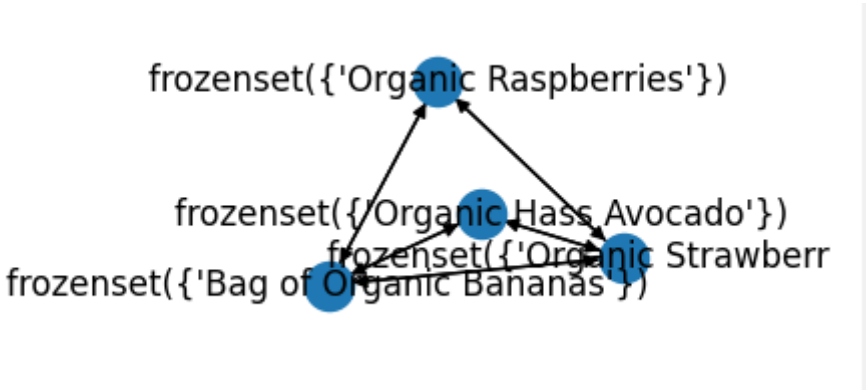

Figure 13: Network plot for FP growth

4.6 Technique 3| Recommender System

Recommender system is to provide personalized product suggestions to users based on their historical purchase data and similarities to other users' preferences. This approach helps in enhancing user experience and potentially increasing sales by making relevant product recommendations.

4.6.1 Data Preparation and Analysis

From the entire dataset, the top 50 most frequently purchased products are identified, and a random 10% sample of users is selected. This simplifies the data and focuses the analysis on the most relevant items. A pivot table is created with users as rows and products as columns. Entries in this matrix indicate whether a user has reordered a product, capturing user preferences.

4.6.2 Item-User Matrix Transformation

The item-user matrix is converted into a sparse format using the CSR (Compressed Sparse Row) method to optimize memory usage and computational efficiency, especially important when dealing with large datasets.

4.6.3 Similarity Calculations

Cosine similarity is computed for the item-user sparse matrix to determine how similar products are based on user purchasing patterns. This similarity metric helps identify which products tend to be bought together, reflecting shared user preferences. A heatmap is used to visualize the item similarity scores among the top 50 items, helping to quickly identify strong associations between products.

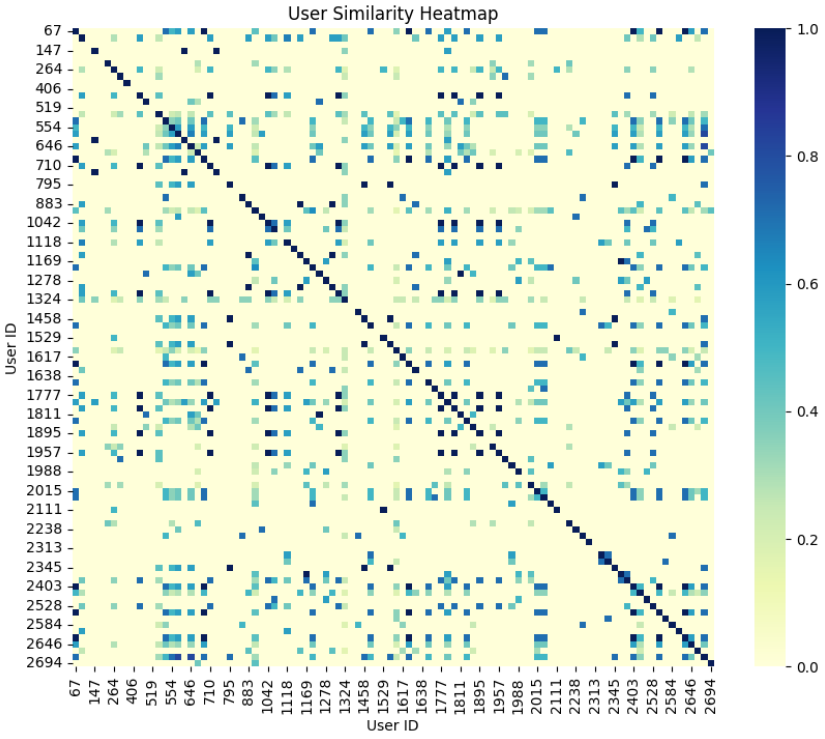

Figure 14: User Similarity Heatmap

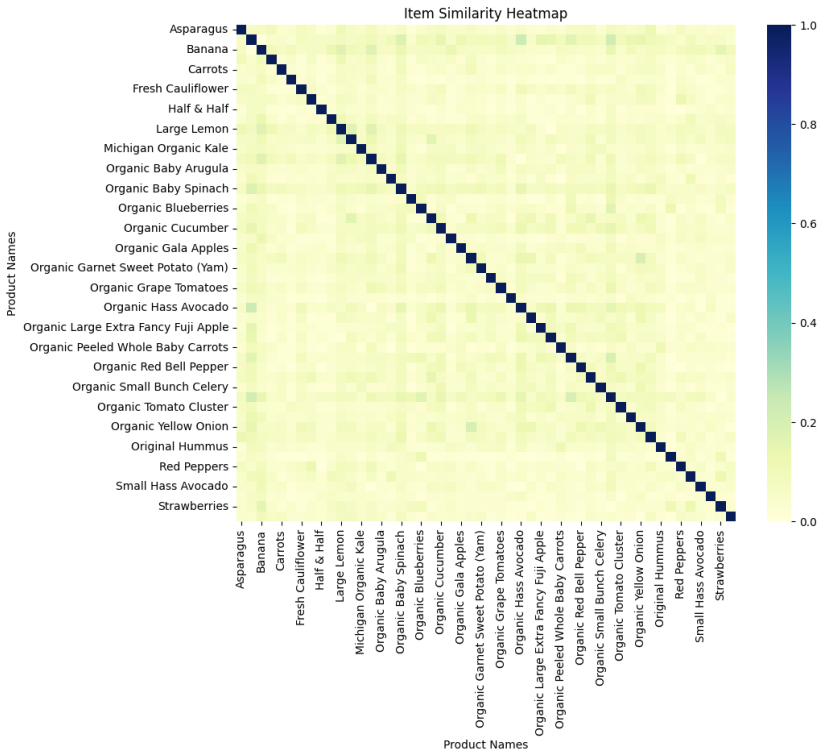

Figure 15: Item Similarity Heatmap

4.6.4 Recommendation Engine

A function is developed to recommend products to a user. This function calculates recommendations based on item similarity scores and user's purchasing history. For each product a user has bought, the function retrieves similar products from the item similarity DataFrame. It aggregates the scores for these similar products, ensuring that already purchased items are excluded from the recommendations. The top 'n' products with the highest aggregated similarity scores are recommended.

4.6.5 Final Output

The recommender system outputs a list of products that a user might be interested in, based on similarity to their past purchases and the purchasing behavior of similar users.

```
# Example: Recommend items for user_id 264
print(recommend_items(264, item_similarity_df, reduced_data, top_n=5))
# Example: Recommend items for user_id 406
print(recommend_items(406, item_similarity_df, reduced_data, top_n=5))
```

```
['Large Lemon', 'Organic Baby Spinach', 'Organic Strawberries', 'Honeycrisp Apple', 'Limes']
['Organic Yellow Onion', 'Organic Red Onion', 'Organic Cilantro', 'Large Lemon', 'Red Peppers']
```

Figure 16: Output of Recommender System

4.7 Technique 4| PCA and Clustering for Customer Segmentation

Principal Component Analysis (PCA) combined with clustering techniques like K-Means in customer segmentation uncovers underlying patterns in customer purchasing behaviors, reduce the complexity of data, and identify distinct groups or segments that share similar characteristics. This approach is vital for targeted marketing, improving customer service, and enhancing strategic business decisions.

4.7.1 Steps

1.Data Preparation: Multiple data sources including user transactions, product details, and order information are merged to form a comprehensive dataset that captures various aspects of customer behavior.

2.Applying PCA: PCA is utilized to reduce the dimensionality of the dataset, which originally includes a large number of aisles (features). This reduction is achieved by transforming the original variables into a new set of variables (principal components), which are linear combinations of the original variables. The principal components are chosen in such a way that they capture the maximum variance within the dataset, with the first few components usually capturing the majority of the information.

3.Clustering with K-Means: After PCA, the K-Means clustering algorithm is applied to the transformed dataset. K-Means identifies clusters based on the proximity of data points to the centroid of clusters, which minimizes the variance within each cluster.

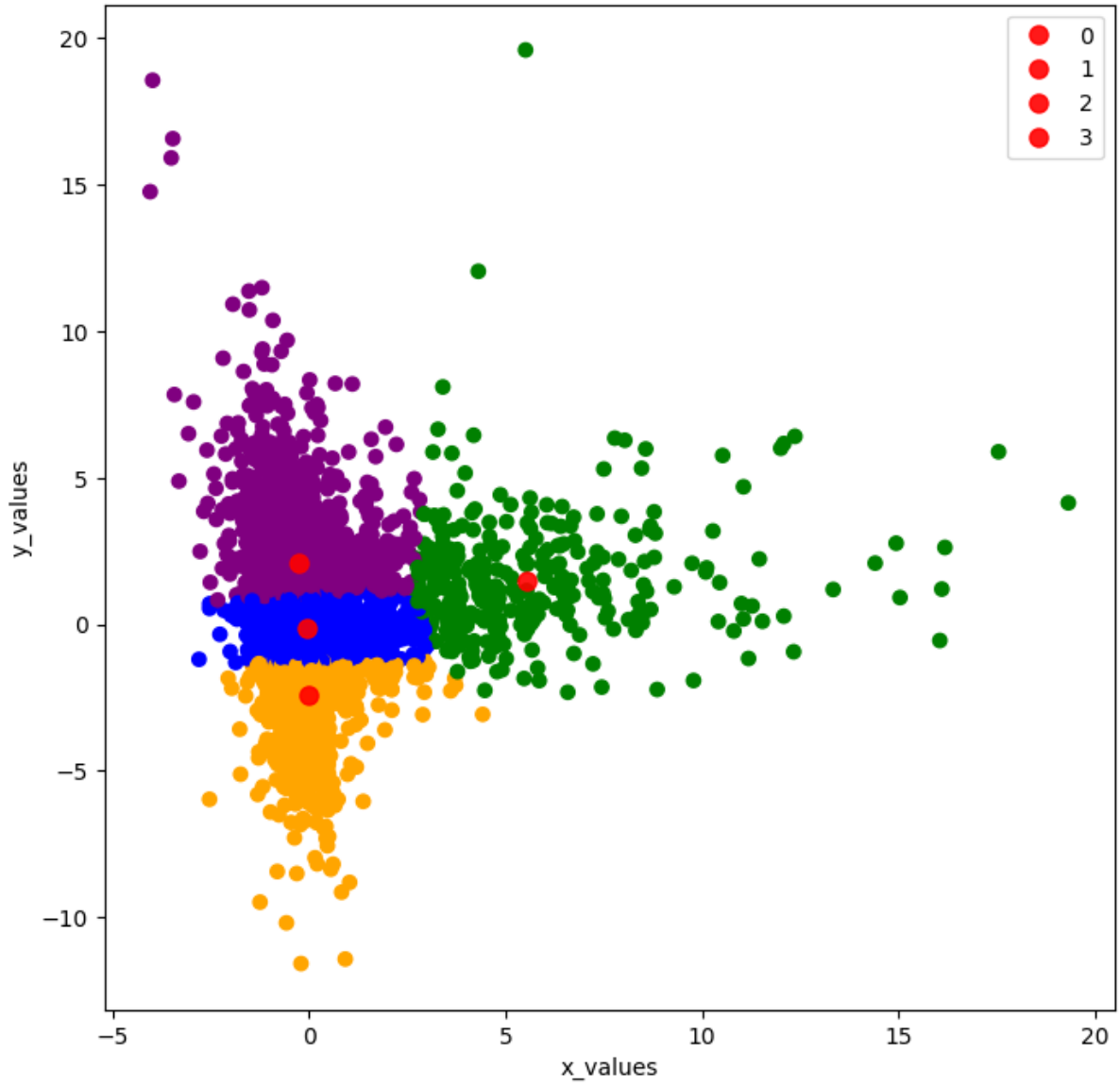

Figure 17: Clustering using KMeans

4.Analyzing Cluster Output: Each cluster is analysed to determine the defining characteristics of the segment it represents. This visualization aids in understanding how each segment differs from others in terms of product preferences.

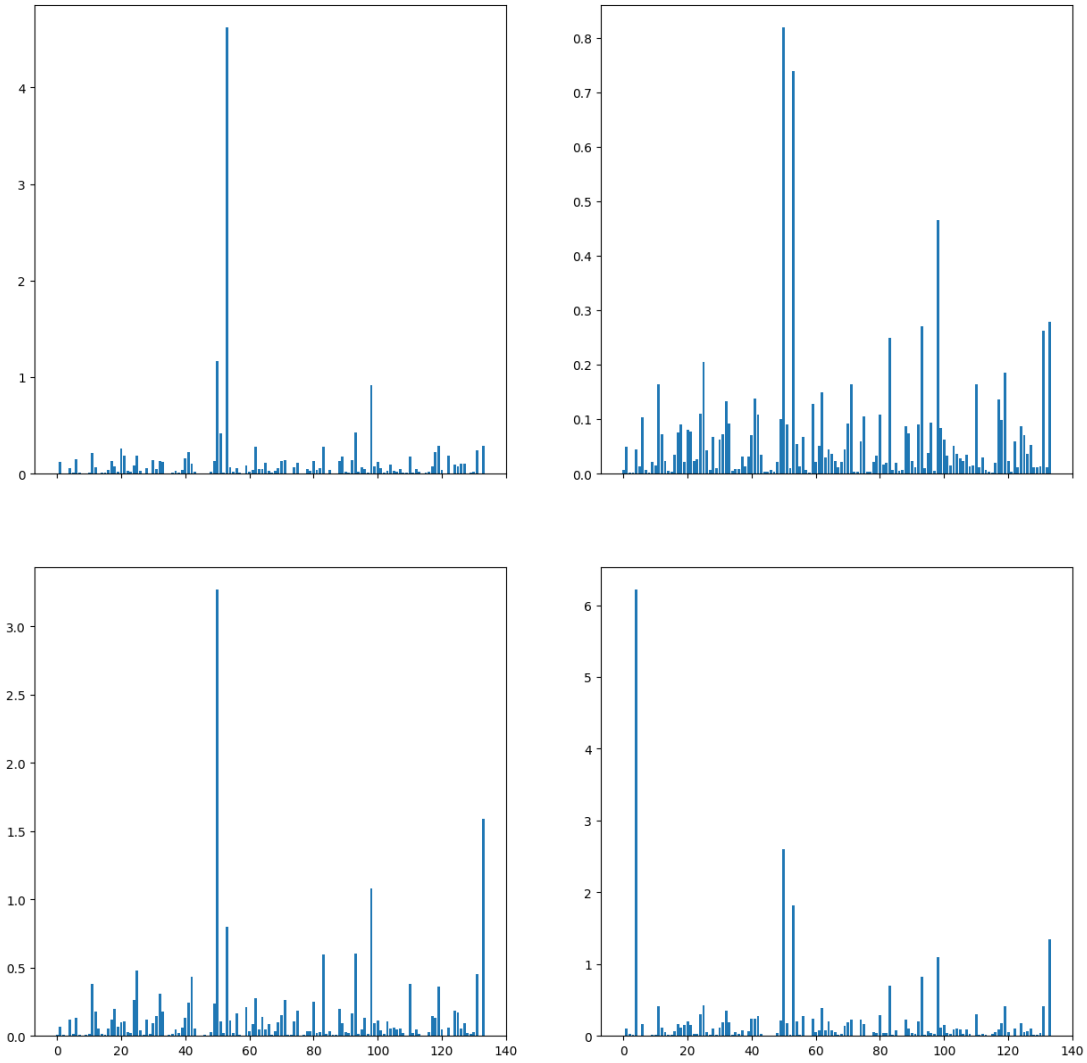

Figure 18:Analysing clusters

4.8 Technique 5| PageRank using Hadoop

PageRank algorithm is applied to the dataset. Instead of ranking web pages, some changes are made in the algorithm to analyze how frequently each day of the week appears in the orders data. The Java code is a simplified form of PageRank algorithm to days of the week based on orders data. Here,DayOfWeekMapper class is responsible for mapping each record of input data to key-value pairs. In this case, it extracts the day of the week from each record and emits it as a key along with the value 1. DayOfWeekReducer class takes the output of the mapper, which is a key-value pair with the day of the week as the key and a list of 1s as the value, and sums up the counts for each day of the week.

4.8.1 Code of Mapper and Reducer

```
import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;
```

```
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Job;
import org.apache.hadoop.mapreduce.Mapper;
import org.apache.hadoop.mapreduce.Reducer;
import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;
import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;
import java.io.IOException;
public class DayOfWeekPageRank {
   public static class DayOfWeekMapper extends Mapper<Object, Text, Text, IntWritable> {
      private Text dayPair = new Text();
      private final static IntWritable one = new IntWritable(1);
      public void map(Object key, Text value, Context context) throws IOException,
InterruptedException {
         String[] fields = value.toString().split("\t");
         if (fields.length > 3) {
            String orderDow = fields[4]; // Assuming order_dow is the 5th field
            dayPair.set(orderDow);
            context.write(dayPair, one);
         }
      }
   }
   public static class DayOfWeekReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
      private IntWritable result = new IntWritable();
      public void reduce(Text key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
         int sum = 0;
         for (IntWritable val : values) {
            sum += val.get();
         }
         result.set(sum);
         context.write(key, result);
```

```
    }
  }
  public static void main(String[] args) throws Exception {
    Configuration conf = new Configuration();
    Job job = Job.getInstance(conf, "Day of Week PageRank");
    job.setJarByClass(DayOfWeekPageRank.class);
    job.setMapperClass(DayOfWeekMapper.class);
    job.setCombinerClass(DayOfWeekReducer.class);
    job.setReducerClass(DayOfWeekReducer.class);
    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(IntWritable.class);
    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));
    System.exit(job.waitForCompletion(true) ? 0 : 1);
  }
}
```

4.8.2 Steps

1. Compiled code

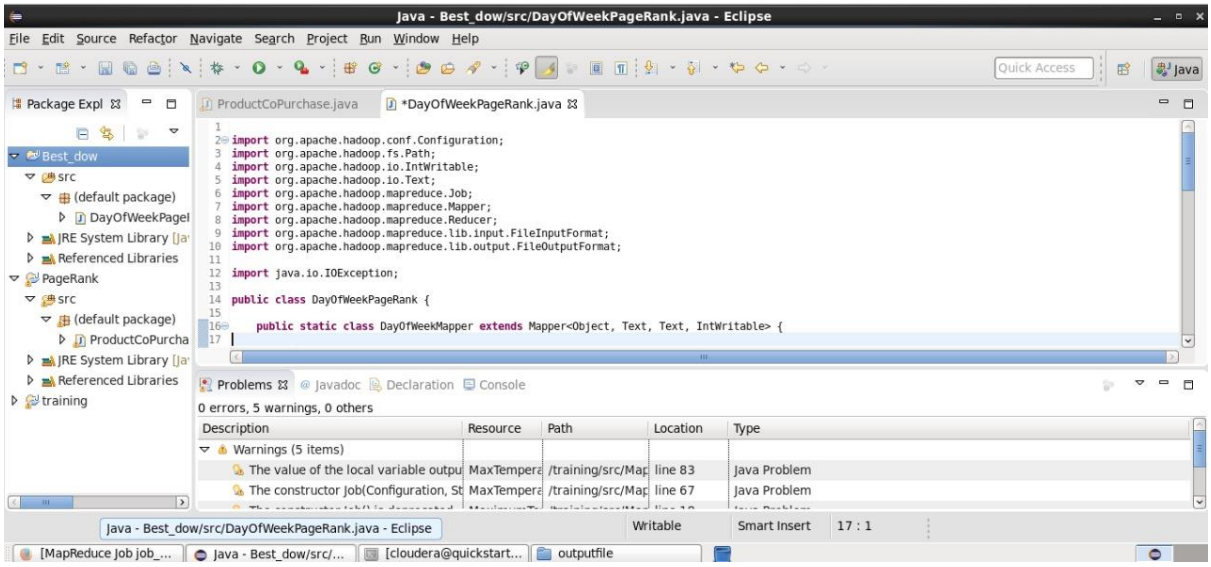

Figure 19:Compiled code

2. Created a JAR file:

Figure 20:JAR file

3. Running the JAR file:

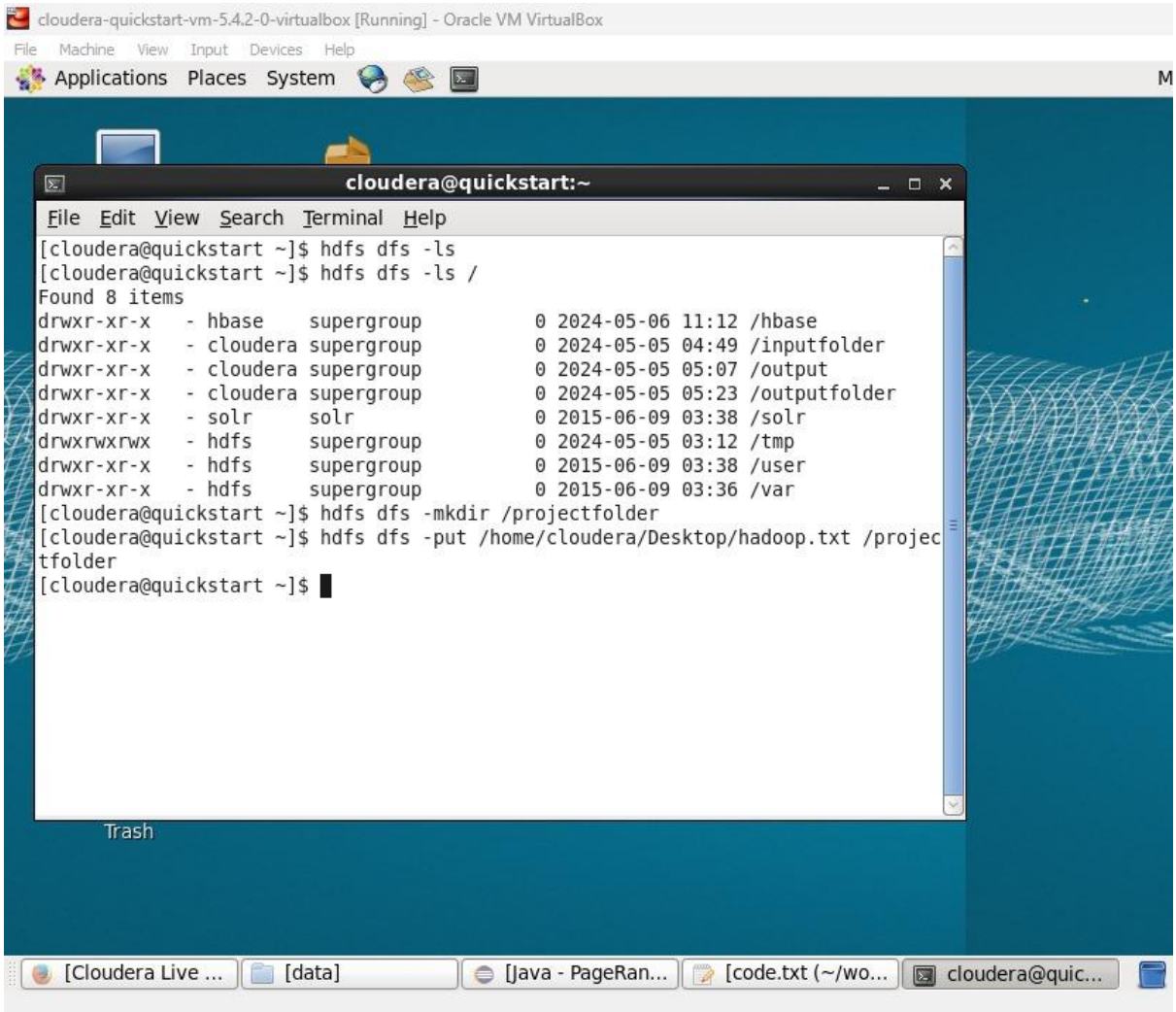

Figure 21: Running JAR file through cloudera

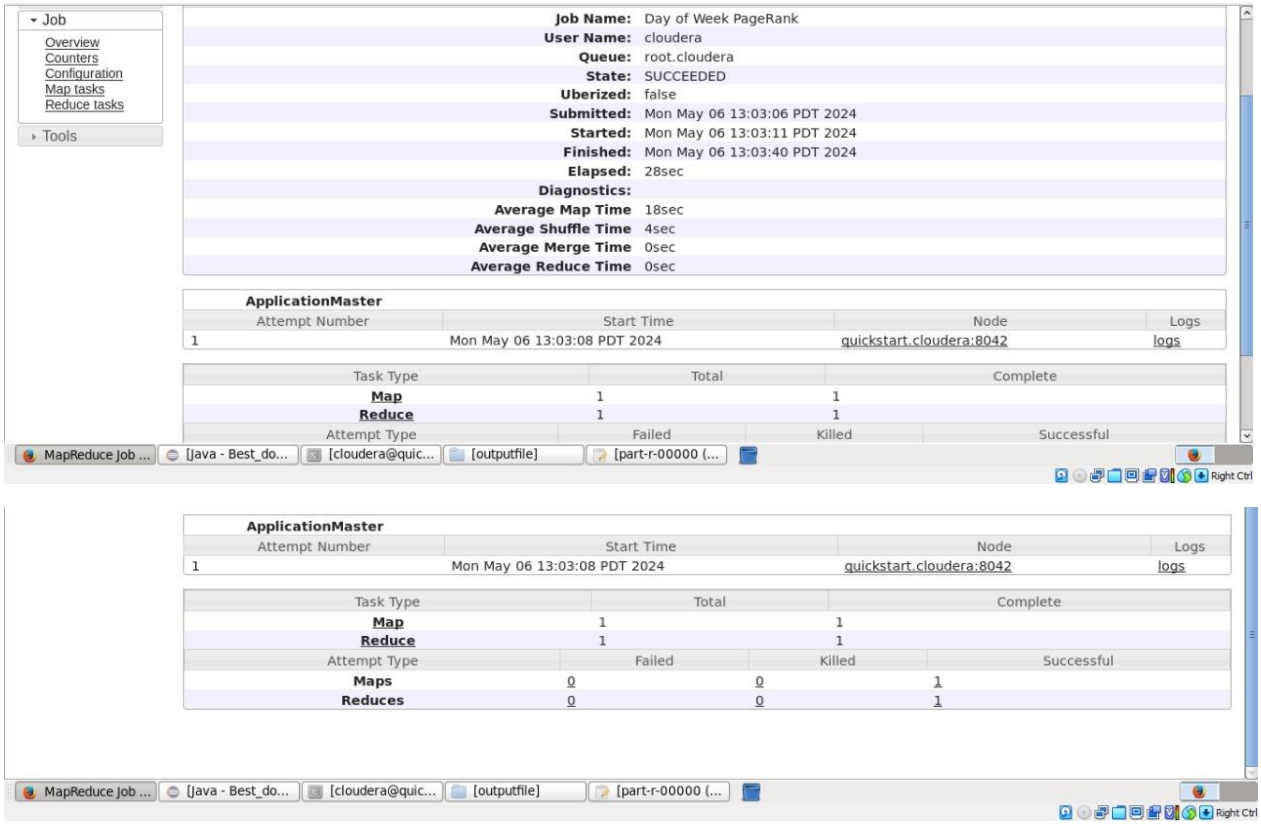

Figure 22: Statistics of MapReduce job

4. Output:

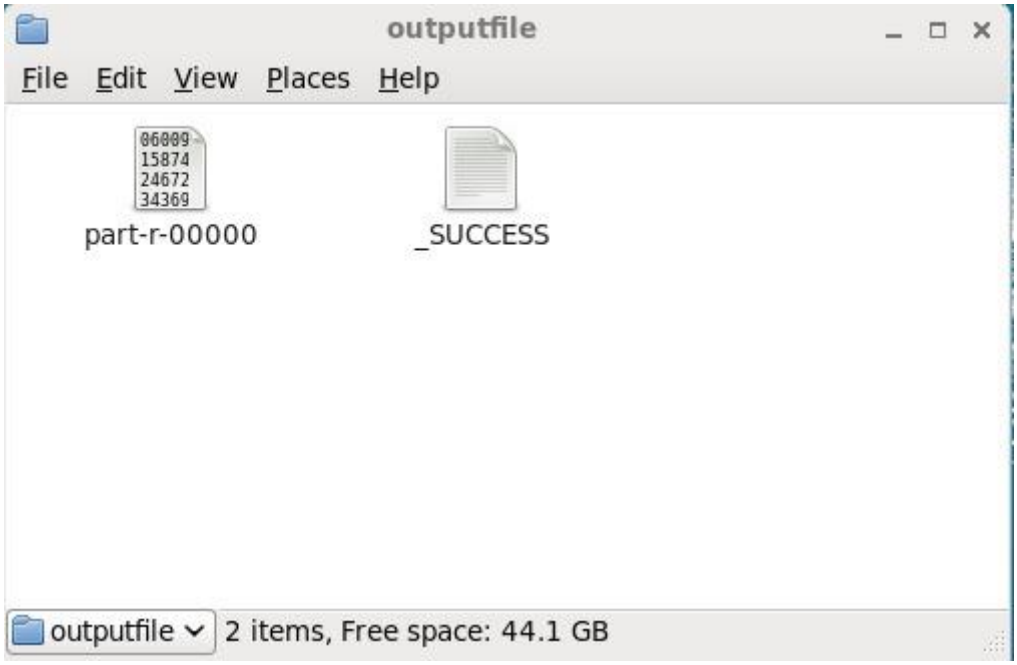

Figure 23:Output file

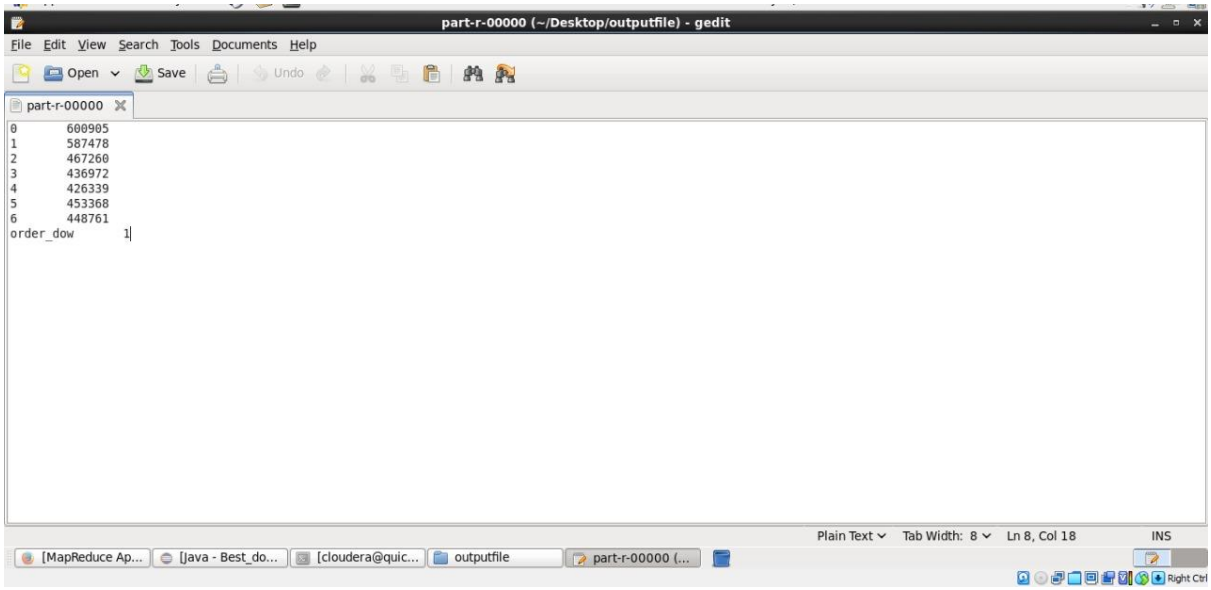

Figure 24: Output file content

4.8.2 Results Explanation

Row 0: 600905 orders were placed on day 0 (Sunday).

Row 1: 587478 orders were placed on day 1 (Monday).

Row 2: 467260 orders were placed on day 2 (Tuesday).

Row 3: 436972 orders were placed on day 3 (Wednesday).

Row 4: 426339 orders were placed on day 4 (Thursday).

Row 5: 453368 orders were placed on day 5 (Friday).

Row 6: 448761 orders were placed on day 6 (Saturday).

The order_dow values are listed in ascending order, indicating how the MapReduce job sorted the keys before passing them to the Reducer.

The output can be used to identify which days of the week are busiest. It is evident that Sunday had the most orders, and Thursday had the fewest orders.

5. Analysis of Use Case after BDA Techniques

The ability to segment customers in a retail setting has greatly improved with the use of BDA techniques like FP growth, Apriori, Recommender System, PCA and K-Means clustering. We have obtained a deeper understanding of a wider range of purchasing behaviours. Because of this deeper comprehension, more focused marketing campaigns and better inventory control could now be possible, guaranteeing that product availability reflects consumer preferences and demand trends. In a competitive market environment, these analytics-driven initiatives increase customer satisfaction through personalised marketing strategies and enhanced operational efficiencies, leading to an increase in loyalty and retention.

6. Problems Faced

1. The effectiveness of PCA and clustering depends heavily on the quality and completeness of the data. Missing values, incorrect data entries, and inconsistencies in the data significantly impact the accuracy of the segmentation.
2. As the volume of data increases, the computational load also increases. This leads to performance issues, requiring more sophisticated solutions to handle large-scale data processing efficiently.
3. The recursive nature of the FP-Growth algorithm can lead to performance issues, especially with datasets that have many complex and frequent item patterns as seen in the network diagram.

7. Future Work

1. Incorporating hybrid approaches in your recommender systems that combine collaborative filtering, content-based filtering, and even demographic information to improve recommendation accuracy and tackle the cold start problem more effectively.
2. Enhancing your models by integrating more diverse datasets, such as social media behavior, geographic information, or economic indicators that may influence purchasing patterns.
3. Continuously refining and optimizing the existing algorithms for better performance and scalability, possibly by tuning hyperparameters or employing more efficient data structures.
4. Developing interactive tools and dashboards that allow end-users to explore the data, understand the recommendations, and possibly control the kind of data they are willing to share or the recommendations they wish to receive.

8. Conclusion

In conclusion, the integration of advanced data analytics techniques such as the FP-Growth algorithm, Apriori algorithm for association rules, and Recommender Systems into the project has substantially enhanced our understanding and segmentation of customer behaviours and preferences.

By continually evolving and adapting to the latest technological and methodological innovations, the project is well-positioned to remain at the forefront of the data analytics domain, providing actionable insights and fostering data-driven decision-making processes.