EXPLORATORY ANALYSIS OF GROCERY STORE DATA USING MARKET BASKET ANALYSIS

Major Project Report submitted in partial fulfillment of the requirements for the award of the Degree of B.E in Computer Science and Engineering

By

Dhannarapu Reshmika	160619733011
Yelgoe Vyshnavi	160619733059
Sreeja Sogala	160619733044

Under the Guidance of

Dr Y V S S Pragathi
Prof & HoD, Dept of CSE,
Department of Computer Science & Engineering

Department of Computer Science and Engineering Stanley College of Engineering & Technology for Women (Autonomous)

Chapel Road, Abids, Hyderabad – 500001
(Affiliated to Osmania University, Hyderabad, Approved by AICTE, Accredited by NBA & NAAC with A Grade)
2023

Stanley College of Engineering & Technology for Women (Autonomous)
Chapel Road, Abids, Hyderabad – 500001
(Affiliated to Osmania University, Hyderabad, Approved by AICTE,
Accredited by NBA & NAAC with A Grade)

CERTIFICATE

This is to certify that major project report entitled Exploratory Analysis of Grocery Store Data Using Market Basket Analysis being submitted by

Dhannarapu Reshmika	160619733011
Yelgoe Vyshnavi	160619733059
Sreeja Sogala	160619733044

in partial fulfillment for the award of the Degree of Bachelor of Engineering in Computer Science & Engineering to the Osmania University, Hyderabad is a record of bonafide work carried out under my guidance and supervision. The results embodied in this project report have not been submitted to any other University or Institute for the award of any Degree or Diploma.

Guide
Dr Y V S S Pragathi
Prof & HoD, Dept of CSE

Head of the Department
Dr Y V S S Pragathi
Prof & HoD, Dept of CSE

Project Coordinator
Dr. B.V. Ramana Murthy
Vice Principal, Prof. Dept Of CSE

K.Srilatha
Assistant Professor, Dept of CSE

External Examiner

DECLARATION

We hereby declare that major project work entitled **Exploratory Analysis of Grocery Store Data Using Market Basket Analysis** submitted to the Osmania University, Hyderabad, is a record of original work done by us. This project work is submitted in partial fulfilment of the requirements for the award of the degree of the B.E in Computer Science and Engineering.

Dhannarapu Reshmika	160619733011
Yelgoe Vyshnavi	160619733059
Sreeja Sogala	160619733044

ACKNOWLEDGEMENT

First of all, we are grateful to **The Almighty God** for establishing us to complete this Major Project. We pay our respects and love to our parents and all our family members and friends for their love and encouragement throughout our career.

We wish to express our sincere thanks to Sri. Kodali Krishna Rao, Correspondent and Secretary Stanley College of Engineering & Technology for Women, for providing us with all the necessary facilities.

We place on record, our sincere gratitude to Prof. Satya Prasad Lanka, Principal, for his constant encouragement.

We deeply express our sincere thanks to our Head of the Department, Prof Y V S S Pragathi, for encouraging and allowing us to present the Major Project on the topic **Exploratory Analysis of Grocery Store Data Using Market Basket Analysis** at our department premises for the partial fulfillment of the requirements leading to the award of the B.E. degree.

It is our privilege to express sincere regards to our project guide Prof Y V S S Pragathi, for the valuable inputs, able guidance, encouragement, whole-hearted co-operation and constructive criticism throughout the duration of our project.

We take this opportunity to thank all our faculty, who have directly or indirectly helped our project. Last but not least, we express our thanks to our friends for their co-operation and support.

STANLEY COLLEGE OF ENGINEERING AND TECHNOLOGY FOR WOMEN

Chapel Road, Abids, Hyderabad.

Department of Computer Science & Engineering

Institute Vision:

Empowering girl students through professional education integrated with values and character to make an impact in the World.

Institute Mission:

M1: *Enabling quality engineering education for girl students to make them competent and confident to succeed in professional practice and advanced learning.*

M2: *Providing state-of-art-facilities and resources towards world class education.*

M3: *Integrating qualities like humanity, social values, ethics, leadership towards their contribution to society.*

Department Vision:

Empowering girl students with the contemporary knowledge in Computer Science and Engineering for their success in life.

Department Mission:

M1: *To impart quality education for girl students to learn and practice various hardware and software platforms prevalent in industry.*

M2: *To achieve self-sustainability and overall development through research and development activities.*

M3: *To provide education for life by focusing on the inculcation of human & moral values through and honest and scientific approach*

M4: *To groom students with good attitude, team work and personality skills.*

Programme Educational Objectives

PEO1: Our graduates shall have enhanced skills and contemporary knowledge in software and hardware technologies for professional excellence, towards successful employment, advanced learning and research.

PEO2: Our graduates shall have life-long learning attitude, innovation and creativity to master latest technologies, devise solutions for realistic and social issues in the society.

PEO3: Our graduates shall have good attitude and personality skills, ethical values, teamwork and leadership skill towards professionalism and ethical practices within the organization and the society.

Program Outcomes:

PO1: Engineering Knowledge: Apply knowledge of mathematics and science, with fundamentals of Computer Science & Engineering to be able to solve complex engineering problems related to CSE.

PO2: Problem Analysis: Identify, Formulate, review research literature and analyze complex engineering problems related to CSE and reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.

PO3: Design/Development of solutions: Design solutions for complex engineering problems related to CSE and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety and the cultural societal and environmental considerations.

PO4: Conduct Investigations of Complex problems: Use research–based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tool Usage: Create, Select and apply appropriate techniques, resources and modern engineering and IT tools including prediction and modeling to computer science related complex engineering activities with an understanding of the limitations.

PO6: The Engineer and Society: Apply Reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the CSE professional engineering practice.

PO7: Environment and Sustainability: Understand the impact of the CSE professional engineering solutions in societal and environmental contexts and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply Ethical Principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team Work: Function effectively as an individual and as a member or leader in diverse teams and in multidisciplinary Settings.

PO10: Communication: Communicate effectively on complex engineering activities with the engineering community and with society at large such as able to comprehend and with write effective reports and design documentation, make effective presentations and give and receive clear instructions.

PO11: Project Management and Finance: Demonstrate knowledge and understanding of the engineering management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multi disciplinary environments.

PO12: Life-Long Learning: Recognize the need for and have the preparation and ability to engage in independent and life-long learning the broadest context of technological change.

Program Specific outcomes:

PSO1: Problem-Solving Skills: The ability to apply standard practices and strategies in software project development using open-ended programming environments to deliver a quality product for the benefit of students.

PSO2: Design, Implement, Test and Evaluate a computer system, component or algorithm to meet desired needs and to solve a computational problem.

Major Project Expected Program Outcomes

PO2: PROBLEM ANALYSIS

PO3:DESIGN/DEVELOPMENT OF SOLUTION

PO5: MODERN TOOLS USAGE

PO6: THE ENGINEER AND SOCIETY

PO9: INDIVIDUAL AND TEAM WORK

PO11: PROJECT MANAGEMENT

PO12: LIFELONG LEARNING

Program Specific outcomes

PS01: PROBLEM-SOLVING SKILLS

PSO2: DESIGN, IMPLEMENT, TEST

ABSTRACT:

Data mining has become a widely accepted process for organizations to enhance their organizational performance and gain a competitive advantage. Because the data mining process is a relatively new concept, it has been defined in various ways by various authors in the recent past. Data mining allows managers to make more knowledgeable decisions by predicting future trends and behaviors.

One of the most widely used areas of data mining for the retail industry is in marketing. Market basket analysis is a marketing method used by many retailers to determine the optimal locations to promote products. It is a powerful tool for the implementation of cross-selling strategies. As we know that Market Basket Analysis is an example of Association rules in data mining, which is one of the data mining techniques which is used for identifying the relation between one item to another, to find these rules we are using Apriori Algorithm.

We are creating a model which analyses the consumer purchase data of a grocery store and gives out simple statistics like How many items people order in a single purchase? Most ordered purchases etc. in form graphs and finding out the association rules.

Keywords: Data Mining, Market Basket Analysis, Apriori Algorithm, Association Rules.

TABLES OF CONTENTS

List of Figures

CHAPTER 1

INTRODUCTION

1. INTRODUCTION

1.1 ABOUT THE PROJECT

Market basket analysis is a data analysis technique used by retailers, particularly grocery stores, to identify products that are frequently purchased together. The goal of this analysis is to understand the purchasing patterns of customers, which can help the store optimize its product offerings, promotions, and layouts to drive sales and increase customer satisfaction.

The analysis is typically performed on transactional data, which records the items purchased by customers in each transaction. The data is then analyzed to find the most frequent combinations of items, referred to as "itemsets", that are purchased together. These itemsets can then be used to create "association rules", which express the likelihood of one item being purchased when another item is purchased.

For example, a common association rule in a grocery store might be "If a customer purchases bread, they are 80% likely to purchase butter." This information can then be used by the store to strategically place these items near each other in the store, or to create promotions that bundle the two items together to drive sales.

Overall, market basket analysis provides valuable insights into customer behavior and can be used to make data-driven decisions to improve the performance of a grocery store.

1.2 OBJECTIVES OF THE PROJECT

The main objective of the project is to make Instacart retailers to understand the current customer's behavior and to predict future customers' purchasing behavior. Leveraging customer transaction data can help in understanding customers' purchasing behavior, offering right bundles and promotions, assortment planning and inventory management to retain customers, improve sales and extend their relationship with customers.

The specific objectives of the project are as listed below

1. To understand the purchasing pattern of products that comprise the customers' basket.
2. To study about many products usually purchased by the customers.
3. To study the most likely products purchased by the customers along with a particular product category.

1.3 SCOPE OF THE PROJECT

The scope of Market Basket Analysis is wide and can be applied in several industries and domains such as retail, e-commerce, finance, and healthcare, among others. Some of the common applications of Market Basket Analysis include:

Retail Industry: It helps retailers to understand customer purchasing behavior and identify frequently purchased items together, which can be used to optimize product placement, promotions, and inventory management.

E-commerce: Market basket analysis can be used to identify the relationships between items purchased by customers online and suggest related products to increase sales.

Finance: It can be used to identify correlations between financial products such as stocks and bonds and identify investment opportunities.
Healthcare: Market basket analysis can be used to identify patterns in patient data, such as co-occurring diseases or frequently prescribed medications, to improve patient care.

These are just a few examples, and the scope of Market Basket Analysis can be extended to other industries and domains as well, depending on the available data and the specific business needs.

1.4 ADVANTAGES

The advantages of Market Basket Analysis in the retail industry are:

Understanding Customer Purchasing Behavior: Market basket analysis provides insights into the relationships between items purchased by customers, helping retailers to understand their customers' purchasing behavior.

Optimizing Product Placement: Retailers can use the insights generated from Market Basket Analysis to optimize product placement, for example, by placing frequently purchased items together in the store.

Improving Inventory Management: By identifying frequently purchased items, retailers can optimize their inventory management and reduce the risk of stockouts.

Personalizing Promotions: Market basket analysis can be used to target promotions and advertisements to specific customer segments based on their purchasing behavior, leading to a more personalized customer experience.

Increasing Sales: By optimizing product placement, improving inventory management, and personalizing promotions, Market Basket Analysis can help retailers increase sales and improve their overall performance.

Cost Reduction: Market basket analysis can help retailers to reduce costs by reducing waste and overstocking, and by optimizing promotions and advertising efforts.

Competitor Analysis: Market basket analysis can be used to understand the purchasing behavior of customers in a specific market or industry, allowing retailers to gain insights into their competitors' strategies.

Data-Driven Decisions: Market basket analysis allows retailers to make data-driven decisions, improving the accuracy and efficiency of their decision-making process.

These advantages highlight the importance of Market Basket Analysis as a valuable tool for retailers looking to better understand their customers and make informed business decisions, ultimately leading to increased customer satisfaction and profitability.

1.5 DISADVANTAGES

Market Basket Analysis is a powerful tool for the retail industry, but it has certain limitations that should be taken into consideration:

Data Quality: The accuracy of the results depends on the quality of the data used, and incorrect or missing data can lead to incorrect results and conclusions.

Association Rules: Association rules generated by Market Basket Analysis are based on correlations between items, but they do not prove causality. This means that just because items are frequently purchased together, it does not mean that one item causes the other to be purchased.

Sparse Data: In some cases, the data may be sparse, meaning that there are few transactions or items, leading to low support values and a limited number of association rules.

Multiple Rules: Market Basket Analysis can generate multiple association rules for the same items, making it difficult to identify the most relevant ones for a particular business problem.

Interpreting Results: The results of Market Basket Analysis can be complex and difficult to interpret, and additional analysis may be needed to make sense of the results and apply them to real-world problems.

These limitations should be taken into consideration when using Market Basket Analysis in the retail industry, and appropriate measures should be taken to mitigate their effects. Despite these limitations, Market Basket Analysis remains a valuable tool for organizations looking to improve their performance by understanding their customers and making informed business decisions.

1.6 APPLICATIONS

Market basket analysis is a technique used to discover associations or relationships between products or items that customers tend to purchase together. It has several applications, including:

1. Recommender Systems: Market basket analysis is used to build recommender systems that recommend products or services to customers based on their purchase history. These systems are used by e-commerce websites, online retailers, and content providers to personalize the user experience and increase customer engagement.

2. Inventory Management: Market basket analysis can help retailers optimize their inventory by identifying which products are frequently purchased together. By stocking these products together, retailers can improve their in-store layouts and reduce stock-outs, which can lead to lost sales.

3. Cross-Selling and Up-Selling: Market basket analysis is also used for cross-selling and up-selling purposes. By identifying which products are frequently purchased together, retailers can offer complementary or higher-end products to customers, increasing revenue and customer satisfaction.

Exploratory data analysis (EDA) is a technique used to analyse and summarize datasets to gain insights into the data. It has several applications, including:

1. Data Cleaning: EDA is used to identify missing or erroneous data points, outliers, and inconsistencies in the dataset, enabling data scientists to clean the data before analysing it.
2. Feature Engineering: EDA is used to identify features or variables that are most predictive of the target variable. These features are used to build machine learning models that can predict the target variable with high accuracy.
3. Data Visualization: EDA is used to create visualizations that help data

scientists and stakeholders understand the data better. Visualizations can help identify patterns, trends, and relationships in the data that might not be apparent in the raw data.

4. Hypothesis Testing: EDA is used to test hypotheses and validate assumptions about the data. Hypothesis testing can help identify correlations, causal relationships, and patterns in the data that can be used to make business decisions.

1.7 HARDWARE AND SOFTARE REQUIREMENTS

1.7.1 Hardware Requirements

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. What the system does and not how it should be implemented.

Processor	: 8th Generation Intel Core i5/i7
Hard Disk	: 50GB and above
RAM	: 8 GB and above

1.7.2 Software Requirements

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

Operating system	: Windows 11
IDE	: Google Colab / Microsoft VSCode
Language	: Python

Google Colab: Google Colab is a free online platform that allows users to write and run Python code collaboratively in a web-based environment. It is a cloud-based service provided by Google that offers a Jupyter notebook-like interface and allows users to run Python code on Google's servers, eliminating the need for users to install any software on their local machines. Google Colab provides a range of pre-installed packages, including popular data science libraries like TensorFlow, Keras, and PyTorch.

Google Colab allows users to share and collaborate on code in real-time, making it an ideal platform for working on group projects or for receiving feedback from peers. Users can also store their notebooks on Google Drive and access them from anywhere with an internet connection. Additionally, Google Colab provides free access to GPU and TPU resources, making it a powerful tool for training deep learning models and processing large datasets.

Microsoft Visual Studio Code:

Microsoft Visual Studio Code, commonly referred to as VS Code, is a free, open-source code editor that is widely used by developers across various platforms. It was first released in 2015 and has since gained popularity due to its fast performance, ease of use, and extensive support for programming languages, frameworks, and tools.

One of the most notable features of VS Code is its robust extension system, which allows developers to customize the editor with a wide range of plugins, themes, and snippets. The extensions cover a vast range of functionalities, including code formatting, debugging, Git integration, and language-specific tools, among others.

VS Code is available for Windows, Linux, and macOS, making it accessible to a wide range of users. It also offers built-in support for Git, making it easy for developers to collaborate on code and manage version control. Additionally, VS Code has a built-in terminal, making it easy to run commands, and it supports debugging for multiple programming languages, including JavaScript, Python, and C++.

Overall, Microsoft Visual Studio Code is a versatile and powerful code editor that provides developers with a fast, customizable, and efficient environment for their coding needs. It is well-suited for both individual developers and teams working on a variety of projects, and its widespread adoption is a testament to its popularity and usefulness.

Python: Python is a high-level, interpreted programming language that was first released in 1991 by Guido van Rossum. Since then, it has become one of the most popular programming languages worldwide, thanks to its simplicity, versatility, and extensive community support. Python is widely used for a broad range of applications, including web development, scientific computing, data analysis, artificial intelligence, and machine learning.

One of the most notable features of Python is its easy-to-learn syntax, which emphasizes readability and reduces the cost of program maintenance. Python code is typically shorter and more concise than other programming languages, making it easier to write, read, and debug. Python also has a vast collection of libraries and frameworks, including NumPy, Pandas, Django, Flask, TensorFlow, and PyTorch, among others, which simplifies the development of complex applications and speeds up the development process.

Python is also cross-platform, meaning that it can run on a wide range of operating systems, including Windows, macOS, Linux, and Unix. Python's cross-platform nature makes it an excellent choice for building applications that are meant to run on different devices, such as desktops, servers, and mobile devices.

Overall, Python is a versatile, powerful, and easy-to-learn programming language that has become an essential tool for developers, data scientists, and researchers worldwide. Its extensive community support, large collection of libraries and frameworks, and cross-platform nature make it an ideal choice for building a broad range of applications.

Pandas: Pandas is a popular Python library used for data manipulation and analysis. It was first released in 2008 by Wes McKinney and has since become one of the most widely used data analysis tools worldwide. Pandas provides a range of data structures and functions that allow users to manipulate and analyse tabular data, such as spreadsheets and SQL tables, with ease.

One of the most notable features of Pandas is its two main data structures, Series and DataFrame. A Series is a one-dimensional labelled array that can hold any data type, while a DataFrame is a two-dimensional table with rows and columns, similar to a spreadsheet. Pandas provides a wide range of functions for manipulating and analysing these data structures, including filtering, sorting, grouping, and merging.

Pandas also provides excellent support for data cleaning and preparation, making it an essential tool for data scientists and analysts. It offers functions for handling missing data, converting data types, and reshaping data, among others. Additionally, Pandas integrates well with other Python libraries, such as NumPy, Matplotlib, and Scikit-learn, allowing users to perform advanced data analysis and visualization tasks.

NumPy: NumPy, short for "Numerical Python," is a popular Python library used for scientific computing and data analysis. It was first released in 1995 and has since become one of the most widely used numerical computing libraries worldwide. NumPy provides a powerful N-dimensional array object, as well as functions for performing mathematical operations on these arrays.

One of the most notable features of NumPy is its ndarray object, which is a multidimensional array used for storing and manipulating large volumes of data. NumPy's ndarray is more efficient than Python's built-in lists, allowing users to perform complex mathematical operations quickly and easily. NumPy provides a range of functions for manipulating and analysing these arrays, including indexing, slicing, reshaping, and broadcasting.

NumPy also provides a range of mathematical functions, including trigonometric functions, exponential functions, and statistical functions, among others. These functions are optimized for use with NumPy's arrays and can perform complex calculations quickly and efficiently.

NumPy integrates well with other Python libraries, such as Pandas, Matplotlib, and Scikit-learn, making it an essential tool for data analysis, scientific computing, and machine learning applications. NumPy's ease of use, speed, and extensive functionality make it an essential tool for data scientists, engineers, and researchers worldwide.

Nltk: Natural Language Toolkit, or NLTK, is a popular Python library used for natural language processing and text analysis. It was first released in 2001 and has since become one of the most widely used natural language processing libraries worldwide. NLTK provides a range of functions for processing and analysing natural language data, including text tokenization, stemming, and part-of-speech tagging.

One of the most notable features of NLTK is its extensive collection of language data sets and corpora, which includes text samples from a wide range of sources, including books, news articles, and social media. NLTK also provides a range of tools for language modelling, machine translation, and sentiment analysis, among others.

NLTK integrates well with other Python libraries, such as Scikit-learn, Pandas, and Matplotlib, making it an essential tool for data analysis, machine learning, and text mining applications. NLTK's ease of use, extensive functionality, and large community support make it an ideal choice for researchers, developers, and data scientists worldwide who work with natural language data.

Matplotlib: Matplotlib is a data visualization library for Python. It provides a variety of plotting functions for creating line plots, scatter plots, bar charts, histograms, and other types of visualizations. Matplotlib is highly customizable and provides fine-grained control over every aspect of a plot, making it a powerful tool for creating publication-quality graphics.

Networkx: NetworkX is a popular Python library used for the creation, manipulation, and analysis of complex networks, such as social networks, transportation networks, and biological networks. It was first released in 2002 and has since become one of the most widely used network analysis libraries worldwide. NetworkX provides a range of functions for generating and analysing network graphs, including methods for computing network properties, such as centrality, clustering, and shortest paths.

One of the most notable features of NetworkX is its flexibility and ease of use, which makes it an ideal choice for users with different levels of expertise. NetworkX provides a range of graph classes, including directed graphs, undirected graphs, and multigraphs, as well as a range of algorithms for generating and analysing these graphs. NetworkX also provides a range of visualization tools, allowing users to create custom visualizations of their networks.

NetworkX integrates well with other Python libraries, such as Matplotlib, Pandas, and NumPy, making it an essential tool for data analysis, machine learning, and network science applications. NetworkX's extensive functionality, flexibility, and large community support make it an ideal choice for researchers, developers, and data scientists worldwide who work with complex network data.

Seaborn: Seaborn provides a range of statistical tools, such as regression analysis and hypothesis testing, allowing users to perform complex statistical analyses and visualize the results. Seaborn also provides a range of functions for working with time series data, which makes it an ideal choice for users working with financial or economic data.

Overall, Seaborn is a versatile and powerful data visualization library that simplifies the process of creating informative and visually appealing plots in Python. Its ease of use, extensive functionality, and integration with other Python libraries make it an essential tool for data analysts, data scientists, and researchers worldwide.

Functools: The functools module also provides a range of other functions, including caching decorators, tools for handling keyword arguments, and utilities for creating decorators. These functions can be used to simplify and optimize code, particularly in situations where a function is called multiple times with the same arguments.

CHAPTER 2

LITERATURE SURVEY

2. LITERATURE SURVEY

RESEARCH PAPER – 1:

Title: Fast Algorithm for Mining Association Rules

Authors: Agrawal R. and Srikant R.

Agrawal R. and Srikant R.'s paper titled "Fast Algorithm for Mining Associations in Large Databases" proposes an efficient algorithm for discovering frequent itemsets in large datasets. The proposed algorithm, called Apriori, works by first identifying all frequent items (items that appear in a minimum percentage of transactions) and then generating larger itemsets by combining these frequent items.

The Apriori algorithm prunes the search space by using the "Apriori property", which states that if an itemset is frequent, then all of its subsets must also be frequent. This property allows the algorithm to avoid exploring large portions of the search space that are guaranteed to be unproductive.

The authors also introduce several optimizations to improve the efficiency of the algorithm, such as reducing the size of the candidate itemsets and using a hash-based structure to store the support counts.

Overall, the Apriori algorithm is a highly efficient and widely used method for discovering frequent itemsets in large datasets, making it a valuable tool for data mining applications.

RESEARCH PAPER – 2:

Title: Data Mining Techniques for Marketing, Sales and Customer Relationship Management.

Authors: Berry and Linoff

In their paper titled "Data Mining Techniques for Marketing, Sales and Customer Relationship Management," Berry and Linoff discuss the various ways in which data mining can be used to improve marketing, sales, and customer relationship management. The authors explain how data mining can be used to identify customer segments, predict customer behaviour, and personalize marketing campaigns. They also highlight the importance of data quality and data pre-processing in ensuring the accuracy and reliability of the results.

Additionally, the authors provide several case studies demonstrating the practical applications of data mining in various industries. Overall, the paper emphasizes the potential of data mining techniques in enhancing marketing, sales, and customer relationship management strategies.

RESEARCH PAPER – 3:

Title: Market Basket Analysis Using Apriori Algorithm In R Language

Authors: Nidhi Makarand Kawale, Dr. Snehil Dahima

In their paper titled "Market Basket Analysis Using Apriori Algorithm In R Language," Nidhi Makarand Kawale and Dr. Snehil Dahima describe the use of the Apriori algorithm for performing market basket analysis, a technique that involves identifying co-occurring items in a transactional database.

The authors explain the steps involved in implementing the Apriori algorithm using R language, including data pre-processing, generating frequent itemsets, and generating association rules. They also provide examples of how market basket analysis can be used for improving product recommendations, optimizing store layout, and increasing cross-selling opportunities.

The paper concludes by highlighting the potential of the Apriori algorithm for gaining insights into consumer behavior and improving business operations.

2.1 Existing System

Market Basket Analysis: Agrawal, Imielińksi and Swami (1993) (Agrawal, n.d.) apparently first used Market Basket Analysis who had a large collection of consumer transaction data previously collected and the association rules between items purchased were discovered.

The method was rapidly implemented as a standard method for a number of practical applications in the field of marketing (Chen et al., 2005). Kanagawa, Matsumoto, Koike and Imamura posted surveys completing open-choice checklists with respect to related allergenic foods (Kanagawa et al., 2009).

As a result, researchers found that certain food allergens appear to happen in the same person together. (2017) Russell et al. (1999) (Singh and Sinwar, 2017) indicated that it was conceivable to use Market Basket Analysis by marketing researchers in developing multi-category decision-making, theoretical model purchasing decisions involving products. Researchers Y.-L. Chen, Tang, Shen, & Hu, in 2005 have proposed to establish inductive hypotheses from theoretical point of view (Chen et al., 2005).

The findings indicate the possibility that customers may have mental templates among several other collections of objects, include connections that are complementary in terms of its activities and interests (for example, running shoes and water bottles).

At a logical point of view, the findings of this study can be used to make decisions such as displaying the two products close to each other, enhancing the probability that consumers can find and buy easily two goods, rather than just one.

<u>Association Rules:</u>

Apriori algorithm was proposed by Ramakrishnan Srikant and Rakesh Agrawal (Agrawal, n.d.), which is one of the most traditional algorithms to find frequent patterns of the Boolean rules. In large relational tables the authors elaborate on the concept of quantitative rules in mining.

Shruthi Gurudath, Julander(2007), analysed the percentage of customers buying a certain product and the percentage of overall revenue produced by this same product. By creating such associations one can easily find out about the leading products and what their sales share is. It is extremely important to measure which products are the leading products, as a great number of customers come into contact each day with these specific products.

Given the large traffic created by departments with leading products, it is essential to use this data to position other similar products in the vicinity. Thus, the process of generating association rules for the products has been analysed in the results. In the research, Raeder and Chawla (2011) followed a different approach to data on mining processes (Aulakh, 2015).

They have found exceptional communities (clusters) in data by modelling data as a product network. In the network method, they have shown that inferences between products can be extracted specifically and that the need for a set of aggregate rules is reduced. First, they analysed the characteristics of the successful networks and showed in the results above the defining groups in those networks would expose significant links between rules of combination. Berry and Linoff(2004) aimed to discover patterns by extracting associations or co-occurrences from the transactional information provided by a store(Berry and Linoff, 2004).

Customers who buy bread also often buy several bread related products such as milk, butter, or jam. The results are shown on how it makes sense to place these area units of groups side by side in a retail centre so that customers can quickly access them. Additionally, such related product groups should be placed side-by - side to remind customers of related products and very logically guide them through the centre. Ibrahim Cil (2012) (Anon, n.d.) has provided a new plan for supermarket-placement problems with association rules and multidimensional scaling evaluation.

He took customer receipt and product barcode details as data in his research for the Migros Türk supermarket, which holds an important position in the retail sector, and examined it using the Apriori algorithm according to the association rules process. Researcher then suggested a new development design for the store based on the resulting rules. Erpolat(2012) provided information on the Apriori and FP-Growth algorithms which have been widely used in the research of association rules, implemented these algorithms to the consumer shopping data of an approved automotive service and compared the results(Mostafa, 2015).

As a result of the analysis, researcher noted that the FP-Growth algorithm is more suitable than the Apriori algorithm to analyse a data set because of the computation speeds. Shruthi Gurudath 2990078 Kaur and Kang (2016), on dynamic data, suggested an algorithm which would conduct association rule mining (Singh and

Sinwar,2017). They did periodic mining by working with the principle of change modelling. The results are noted on methods to find the outliers and about predicting the future association rules.

The words shopping basket were proposed by Cachon and Kok (2007). The shopping basket establishes the set of products the consumer would like to purchase on a single journey to a shop (Anon, n.d.). According to researchers, the customer would not be treated on subsequent shopping trips if he cannot bring out his shopping cart in a specific store. So, even though the merges for profit are smaller than other products, some products must be offered to the consumer by the retailer. Thus, researchers created a method and approach that tests the productivity of the shopping basket, contributing to the retailer's higher projected return than that of category management.

2.2 Proposed System

2.2.1 Description of Proposed system

The Project focuses on Instacart Dataset and the aim is to build a system that explores the data and visualises the important information extracted from the given data to gain a better understanding of dataset and to give an overview on this data to the user, this stage is called Exploratory Data Analysis.

Second part of the project focuses on Data Mining and involves using algorithms such as Apriori to generate frequent itemsets and association rules from transactional data. The resulting rules can be used to inform business decisions such as product placement, cross-selling, and promotions.

The Various steps involved as follows:

1. Dataset Collection: The grocery store data is taken from Kaggle, consisting about 3 million transactions which are used to perform Data Analysis and Data Mining. Collect the transaction data from the market that you want to analyse. This data should contain the items purchased in each transaction and the transaction ID.

2. Load Data**:** The first step is to load the data into the Python programming environment. This can be done by reading the data from a CSV file or any other database.
3. Data Preparation and Cleaning: Clean and pre-process the data to make it ready for analysis. This includes removing missing values, converting the data into a suitable format, and aggregating the data based on the items purchased in each transaction.
4. Load Data: The first step is to load the data into the Python programming environment. This can be done by reading the data from a CSV file or any other database.

Exploratory Data Analysis:

1. **Data Visualization**: Visualizing the data using various plots and graphs to gain insights into the relationships between variables, patterns in the data, and outliers.

2. **Descriptive statistics:** Calculating basic descriptive statistics such as mean, median, and standard deviation to summarize the data.

3. **Data distribution:** Plotting histograms and kernel density plots to visualize the distribution of the data and to identify any skewness or outliers.

4. **Presentation and Communication**: Communicating the results of the exploratory data analysis by presenting the findings in a clear and concise manner.

Market Basket Analysis:

Use the apriori algorithm to generate association rules that show the relationship between items in the market basket. The apriori algorithm uses the transaction data to find the items that are frequently purchased together.

1. **Rule Generation**: Generate association rules based on the results of the market basket analysis. These rules show the relationship between items and the support, confidence, and lift metrics associated with each rule.

2. **Rule Evaluation**: Evaluate the association rules generated to determine their significance and reliability. This can be done by using metrics such as lift, confidence, and support, and also by visually representing the results using graphs and charts.

3. **Rule Interpretation**: Interpret the results of the analysis to gain insights into the market basket behaviour of customers. This information can be used to inform marketing strategies, product placement, and inventory management.

2.2.2 Target Users

The target users of exploratory data analysis (EDA) are typically data scientists, analysts, researchers, and decision-makers in various fields such as finance, healthcare, marketing, social sciences, and more. These individuals use EDA techniques to explore and analyse large datasets, identify patterns and relationships, and gain insights that can inform business decisions. EDA is also useful for data cleaning and preparation, as it helps to identify missing data, outliers, and inconsistencies. Additionally, EDA can be used by data journalists and other individuals interested in exploring data for insights and storytelling purposes.

The target users of market basket analysis (MBA) are typically retailers, marketers, and business analysts who want to understand the relationships between products and customer purchasing behaviour. MBA can also be used in various other industries such as healthcare, finance, and telecommunications. MBA techniques can help retailers to optimize store layouts, create targeted marketing campaigns, and improve product recommendations. Business analysts can use MBA to identify cross-selling and up-selling opportunities, analyse customer behaviour, and identify trends and patterns in sales data. MBA can also be used by data scientists and researchers to explore associations between variables and make predictions.

CHAPTER 3

PROPOSED ARCHITECTURE

3. PROPOSED ARCHITECTURE

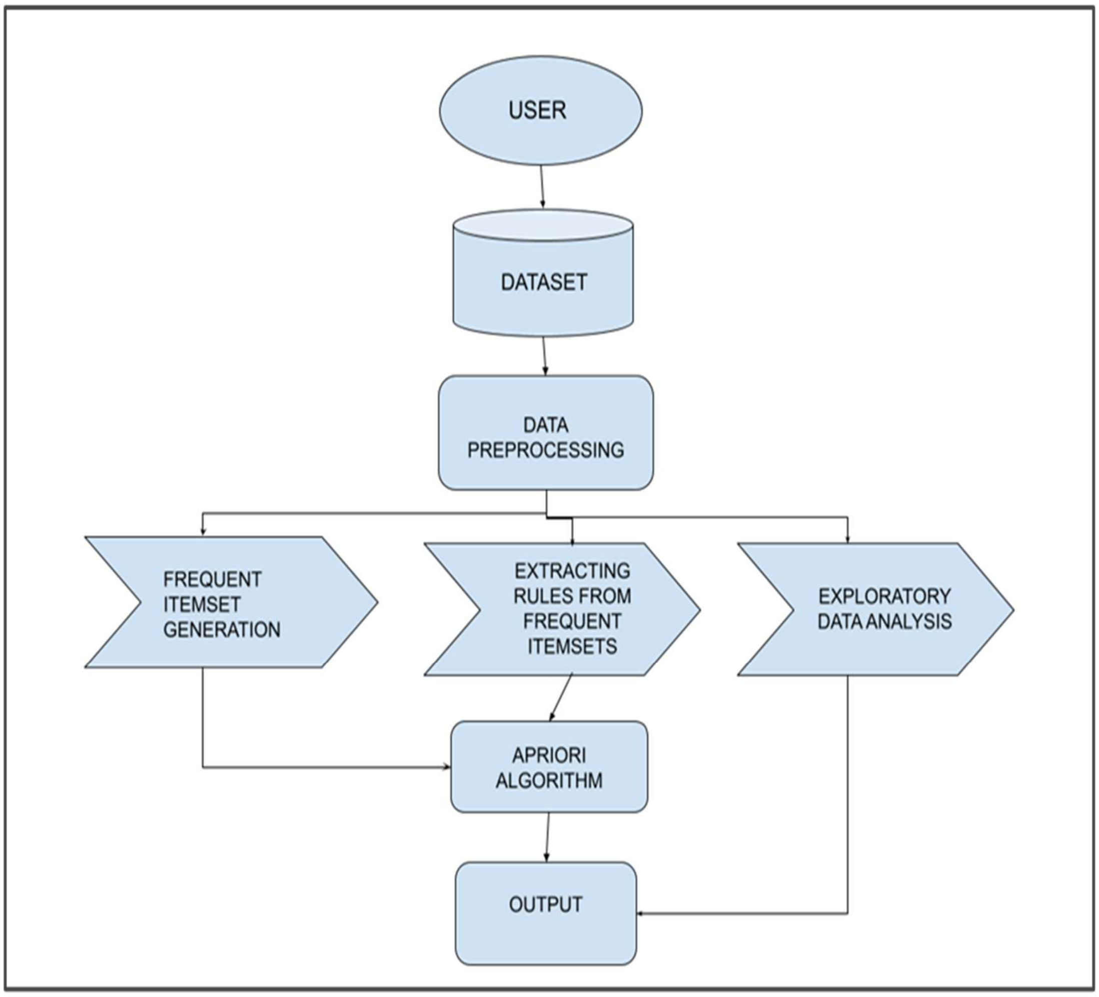

USER
DATASET
DATA PREPROCESSING
FREQUENT ITEMSET GENERATION
EXTRACTING RULES FROM FREQUENT ITEMSETS
EXPLORATORY DATA ANALYSIS
APRIORI ALGORITHM
OUTPUT

Fig:3.1 Proposed System Architecture

1. Data Preparation:
    - The dataset consists of CSV files which each contain details such as purchase history, orders, frequent orders etc.
    - This Data is then loaded to perform further steps.
2. Data Pre-processing:
    - Formatting files while reading - converting csv to dataframes
    - Merging multiple files - using joins
    - Deleting redundant data frames - to save memory
    - Data analysis with the processed data

3. Frequent Itemset Generation:
    - The first step is to calculate the support of each item in the transactional dataset.
    - The second step is to generate candidate itemsets, which are combinations of items that meet a minimum support threshold.
    - The third step is to prune the candidate itemsets to generate frequent itemsets. This step involves removing itemsets that do not meet the minimum support threshold.
4. Extracting Rules from Frequent Itemsets:
    - The first step is to generate association rules from the frequent itemsets.
    - The second step is to filter the association rules based on criteria such as support, confidence, and lift.
    - The third step is to interpret the association rules and take action based on the insights gained.
5. Exploratory Data Analysis:
    - Prepare data to only focus on important data
    - Generate graphs and plots to visualize this important or necessary data.
6. Output:
    - The output is displayed in terms of various graphs and plots for both Market Basket Analysis and Exploratory Data Analysis.

CHAPTER 4

IMPLEMENTATION

4. IMPLEMENTATION

4.1 ALGORITHM

4.1.1 APRIORI ALGORITHM

The Apriori Algorithm is a basic algorithm proposed by Agrawal & Srikant in 1994 for the determination of the frequent itemset for Boolean association rules. The principles of Apriori state that "if an itemset is frequent, then all its subset items will be frequent". If the support for the itemset is more than the support level, the itemset is "frequent".

The algorithm is based on the prediction of items, which move from the previous stage on a regular basis. The name derived from the term "prior". Apriori algorithm includes the type of association rules in data mining. The rule that states associations between multiple attributes is often called affinity analysis or market basket analysis.

For understanding Apriori principle, let us consider an instance, if the item set {b, d, e} from the dataset which is a frequent itemset, i.e., its support measure (0.35) is greater than the minimum support measures (0.25), then all of its subsets such as b, d, e, {b, d}, {b, e}, {d, e} will also be frequent item sets.

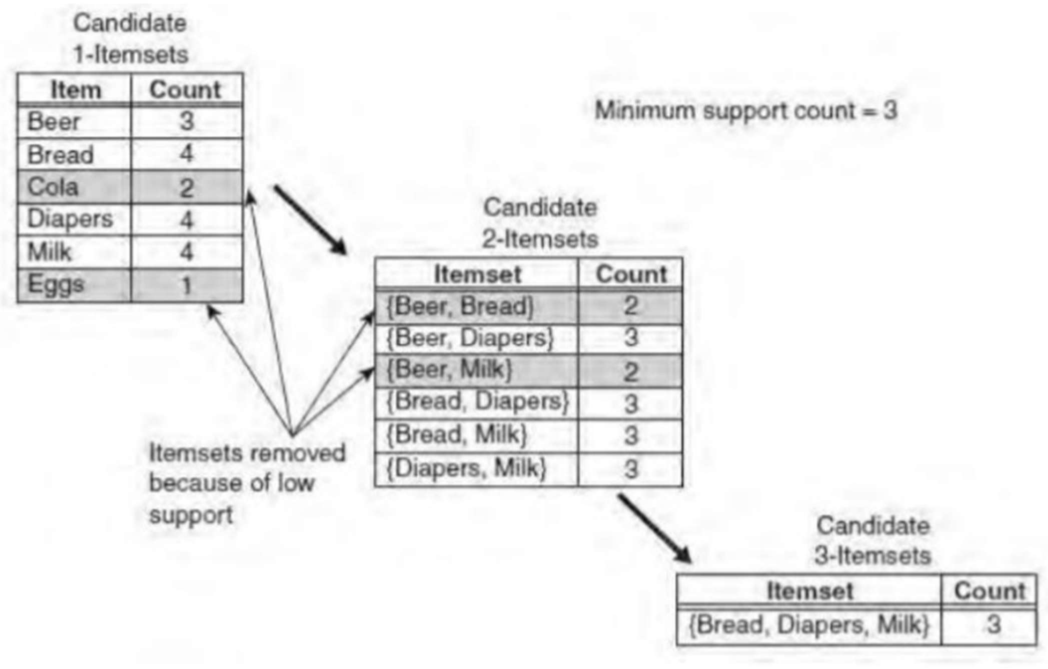
As a result, therefore all subtypes b, d, e has to be regular if {b, d, e} is frequent. On the contrary, if any item sets like {a, b} are uncommon then all the supersets must be even rare. The entire segment containing the {a, b} supersets can be trimmed right away. The pruning method of the linear direction relying on the support measure is called as Support-based pruning. The sort of pruning process is achieved by a major objective of the support measure. This characteristic is also known as the antimonotone property of the support measure.

The Apriori algorithm works in two steps:

Prune and Join:

1. Generate all frequent item sets – A frequent item set is an item set that has transaction support above minimum support.

2. Generate all confident association rules from frequent item sets – A confident association rule is a rule with confidence above minimum confidence.


Candidate 1-Itemsets
Minimum support count = 3
Candidate 2-Itemsets
Itemsets removed because of low support
Candidate 3-Itemsets

Item	Count
Beer	3
Bread	4
Cola	2
Diapers	4
Milk	4
Eggs	1

Itemset	Count
{Beer, Bread}	2
{Beer, Diapers}	3
{Beer, Milk}	2
{Bread, Diapers}	3
{Bread, Milk}	3
{Diapers, Milk}	3

Itemset	Count
{Bread, Diapers, Milk}	3

Fig: 4 1 Apriori Algorithm Example

4.1.2 MARKET BASKET ANALYSIS

Market basket analysis, also known as association rule learning, is a technique used in data mining and machine learning to identify relationships between items in large datasets. It is used to analyse shopping patterns and discover which items are frequently purchased together.

The process of market basket analysis starts with creating a transaction database, which is a record of all the items that have been purchased in a specific period of time. This database is then analysed to identify items that are frequently purchased together, also known as associations.

The associations are then quantified using metrics such as support, confidence, and lift. Support measures the frequency of an item or item set in the transaction database, confidence measures the likelihood of an item set being purchased given that a certain item is purchased, and lift measures the strength of the association between two items.

Market basket analysis can be used to support marketing and sales initiatives by providing insights into customer behaviour and preferences. For example, a retailer can use market basket analysis to identify the most popular items, cross-sell opportunities, and promotions that drive sales.

Overall, market basket analysis provides valuable insights into customer behaviour and helps businesses make informed decisions about their marketing and sales strategies.

4.1.3 EXPLORATORY DATA ANALYSIS

EDA stands for Exploratory Data Analysis, which is a technique used for analyzing and understanding data. It involves visualizing and summarizing data to identify patterns, trends, and relationships that exist within the data set.

The goal of EDA is to gain insights into the data and to identify potential issues and outliers, which can then be addressed before proceeding with more advanced data analysis techniques. EDA can be performed using various tools and techniques, such as histograms, scatter plots, box plots, and regression analysis, to visualize and explore the data and to understand the underlying structure and relationships within the data. In this project we will display the following graphs:

How many items do people order in a single purchase?
What are Instacart's most ordered items?
What products do people order the most: organic vs non organic products?
What hour of day and day of week most products are ordered?
What items are first added to the cart by Instacart's users?
What are Instacart's most reordered products?
When do Instacart's users place the next order?

4.2 CODE IMPLEMENTATION

4.2.1 EXPLORATORY DATA ANALYSIS

```
## Load Libraries
import pandas as pd # dataframes
import numpy as np # algebra & calculus
import nltk # text preprocessing & manipulation

import matplotlib.pyplot as plt # plotting
import seaborn as sns # plotting

from functools import partial # to reduce df memory consumption by
applying to_numeric

color = sns.color_palette() # adjusting plotting style
import warnings
warnings.filterwarnings('ignore') # silence annoying warnings

%matplotlib inline
# aisles
aisles = pd.read_csv('aisles.csv')
print('Total aisles: {}'.format(aisles.shape[0]))
aisles.head()
# departments
departments = pd.read_csv('departments.csv')
print('Total departments: {}'.format(departments.shape[0]))
departments.head()
# products
products = pd.read_csv('products.csv')
print('Total products: {}'.format(products.shape[0]))
products.head(5)
### Combine it all together into 1 DataFrame "GOODS"
# combine aisles, departments and products (left joined to products)
goods = pd.merge(left=pd.merge(left=products, right=departments,
how='left'), right=aisles, how='left')
# to retain '-' and make product names more "standard"
goods.product_name = goods.product_name.str.replace(' ', '_').str.lower()

goods.head()
# basic group info (departments)
plt.figure(figsize=(12, 5))
goods.groupby(['department']).count()['product_id'].copy()\
.sort_values(ascending=False).plot(kind='bar',
                          #figsize=(12, 5),
                          title='Departments: Product #')
```

```
# basic group info (top-x aisles)
top_aisles_cnt = 15
plt.figure(figsize=(12, 5))
goods.groupby(['aisle']).count()['product_id']\
.sort_values(ascending=False)[:top_aisles_cnt].plot(kind='bar',
                           #figsize=(12, 5),
                           title='Aisles: Product #')

# plot departments volume, split by aisles
f, axarr = plt.subplots(6, 4, figsize=(12, 30))
for i,e in enumerate(departments.department.sort_values(ascending=True)):
   axarr[i//4, i%4].set_title('Dep: {}'.format(e))
   goods[goods.department==e].groupby(['aisle']).count()['product_id']\
   .sort_values(ascending=False).plot(kind='bar', ax=axarr[i//4, i%4])
f.subplots_adjust(hspace=2)
## Main Datasets (orders + order details)
# load datasets

# train dataset
op_train = pd.read_csv('order_products__train.csv',
                 dtype={'order_id': np.int32, 'product_id': np.int32,
                      'add_to_cart_order': np.int16, 'reordered': np.int8})
print('Total ordered products(train): {}'.format(op_train.shape[0]))
op_train.head(10)
# test dataset (submission)
test = pd.read_csv('sample_submission.csv')
print('Total orders(test): {}'.format(op_train.shape[0]))
test.head()
# prior dataset
op_prior = pd.read_csv('order_products__prior.csv', engine='c',
                 dtype={'order_id': np.int32, 'product_id': np.int32,
                      'add_to_cart_order': np.int16, 'reordered': np.int8})
print('Total ordered products(prior): {}'.format(op_prior.shape[0]))
op_prior.head()
# orders
orders = pd.read_csv('orders.csv', dtype={'order_id': np.int32,
                                  'user_id': np.int32,
                                  'order_number': np.int32,
                                  'order_dow': np.int8,
                                  'order_hour_of_day': np.int8,
                                  'days_since_prior_order': np.float16})
print('Total orders: {}'.format(orders.shape[0]))
orders.head()
```

```
### Combine (orders, order details, product hierarchy) into 1 dataframe
order_details

from functools import partial

# merge train and prior together iteratively, to fit into 8GB kernel RAM
# split df indexes into parts
indexes = np.linspace(0, len(op_prior), num=10, dtype=np.int32)

# initialize it with train dataset
order_details = pd.merge(
            left=op_train,
             right=orders,
             how='left',
             on='order_id'
      ).apply(partial(pd.to_numeric, errors='ignore', downcast='integer'))

# add order hierarchy
order_details = pd.merge(
            left=order_details,
            right=goods[['product_id',
                     'aisle_id',
                     'department_id']].apply(partial(pd.to_numeric,
                                         errors='ignore',
                                         downcast='integer')),
            how='left',
            on='product_id'
)

print(order_details.shape, op_train.shape)

# delete (redundant now) dataframes
del op_train

order_details.head()
%%time
# update by small portions
for i in range(len(indexes)-1):
   order_details = pd.concat(
      [
         order_details,
         pd.merge(left=pd.merge(
                  left=op_prior.iloc[indexes[i]:indexes[i+1], :],
                  right=goods[[
                     'product_id',
                      'aisle_id',
                      'department_id'
                  ]].apply(partial(pd.to_numeric,
                              errors='ignore',
                              downcast='integer')),
                  how='left',
                  on='product_id'
                  ),
              right=orders,
              how='left',
              on='order_id'
            ) #.apply(partial(pd.to_numeric, errors='ignore',
downcast='integer'))
      ]
   )

print('Datafame length: {}'.format(order_details.shape[0]))
print('Memory consumption: {:.2f}
Mb'.format(sum(order_details.memory_usage(index=True,
                                          deep=True) / 2**20)))
# check dtypes to see if we use memory effectively
print(order_details.dtypes)

# make sure we didn't forget to retain test dataset :D
test_orders = orders[orders.eval_set == 'test']

# delete (redundant now) dataframes
del op_prior, orders
%%time
# unique orders, product_ordered, users
print('Unique users: {}'.format(len(set(order_details.user_id))))
print('Unique orders: {}'.format(len(set(order_details.order_id))))
print('Unique products bought:
{}/{}'.format(len(set(order_details.product_id)), len(goods)))
# unordered products
unordered = goods[goods.product_id.isin(list(set(goods.product_id) -
set(order_details.product_id)))]
print('"Lonesome" products cnt: {}/{}'.format(unordered.shape[0],
len(goods)))
unordered.head()
### Popular Products (Bestsellers)
%%time
# popular products (total set, not only train)
top = 15
top_products = pd.merge(
   # to see train:
    # left=pd.DataFrame(order_details[order_details.eval_set ==
'train'].groupby(['product_id'])['order_id']\
    left=pd.DataFrame(order_details.groupby(['product_id'])['order_id']\
    .apply(lambda x:
len(x.unique())).sort_values(ascending=False)[:top].reset_index('product_id')
),
    right=goods,
    how='left')

f, ax = plt.subplots(figsize=(12, 10))
plt.xticks(rotation='vertical')
sns.barplot(x=top_products.product_name, y=top_products.order_id)
plt.ylabel('Number of Orders, Containing This Product')
plt.xlabel('Product Name')
### Most "Frequently" Bought Products
%%time
# most "frequently" bought products (total set, not only train)
# most "frequently" ~ time between orders (within selected customer's
orders),
# that contain that product, is the least
#(products, which were bought by more than 100 customers, to omit outliers)
top = 15
customer_limit = 100

temp = order_details.groupby(['product_id'])[['days_since_prior_order',
'user_id']]\
.aggregate({'days_since_prior_order': np.mean, 'user_id': len}).reset_index()

frequent_products = pd.merge(
    left=pd.DataFrame(temp[temp.user_id >
customer_limit].sort_values(['days_since_prior_order'],
                                                  ascending=True)[:top]),
    right=goods,
    how='left')

plt.figure(figsize=(12,6))
plt.xticks(rotation='vertical')
sns.barplot(x=frequent_products.product_name,y=frequent_products.days_si
nce_prior_order)
plt.ylabel('Average Days Between Orders, Containing This Product')
plt.xlabel('Product Name')

del temp
### Orders, Split by Product Count
%%time
ord_by_prods = order_details.groupby("order_id")["add_to_cart_order"]\
.aggregate(np.max).reset_index()['add_to_cart_order'].value_counts()

print('Most common order contains: {} products'.format(
   ord_by_prods[ord_by_prods.values ==
ord_by_prods.max()].index.values[0]))

# plot it
plt.figure(figsize=(12, 8))
plt.xticks(rotation='vertical')
sns.barplot(x= ord_by_prods.index, y= ord_by_prods.values)
plt.ylabel('Number of Orders')
plt.xlabel('Number of Products in Order')
plt.xlim([0, 50])
pass
### Products with the Highest Reorder Rate
%%time
# consider products, purchased in more than X orders
order_limit = 100
top = 15

mo_products = order_details.groupby('product_id')[['reordered', 'order_id']]\
.aggregate({'reordered': sum, 'order_id': len}).reset_index()
mo_products.columns = ['product_id', 'reordered', 'order_cnt']

mo_products['reorder_rate'] = mo_products['reordered'] /
mo_products['order_cnt']
mo_products = mo_products[mo_products.order_cnt >
order_limit].sort_values(['reorder_rate'],
                                                   ascending=False)[:top]

mo_products = pd.merge(
   left=mo_products,
   right=goods,
   on='product_id')
mo_products

# plot it
plt.figure(figsize=(12, 6))
plt.xticks(rotation='vertical')
sns.barplot(x=mo_products.product_name,y=
mo_products.reorder_rate*100)
plt.ylabel('Reorder Rate, %')
plt.xlabel('Product Name')
pass
### Orders, Split by Day of Week
plt.figure(figsize=(12,6))
```

```
order_details.groupby('order_dow')['order_id'].apply(lambda x:
len(x.unique())).plot(kind='bar')
plt.xticks(rotation='vertical')
plt.ylabel('Order Count')
plt.xlabel('Day of Week (coded)')
pass
### Orders, Split by Hour
plt.figure(figsize=(12,6))
order_details.groupby('order_hour_of_day')['order_id'].apply(lambda x:
                                                  len(x.unique())).plot(kind='bar')
plt.xticks(rotation='vertical')
plt.ylabel('Order Count')
plt.xlabel('Hour of a Day (0-23)')
pass
### Most Popular Departments
pop_dep = pd.merge(
    left=order_details.groupby('department_id')['order_id'].apply(lambda x:
                                                      len(x.unique())).reset_index(),
    right=goods[['department_id', 'department']].drop_duplicates(),
    how='inner',
    on='department_id'
).sort_values(['order_id'], ascending=False)

# plot it
total_orders = len(set(order_details.order_id))

plt.figure(figsize=(12, 6))
plt.xticks(rotation='vertical')
sns.barplot(x=pop_dep.department,y= pop_dep.order_id / total_orders * 100)
plt.ylabel('% of Orders, Containing Products from Department, #')
plt.xlabel('Department Name')
pass
### Most Popular Aisles
%%time
pop_ais = pd.merge(
    left=order_details.groupby('aisle_id')['order_id'].apply(lambda x:
len(x.unique())).reset_index(),
    right=goods[['aisle_id', 'aisle']].drop_duplicates(),
    how='inner',
    on='aisle_id'
).sort_values(['order_id'], ascending=False)[:top]

# plot it
total_orders = len(set(order_details.order_id))

plt.figure(figsize=(12, 6))
plt.xticks(rotation='vertical')
sns.barplot(x=pop_ais.aisle,y= pop_ais.order_id / total_orders * 100)
plt.ylabel('% of Orders, Containing Products from Aisle, #')
plt.xlabel('Aisle Name')
pass
### Distribution of Order Count per User
%%time
ocpu = order_details.groupby('user_id')['order_id']\
.apply(lambda x:
len(x.unique())).reset_index().groupby('order_id').aggregate("count")

print('Most common user made: {} purchases'.format(
   ocpu[ocpu.user_id == ocpu.user_id.max()].index.values[0]))

plt.figure(figsize=(12, 6))
sns.barplot(x=ocpu.index, y=ocpu.user_id)
plt.xticks(rotation='vertical')
plt.ylabel('User Count')
plt.xlabel('Number of Orders, made by a User')
pass
### Days to Next Order
%%time
dtno = order_details.dropna(axis=0,

subset=['days_since_prior_order']).groupby('order_id')['days_since_prior_ord
er']\
.aggregate("mean").reset_index().apply(np.int32).groupby('days_since_prior
_order').aggregate("count")

print('Most frequently next orders are made once in: {} days'.format(
   dtno[dtno.order_id == dtno.order_id.max()].index.values[0]))

print('We clearly see monthly (>=30) and weekly (7) peaks')

plt.figure(figsize=(12, 6))
sns.barplot(x=dtno.index,y= dtno.order_id)
plt.xticks(rotation='vertical')
plt.ylabel('Order Count')
plt.xlabel('Days Passed Since Last Order')
pass
### Time of last order vs. probability of reorder
%%time

por = order_details.dropna(axis=0, subset=['days_since_prior_order'])\
.groupby('days_since_prior_order')['reordered'].aggregate("mean").reset_inde
x()
```

```
        print('We can see that longer lags leads to lowered probability (new items),\
        \nwhile same day orders tends to have more overlapped product list')

        plt.figure(figsize=(12, 6))
        sns.barplot(x= por.days_since_prior_order, y=por.reordered*100)
        plt.xticks(rotation='vertical')
        plt.ylabel('Probability of Reorder, %')
        plt.xlabel('Days Passed Since Last Order')
        pass
        %%time

        # share of organic/non-organic products and correspondent orders count
        org = pd.merge(
            left=order_details[['product_id', 'order_id']],
            right=goods[['product_id', 'product_name']],
            how='left',
            on='product_id')

        org['organic'] = org.product_name.str.contains('organic').astype(np.int8)
        org = org.groupby('order_id')['organic'].aggregate("max").value_counts()

        # plot it
        plt.figure(figsize=(12, 6))
        sns.barplot(x=org.index,y=( org / org.sum() * 100) )
        plt.xticks(rotation='vertical')
        plt.xlabel('Order Contains Organics (Boolean)')
        plt.ylabel('% of Orders')
pass
```

4.2.2 MARKET BASKET ANALYSIS

```
#imports
import pandas as pd #Python data analysis library
import numpy as np #Python scientific computing
from mlxtend.frequent_patterns import apriori
from mlxtend.frequent_patterns import association_rules
import networkx as nx
import matplotlib.pyplot as plt

#import dataset
trainDf = pd.read_csv("order_products__train.csv")
orderDf = pd.read_csv("orders.csv")
productDf = pd.read_csv("products.csv")
```

```
priorDf = pd.read_csv("order_products__prior.csv")
trainDf = trainDf.append(priorDf,ignore_index = True)

trainDf['reordered'] = 1

productCountDf = trainDf.groupby("product_id",as_index =
False)["order_id"].count()


#Top 100 most frequently purchased products
topLev = 100

#Here order_id is the count so we need to sort the data frame w.r.t order_id
productCountDf = productCountDf.sort_values("order_id",ascending = False)

topProdFrame = productCountDf.iloc[0:topLev,:]
topProdFrame = topProdFrame.merge(productDf,on = "product_id")
productId= topProdFrame.loc[:,["product_id"]]

df = trainDf[0:0]
for i in range(0,99):
   pId = productId.iloc[i]['product_id']
   stDf = trainDf[trainDf.product_id == pId ]
   df = df.append(stDf,ignore_index = False)

df.head()

df.info()

basket = df.groupby(['order_id',
'product_id'])['reordered'].sum().unstack().reset_index().fillna(0).set_index('order_id
)

# Convert the units to 1 hot encoded value
def encode_units(x):
   if x <= 0:
      return 0
   if x >= 1:
      return 1

basket_sets = basket.applymap(encode_units)

basket_sets.head()

basket_sets.size
```

```
# Build up the frequent items
frequent_itemsets = apriori(basket_sets, min_support=0.006, use_colnames=True)

frequent_itemsets

# Create the rules
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
rules

rules[ (rules['lift'] >= 2) &
      (rules['confidence'] >= 0.1) ]

datanew = rules[ (rules['lift'] >= 2) &
      (rules['confidence'] >= 0.1) ]

G = nx.Graph()

G = nx.from_pandas_edgelist(datanew,'antecedents','consequents')

from matplotlib.pyplot import figure
figure(figsize=(10, 8))
nx.draw_shell(G, with_labels=True)
```

CHAPTER 5
RESULTS

5. RESULTS

5.1 EXPLORATORY DATA ANALYSIS

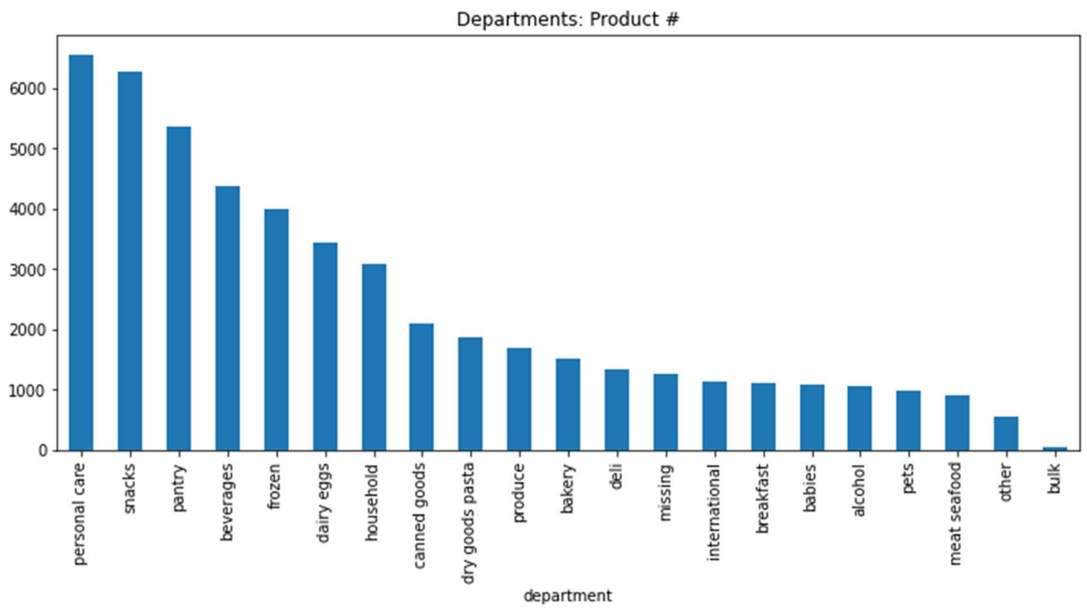


Fig: 5.1.1 Department Volume

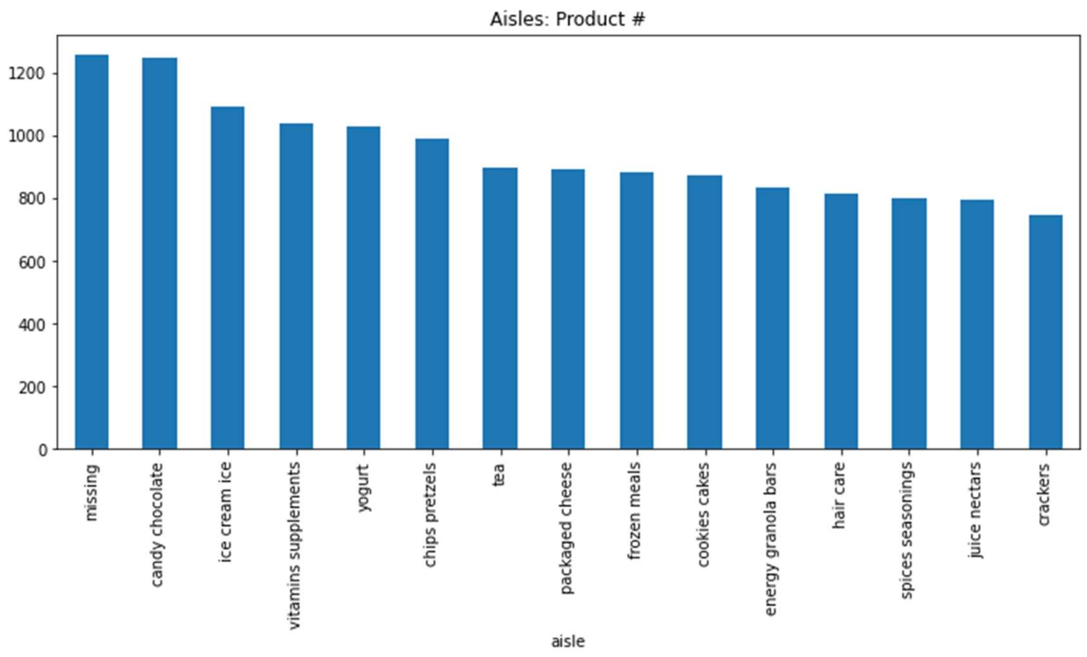


Fig: 5.1.2 Aisles Volume

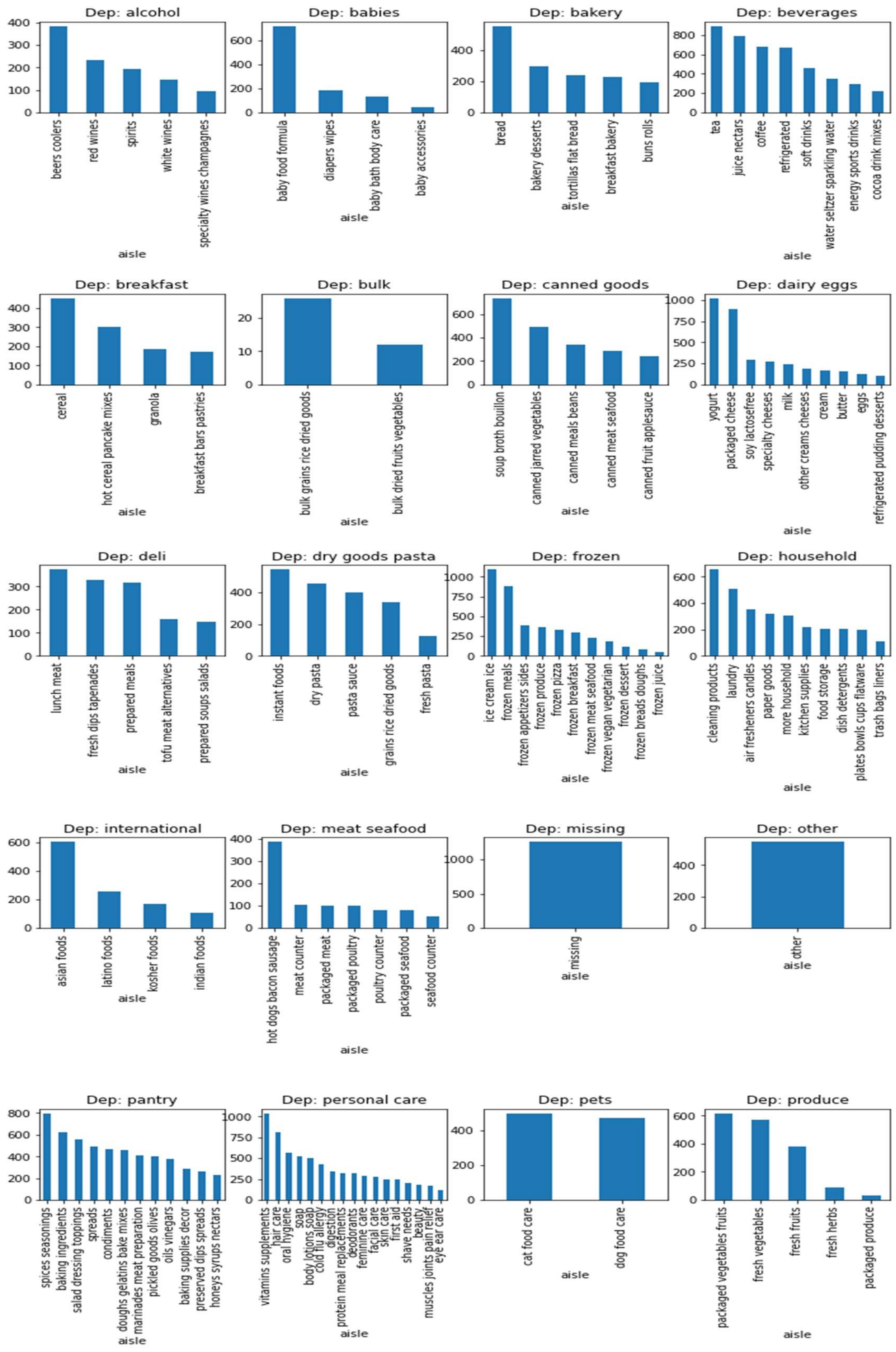


Fig: 5.1.3 Department Information

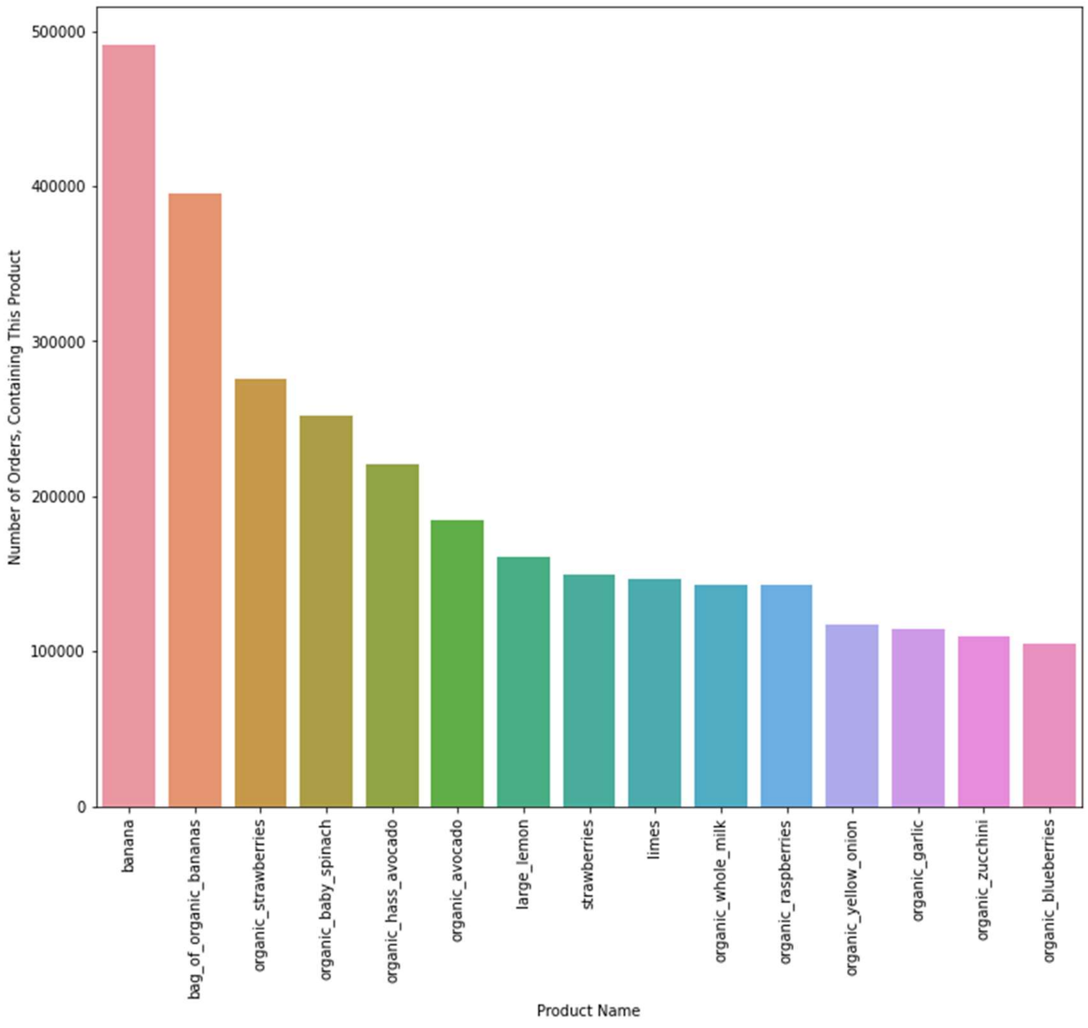


Fig: 5.1.4 Popular Products

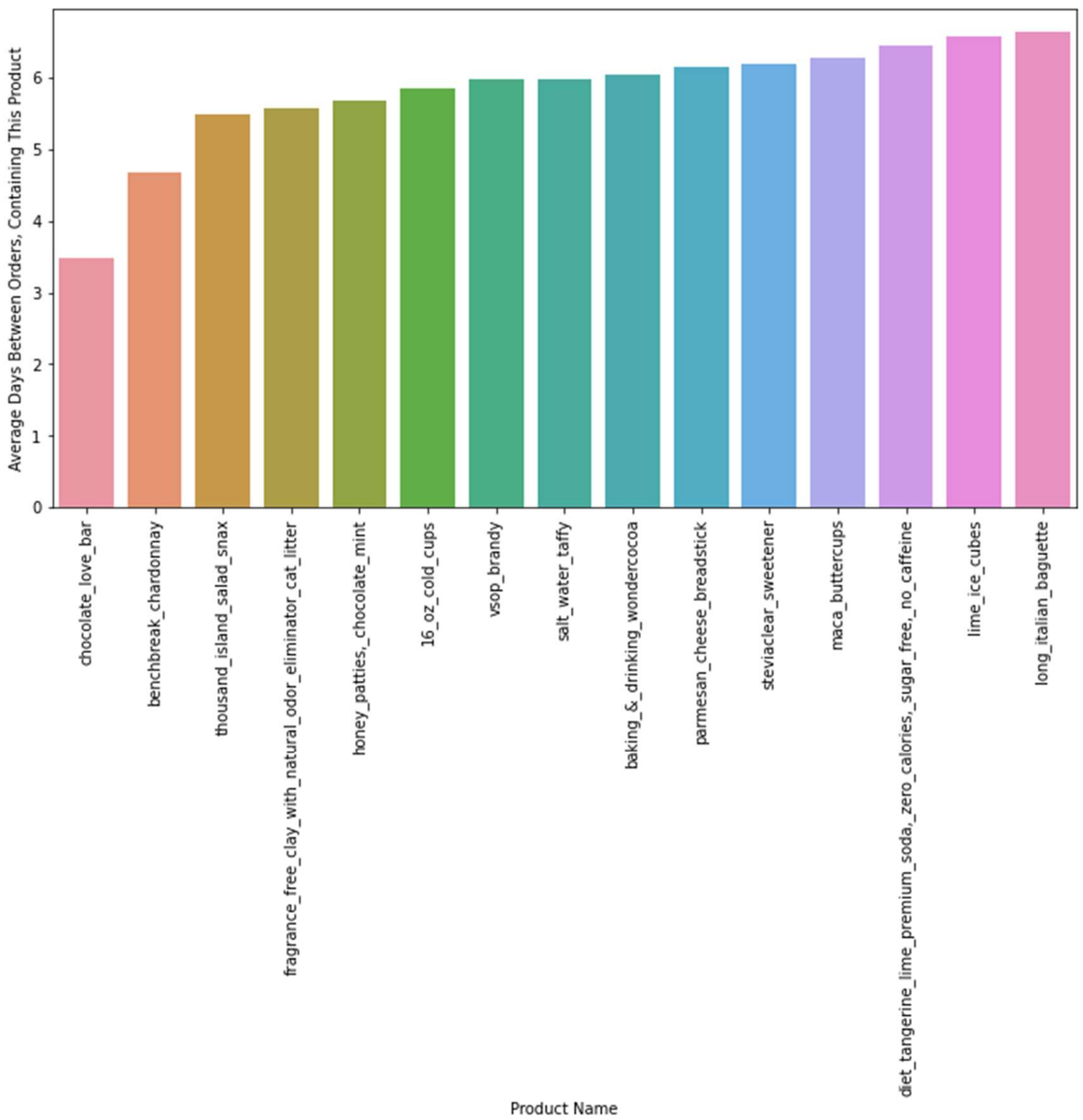


Fig: 5.1.5 Most Frequently Bought Products

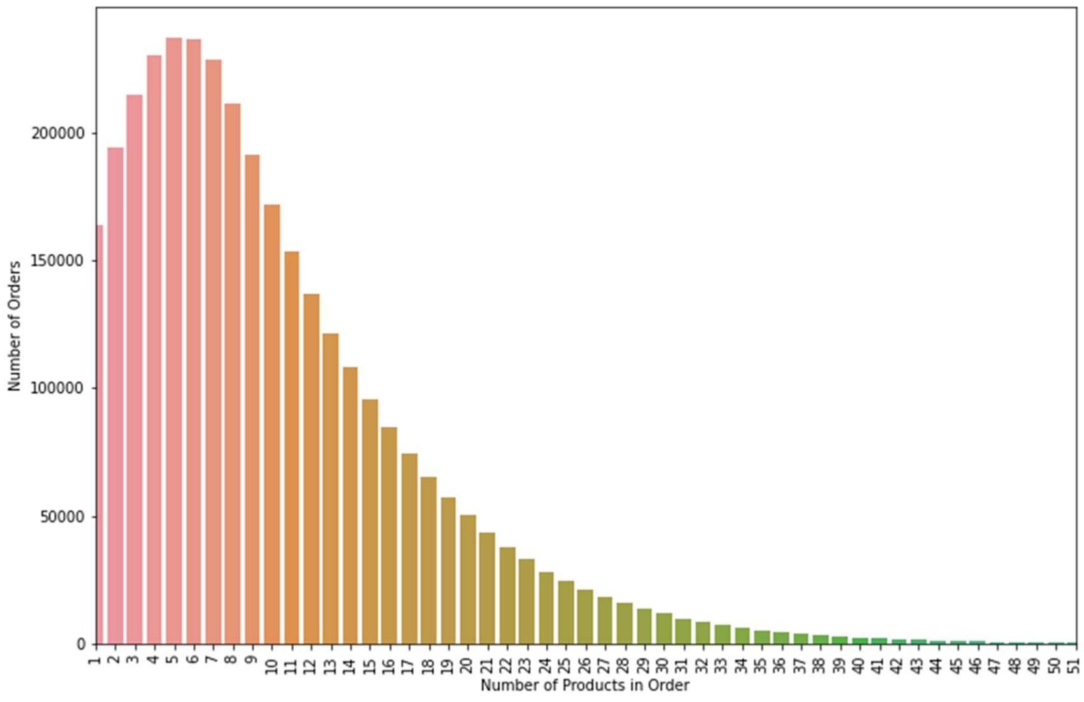


Fig: 5.1.6 Orders, Split by Product Count

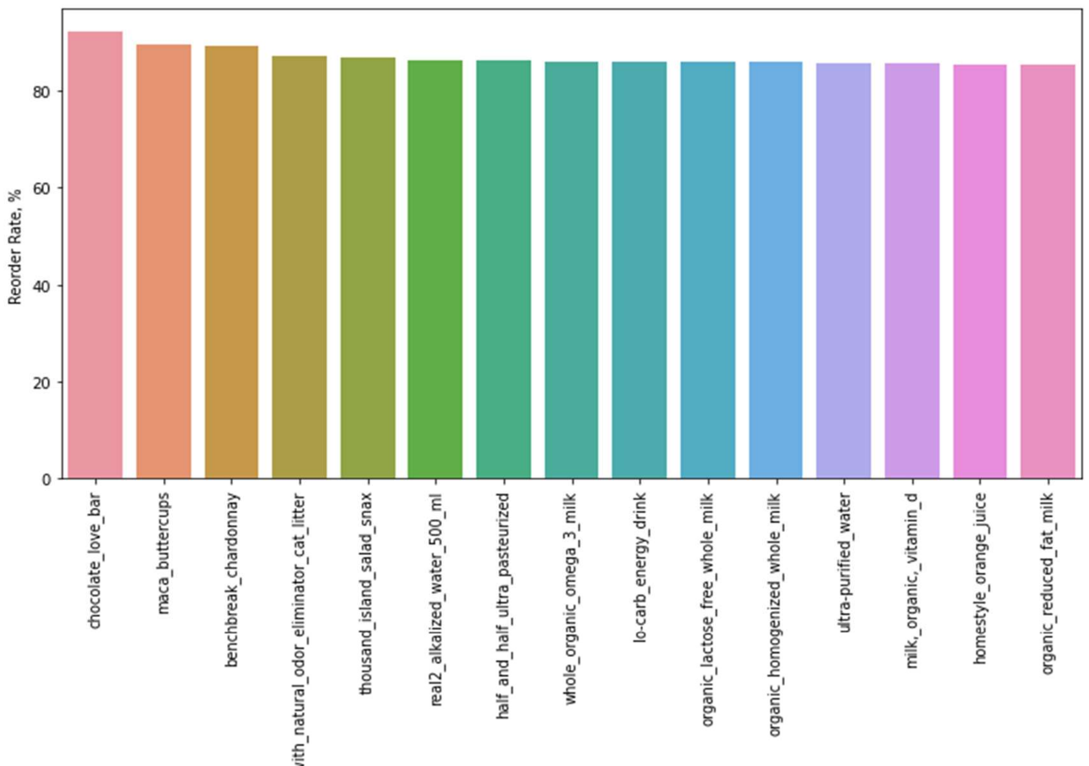


Fig: 5.1.7 Products with the Highest Reorder Rate

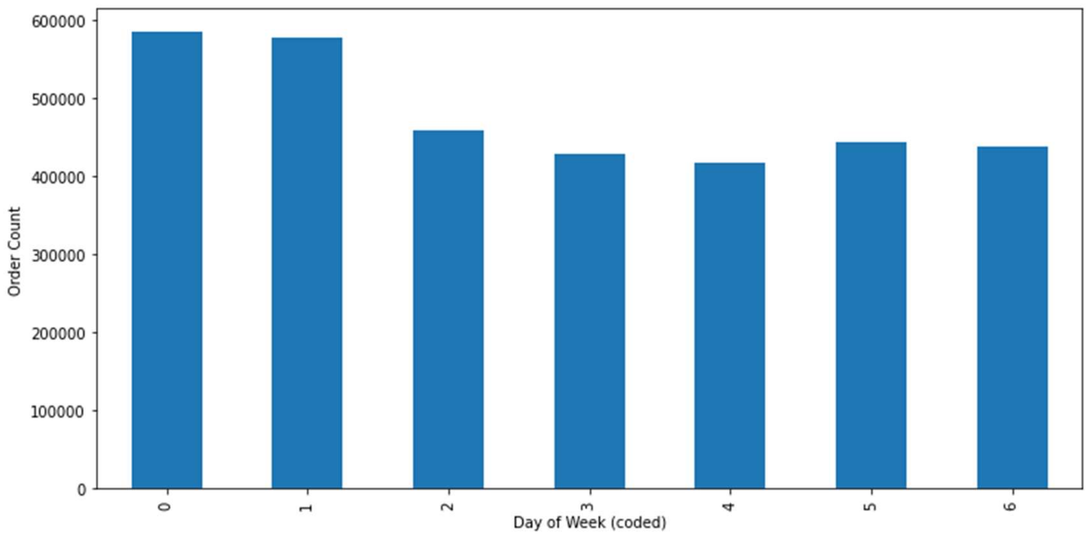


Fig: 5.1.8 Orders, Split by Day of Week

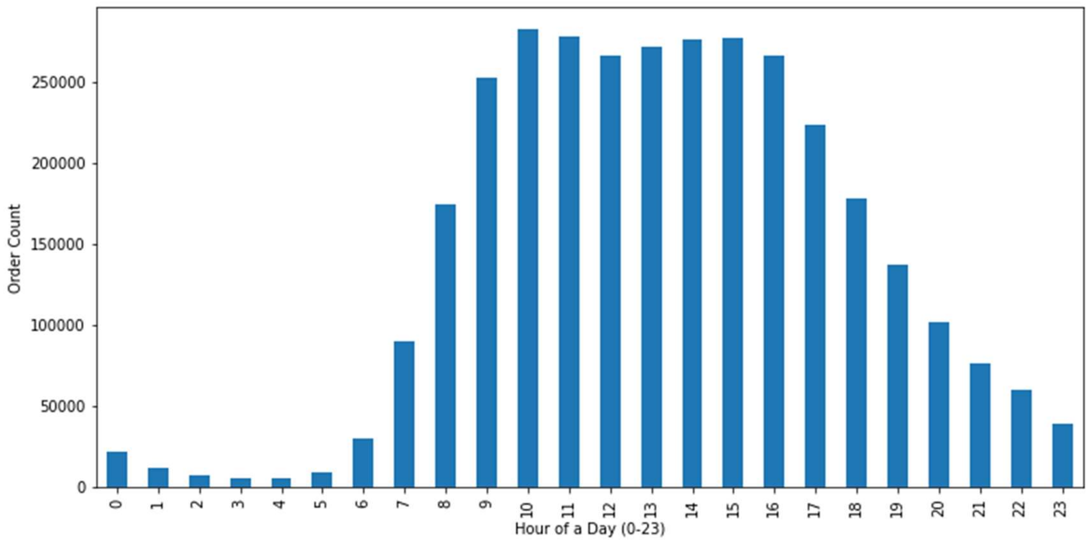


Fig: 5.1.9 Orders, Split by Hour

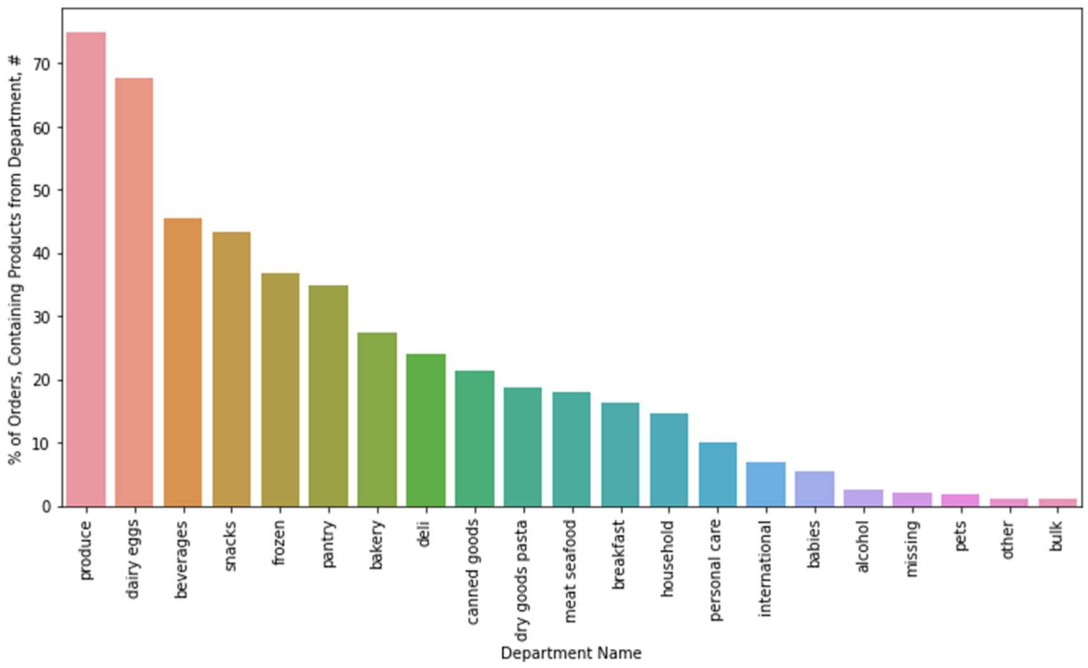


Fig: 5.1.10 Most Popular Departments

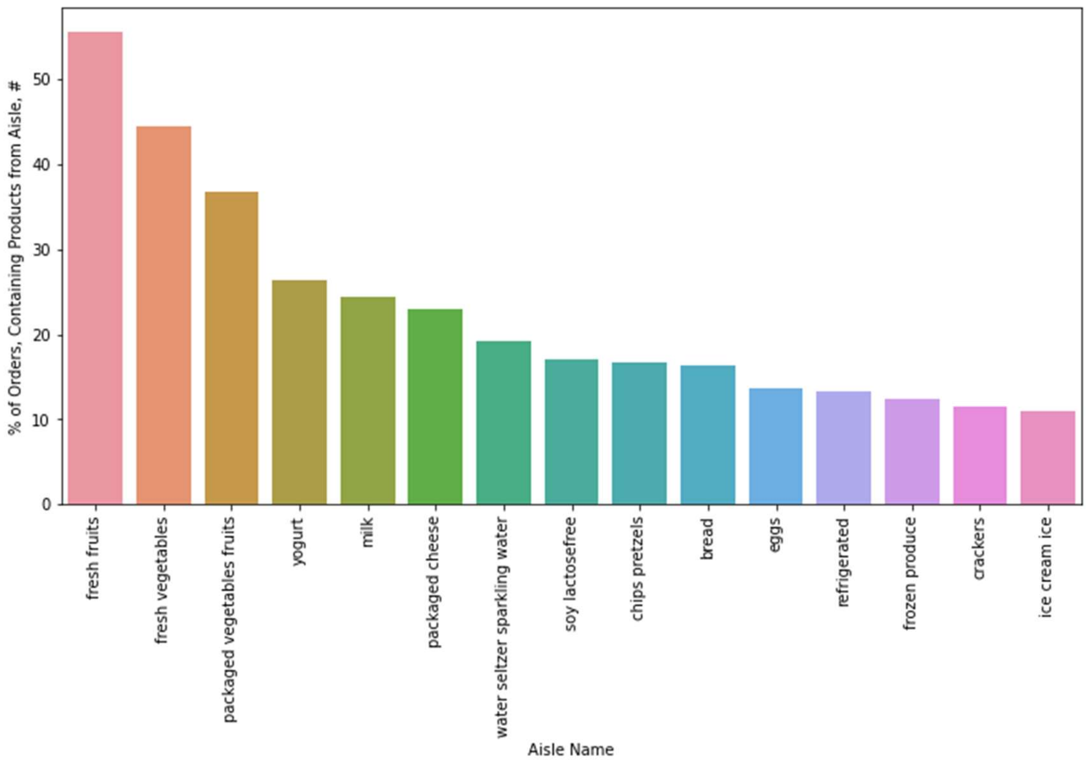


Fig: 5.1.11 Most Popular Aisles

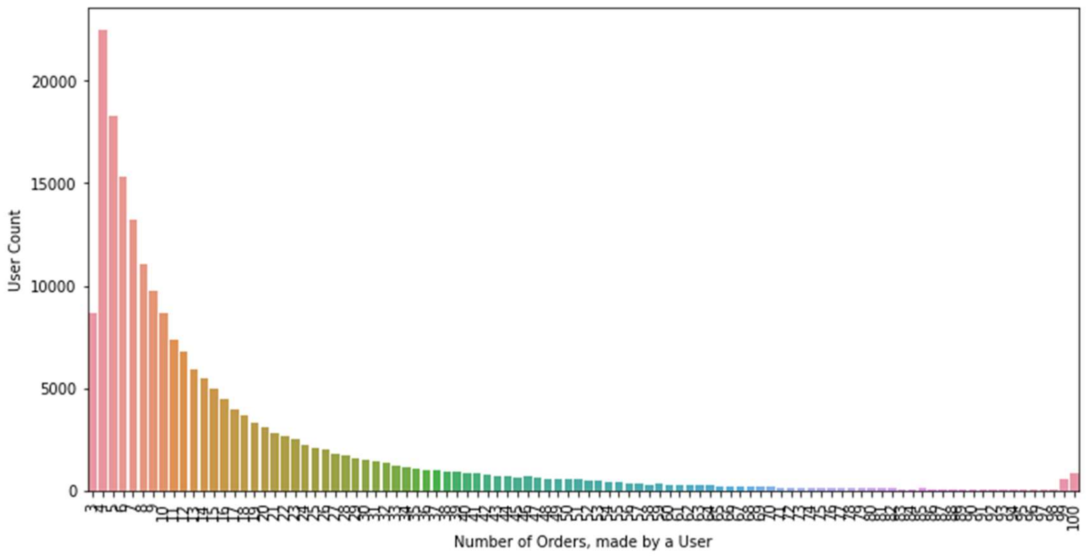


Fig: 5.1.12 Distribution of Order Count per User

\

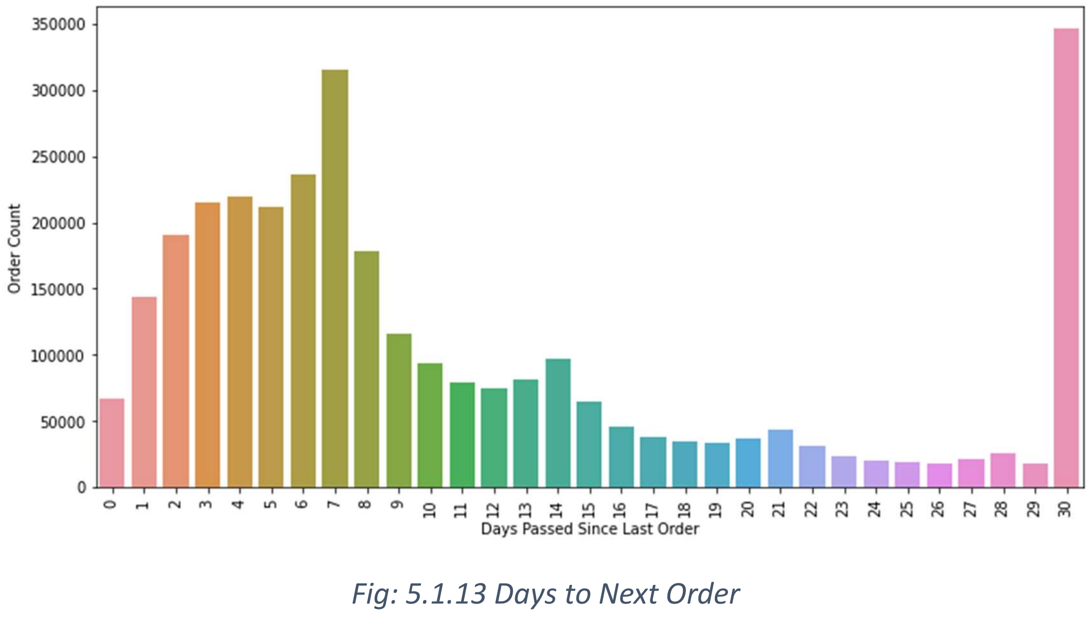


Fig: 5.1.13 Days to Next Order

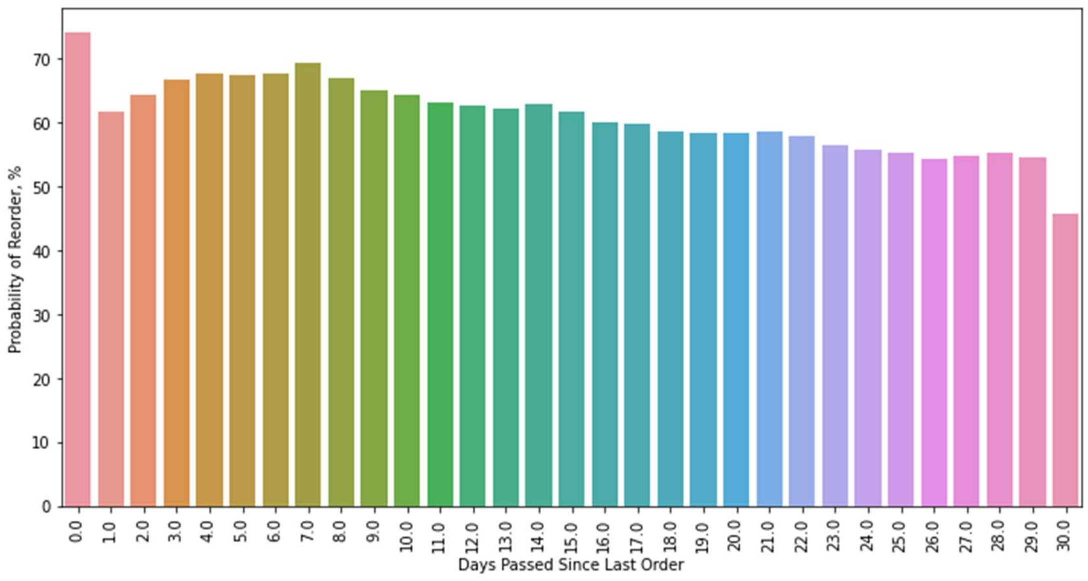


Fig: 5.1.14 Time of last order vs. probability of reorder

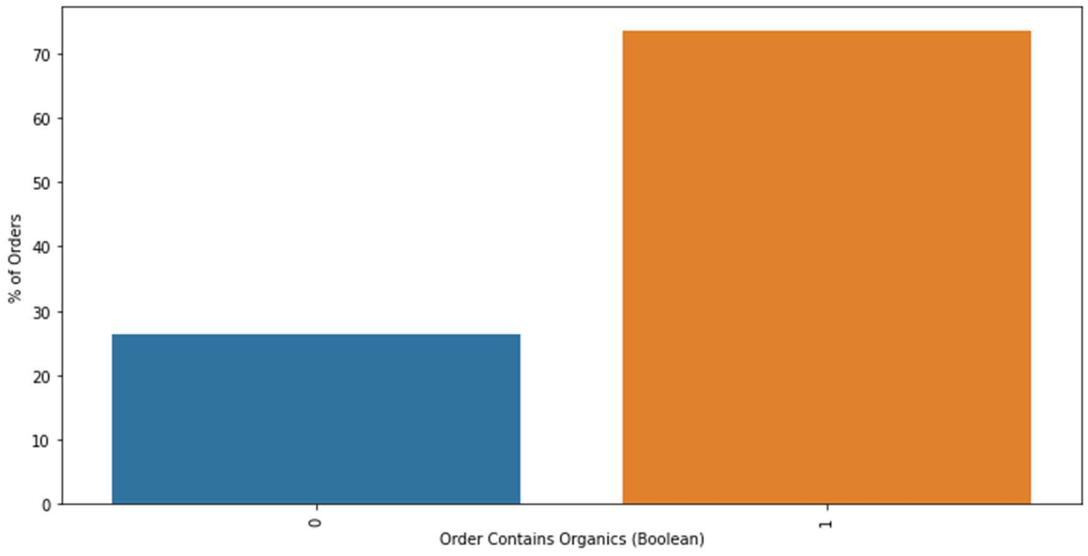
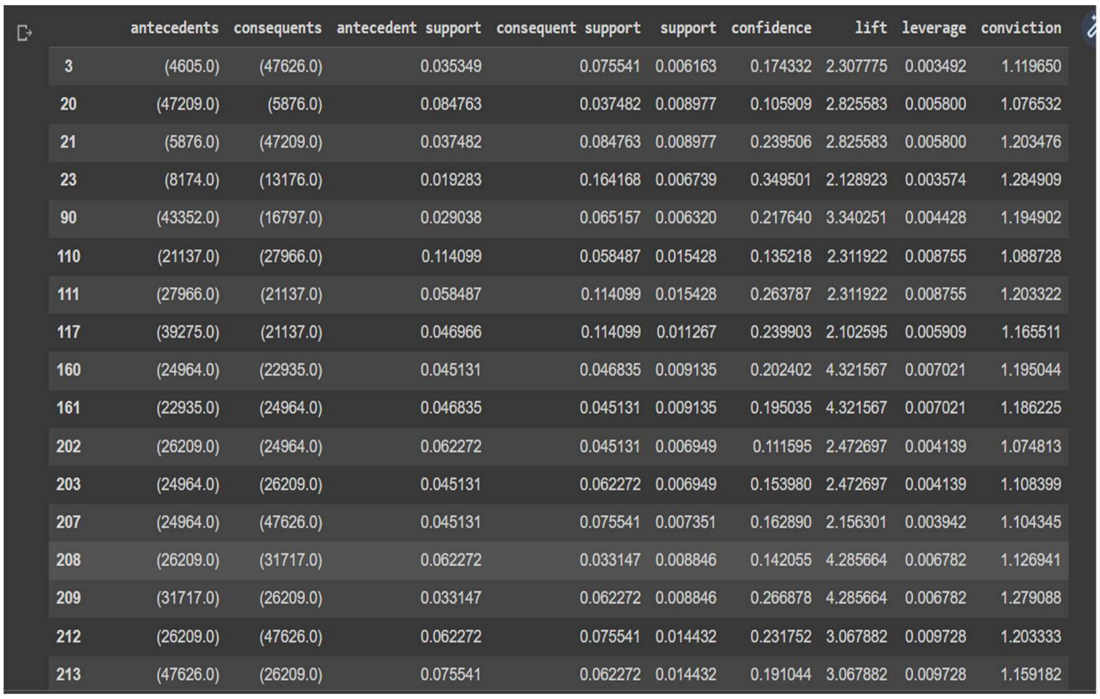


Fig: 5.1.15 Share of organic/non-organic products and correspondent orders count

5.2 MARKET BASKET ANALYSIS



	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
3	(4605.0)	(47626.0)	0.035349	0.075541	0.006163	0.174332	2.307775	0.003492	1.119650
20	(47209.0)	(5876.0)	0.084763	0.037482	0.008977	0.105909	2.825583	0.005800	1.076532
21	(5876.0)	(47209.0)	0.037482	0.084763	0.008977	0.239506	2.825583	0.005800	1.203476
23	(8174.0)	(13176.0)	0.019283	0.164168	0.006739	0.349501	2.128923	0.003574	1.284909
90	(43352.0)	(16797.0)	0.029038	0.065157	0.006320	0.217640	3.340251	0.004428	1.194902
110	(21137.0)	(27966.0)	0.114099	0.058487	0.015428	0.135218	2.311922	0.008755	1.088728
111	(27966.0)	(21137.0)	0.058487	0.114099	0.015428	0.263787	2.311922	0.008755	1.203322
117	(39275.0)	(21137.0)	0.046966	0.114099	0.011267	0.239903	2.102595	0.005909	1.165511
160	(24964.0)	(22935.0)	0.045131	0.046835	0.009135	0.202402	4.321567	0.007021	1.195044
161	(22935.0)	(24964.0)	0.046835	0.045131	0.009135	0.195035	4.321567	0.007021	1.186225
202	(26209.0)	(24964.0)	0.062272	0.045131	0.006949	0.111595	2.472697	0.004139	1.074813
203	(24964.0)	(26209.0)	0.045131	0.062272	0.006949	0.153980	2.472697	0.004139	1.108399
207	(24964.0)	(47626.0)	0.045131	0.075541	0.007351	0.162890	2.156301	0.003942	1.104345
208	(26209.0)	(31717.0)	0.062272	0.033147	0.008846	0.142055	4.285664	0.006782	1.126941
209	(31717.0)	(26209.0)	0.033147	0.062272	0.008846	0.266878	4.285664	0.006782	1.279088
212	(26209.0)	(47626.0)	0.062272	0.075541	0.014432	0.231752	3.067882	0.009728	1.203333
213	(47626.0)	(26209.0)	0.075541	0.062272	0.014432	0.191044	3.067882	0.009728	1.159182

Fig: 5.2.1 Generated Association Rules in tabular form

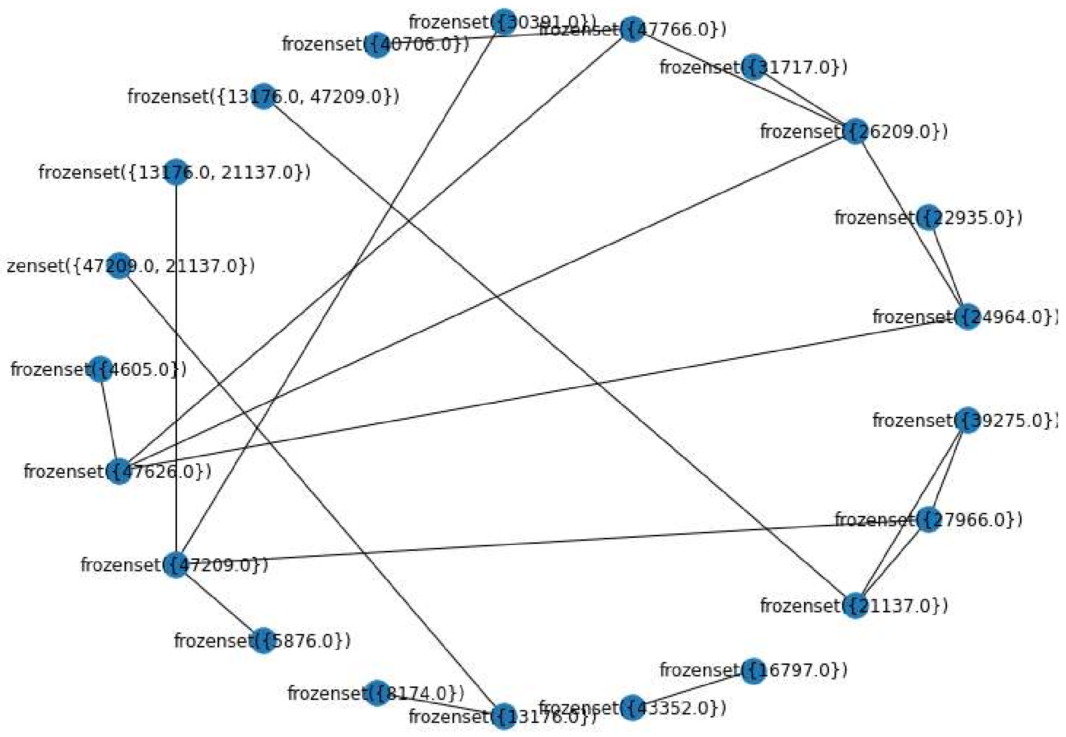


Fig: 5.2. 1: Association Rules in Graphical form

CHAPTER 6

CONCLUSION

6. CONCLUSION

Market Basket Analysis is a conceptual framework that originates in the marketing field and has been used effectively in fields such as bioinformatics, nuclear science, immunology, and geophysics more recently. One reason for MBA's increasing adoption across scientific fields is that by using an inductive approach to theorizing, researchers are able to evaluate the existence of association rules.

Considering all, by summing up the whole, we agree that this model can have an efficient impact on marketing and sales research that can be used to make strategic business decisions.

In conclusion, Market Basket Analysis is a powerful technique for understanding customer purchasing behaviour in the retail industry. It provides insights into the relationships between items purchased, allowing organizations to make informed decisions about product placement, promotions, and inventory management.

By optimizing these aspects of their operations, organizations can increase sales, reduce costs, and improve customer satisfaction. Despite its limitations, such as data quality and the difficulty of interpreting results, Market Basket Analysis remains a valuable tool for organizations looking to make data-driven decisions and improve their overall performance. By leveraging the insights generated from Market Basket Analysis, organizations can gain a competitive advantage and drive growth and success in the retail industry.

CHAPTER 7
FUTURE SCOPE

7. FUTURE SCOPE

The future scope of Market Basket Analysis using the Apriori algorithm is promising and has a lot of potential for growth. Some of the key areas where the Apriori algorithm and Market Basket Analysis can be applied in the future include:

Big Data Analytics: With the increasing amount of data generated by retailers, the Apriori algorithm can be used to analyse big data and uncover valuable insights that were previously not possible to obtain.

Predictive Analytics: The Apriori algorithm can be integrated with predictive analytics to forecast future purchasing behaviour and inform business decisions.

Personalized Recommendations: Market Basket Analysis using the Apriori algorithm can be used to personalize product recommendations for customers, increasing the relevance of promotions and advertisements.

Customer Segmentation: The Apriori algorithm can be used to segment customers based on their purchasing behaviour, allowing retailers to personalize promotions and target advertisements more effectively.

Artificial Intelligence and Machine Learning: Market Basket Analysis using the Apriori algorithm can be integrated with artificial intelligence and machine learning to provide even more advanced and accurate insights.

Supply Chain Optimization: Market Basket Analysis can be applied to the supply chain to optimize inventory management and reduce costs.

These are just a few examples of the potential applications of Market Basket Analysis using the Apriori algorithm in the future. As the retail industry continues to evolve and technology advances, the future scope of Market Basket Analysis is likely to expand even further, providing even more opportunities for organizations to leverage the power of data to drive growth and success.

CHAPTER 8

REFERENCES

8. REFERENCES

[1] Agrawal, R., and R. Srikant. 1994. Fast algorithms for mining association rules. In Proceedings of 20th International Conference on Very Large Data Bases, VLDB, Vol. 1215, 487–99. Santiago, Chile: IBM Almaden Research Centre

[2] Singh, A. and Sinwar, D. (2017) 'Optimization of Association Rule Mining Using FP_Growth Algorithm with GA'. International Journal for Research in Applied Science and Engineering Technology.

[3] A Survey of Patients With Self-Reported Severe Food Allergies in Japan by T Imamura 1, Y Kanagawa, M Ebisawa

[4] S. Rangaswamy, Shobha G., "Optimized Association Rule Mining Using Genetic Algorithm," Journal of Computer Science Engineering and information Technology Research (JCSEITR), Vol.2, Issue 1, pp 1-9, 2012.

[5] S. Jain, S. Kabra. "Mining & Optimization of Association Rules Using Effective Algorithm," International journal of Emerging Technology and Advanced Engineering (IJETAE), Vol.2, Issue 4, 2012.

[6] Berry and Linoff. Data Mining Techniques for Marketing, Sales and Customer Relationship Management (second edition), Hungry Minds Inc., 2004

[7] I. Cil, "Consumption universes based supermarket layout through association rule mining and multidimensional scaling," Expert Syst. Appl., vol. 39, no. 10, pp. 8611–8625, 2012.

[8] Chen, Y.-L. et al. (2005) 'Market Basket Analysis in a Multiple Store Environment'. Decis. Support Syst. DOI: 10.1016/j.dss.2004.04.009.

[9] Shruthi Gurudath (2020) Market Basket Analysis & Recommendation System Using Association Rules. (Article)

[10] Aulakh, R.K. (2015) 'Optimized Association Rule Mining with Maximum Constraints Using Genetic Algorithm'. 3, p. 10.

[11] Gupta, R. (2013). Finding what to Sell Next to your Customer. Retrieved july 25, 2014