

A
COAPPS PROJECT
ON
Grocery Genius: Smart Grocery Shopping Assistant with Basket Analysis

SUBMITTED BY

BEECHU SHASHANK REDDY

PANJALA SRI SAI

PAPPALA BALAJI

MUCHARLA RAVITEJA

FROM

GURUNANAK INSTITUTE OF TECHNOLOGY (HYDERABAD)

SUBMITTED TO



Grocery Genius: Smart Grocery Shopping Assistant with Basket Analysis Using Association Rule Mining

ABSTRACT

In today's fast-paced world, grocery shopping is an essential yet time-consuming task for many individuals. With the advent of technology, the traditional grocery shopping experience is undergoing a significant transformation. This abstract presents a novel concept: the Smart Grocery Shopping Assistant with Basket Analysis.

This system leverages advanced data analytics techniques to enhance the efficiency and convenience of grocery shopping. By integrating machine learning algorithms and real-time data processing, the Smart Grocery Shopping Assistant provides personalized recommendations and insights to shoppers, thereby streamlining the shopping experience.

One of the key features of this system is Basket Analysis, a data mining technique that examines the contents of a shopper's basket to identify patterns and associations among purchased items. By analysing past purchase history and preferences, the system can suggest complementary or related products, helping shoppers make informed decisions and discover new items of interest.

Furthermore, the Smart Grocery Shopping Assistant offers dynamic shopping lists that adapt based on individual preferences, dietary restrictions, and budgetary constraints. Through seamless integration with mobile devices and smart home technology, users can access their shopping lists anytime, anywhere, and receive real-time updates on product availability and promotions.

Moreover, the system provides valuable insights to retailers by aggregating anonymized data on shopping patterns and consumer behaviour. This enables retailers to optimize inventory management, product placement, and marketing strategies, ultimately enhancing customer satisfaction and driving business growth.

TABLE OF CONTENTS

CHAPTER NO.	TITLE	PAGE NO.
1.	CHAPTER 1: INTRODUCTION 1.1 INTRODUCTION 1.2 SCOPE OF THE PROJECT 1.3 OBJECTIVE 1.4 EXISTING SYSTEM 1.5 PROPOSED SYSTEM	6-9 6-8 9 9 9 9
2.	CHAPTER 2: PROJECT DESCRIPTION 2.1 GENERAL 2.2 LIBRARIES	10-12 10 11-12
3.	CHAPTER 3: SYSTEM DESIGN 3.1 SYSTEM ARCHITECTURE 3.2 MODULE DESCRIPTION 3.3 PROJECT FLOW DIAGRAM	13-14 13 13 14
4.	CHAPTER 4: REQUIREMENTS 4.1 SOFTWARE REQUIREMENTS 4.2 HARDWARE REQUIREMENTS 4.3 FUNCTIONAL REQUIREMENTS	15-17 15 16 17
5.	CHAPTER 3: APPROACH TO PROJECT 5.1 DATA COLLECTION 5.2 LOADING DATA	18-41 18 19

	5.3 DATA PRE-PROCESSING	20-23
	5.4 EXPLORATORY DATA ANALYSIS	24-27
	5.5 TRAINING MODEL	28-39
	5.6 LOADING AND DUMPING THE MODEL	40
	5.7 DEPLOYING THE MODEL	41
6.	CHAPTER 6: DEVELOPMENT TOOLS	42-43
	6.1 FEATURES OF PYTHON	42
	6.2 JUPYTER LAB	42
	6.3 VISUAL STUDIO CODE	42
	6.4 STREAMLIT (A PYTHON FRAMEWORK)	43
	6.5 GIT AND GITHUB	43
7.	CHAPTER 7: IMPLEMENTATION	44-53
	7.1 IMPLEMENTATION CODE	44-53
8.	CHAPTER 8: SNAPSHOTS	54-55
	8.1 VARIOUS SNAPSHOTS	54-55
9.	CHAPTER 9: FUTURE ENHANCEMENT	56
	9.1 FUTURE ENHANCEMENTS	56
10.	CHAPTER 10: CONCLUSION AND REFERENCES	57-58
	10.1 CONCLUSION	57
	10.2 REFERENCES	58

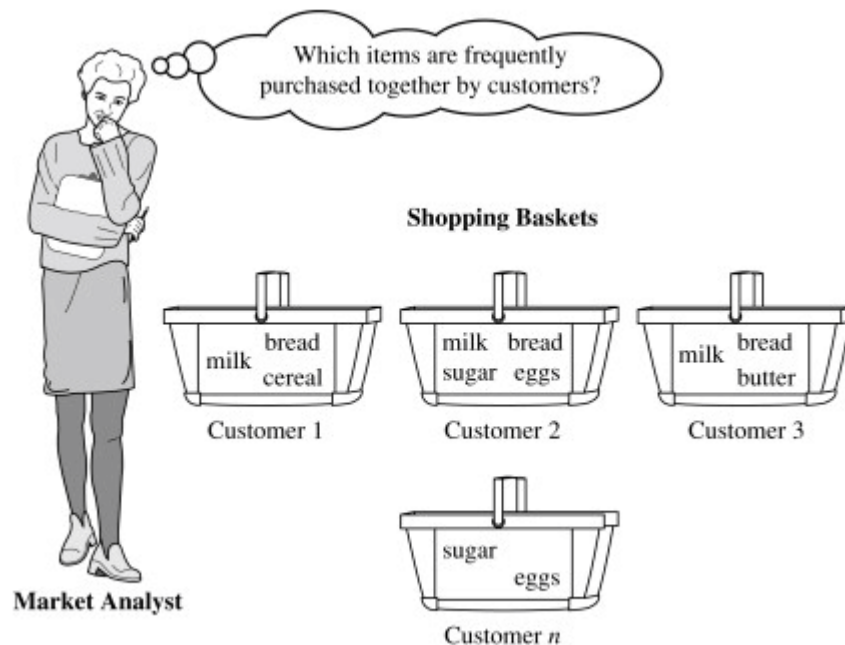
CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Market basket analysis is a strategic data mining technique used by retailers to enhance sales by gaining a deeper understanding of customer purchasing patterns. This method involves examining substantial datasets, such as historical purchase records, to unveil inherent product groupings and identify items that customers tend to buy together.

By recognizing these patterns of co-occurrence, retailers can make informed decisions to optimize inventory management, devise effective marketing strategies, employ cross-selling tactics, and even refine store layout for improved customer engagement. For example, if customers are buying milk, how probably are they to also buy bread (and which kind of bread) on the same trip to the supermarket? This information may lead to an increase in sales by helping retailers to do selective marketing based on predictions, cross-selling, and planning their ledge space for optimal product placement.



Now, just think of the universe as the set of items available at the store, then each item has a Boolean variable that represents the presence or absence of that item. Now, we can represent each basket with a Boolean vector of values assigned to these variables. We can then analyse the Boolean vectors to identify purchase patterns that reflect items

frequently associated or bought together, representing such patterns in the form of association rules.

How Does Market Basket Analysis Work?

- Collect data on customer transactions, such as the items purchased in each transaction, the time and date of the transaction, and any other relevant information.
- Clean and preprocess the data, removing any irrelevant information, handling missing values, and converting the data into a suitable format for analysis.
- Use association rules mining algorithms such as Apriori or FP-Growth to identify frequent item sets, sets of items often appearing together in a transaction.
- Calculate the support and confidence for each frequent itemset, expressing the likelihood of one item being purchased given the purchase of another item.
- Generate association rules based on the frequent item sets and their corresponding support and confidence values. Association rules indicate the likelihood of purchasing one item given the purchase of another item.
- Interpret the results of the market basket analysis, identifying frequent purchases, assessing the strength of the association between items, and uncovering other relevant insights into customer behaviour and preferences
- Use the insights from the market basket analysis to inform business decisions such as product recommendations, store layout optimization, and targeted marketing campaigns.

Applications of Market Basket Analysis

Industry	Applications of Market Basket Analysis
Retail	

Industry	Applications of Market Basket Analysis
	Identify frequently purchased product combinations and create promotions or cross-selling strategies
E-commerce	Suggest complementary products to customers and improve the customer experience
Hospitality	Identify which menu items are often ordered together and create meal packages or menu recommendations
Healthcare	Understand which medications are often prescribed together and identify patterns in patient behaviour or treatment outcomes
Banking/Finance	Identify which products or services are frequently used together by customers and create targeted marketing campaigns or bundle deals
Telecommunications	Understand which products or services are often purchased together and create bundled service packages that increase revenue and improve the customer experience

1.2 SCOPE OF THE PROJECT:

The scope of this project involves conducting Market Basket Analysis (MBA) using the FP-Growth algorithm, a robust method for discovering frequent item sets in large transactional datasets. Beginning with the collection and preprocessing of retail or e-commerce transaction data, the project aims to ensure data accuracy and consistency.

Implementation of the FP-Growth algorithm, either through development or utilizing existing libraries like mlxtend or pyfpgrowth in Python, will generate frequent item sets representing items frequently purchased together. Subsequently, the project will focus on association rule mining, extracting rules based on metrics such as support, confidence, and lift. Results will be visualized through graphs and tables, offering insights into customer purchasing behaviours. The project will also include performance evaluation, assessing the efficiency and scalability of FP-Growth, potentially comparing it with other MBA techniques like Apriori. Additionally, the project might include the development of a recommendation engine based on association rules for personalized product recommendations. Documentation will cover methodology, findings, and code, while a presentation will communicate the project's purpose, methodology, results, and potential business impacts. Possible enhancements could involve exploring time-based patterns and customer segmentation for deeper insights, ensuring scalability for larger datasets or real-time data processing.

1.3 OBJECTIVE

Objectives in market basket analysis define what the organization aims to achieve through this initiative. These objectives should be closely aligned with the problem statement and should provide a clear direction for the analysis. Objectives could include increasing revenue through targeted promotions, reducing inventory costs by optimizing product assortments, or enhancing customer satisfaction by personalizing recommendations. Each objective should be specific, measurable, achievable, relevant, and time-bound to ensure clarity and focus.

1.4 EXISTING SYSTEM

This section evaluates the current state of market basket analysis within the organization. It discusses any existing methods, tools, or systems being used for analysing customer purchase patterns. This could range from manual analysis of transaction data to the use of basic statistical techniques or legacy software solutions. It also identifies the limitations or shortcomings of the current approach, such as scalability issues, limited analytical capabilities, or lack of real-time insights.

1.5 PROPOSED SYSTEM

Here, the focus shifts to the envisioned solution for improving market basket analysis. This could involve implementing new technologies, adopting advanced analytics techniques, or redesigning existing processes to overcome the limitations of the current system. The proposed system should address the identified challenges and align with the objectives set forth earlier. It may involve the integration of data from multiple sources, the deployment of machine learning algorithms for pattern recognition, or the development of custom analytics tools tailored to the organization's needs.

CHAPTER 2

PROJECT DESCRIPTION

2.1 GENERAL

Market Basket Analysis (MBA) is a data mining technique used to uncover relationships between products purchased together by customers. This project focuses on implementing MBA using the FP-Growth algorithm, a powerful method known for its efficiency in handling large transactional datasets. The main objective is to discover frequent item sets, which are combinations of products frequently bought together, and to extract meaningful association rules from these item sets.

The project begins with the collection of transactional data from a retail or e-commerce dataset, ensuring that the data is clean and suitable for analysis. Once the data is prepared, the FP-Growth algorithm is implemented to efficiently mine the frequent item sets. These item sets provide valuable insights into customer purchasing behaviours and can reveal patterns such as "Customers who buy A and B are likely to also buy C."

After generating the frequent item sets, the project focuses on extracting association rules, which describe the relationships between products in terms of metrics like support, confidence, and lift. These rules are then analysed and visualized to provide clear and actionable insights. For instance, a rule with high confidence might indicate a strong association between two products, suggesting potential cross-selling opportunities.

The project also includes performance evaluation to assess the efficiency of the FP-Growth algorithm on the given dataset. Additionally, there is the possibility of building a recommendation engine based on the discovered association rules. This engine could provide personalized product recommendations to customers based on their current basket, enhancing the overall shopping experience.

Documentation of the project will cover the methodology used, the findings from the analysis, the code developed for implementing FP-Growth, and the insights gained from the association rules. The project description will be presented in a structured manner, detailing the steps followed in data preprocessing, algorithm implementation, rule extraction, and result interpretation. The goal is to provide a comprehensive understanding of how Market Basket Analysis was conducted using the FP-Growth algorithm and the insights it yielded for potential business applications.

2.2 LIBRARIES

NumPy:

NumPy is a fundamental package for scientific computing in Python. It provides support for large, multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays. NumPy's array object is the foundation for many other libraries in the Python data ecosystem. It offers efficient operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, and more. NumPy is often used in data manipulation, numerical simulations, statistical operations, and other tasks where efficient array processing is needed. NumPy stands as a fundamental pillar in the realm of scientific computing with Python, offering a robust array manipulation library. Its central object, the NumPy array, provides a foundation for handling multi-dimensional arrays efficiently. Data scientists, engineers, and researchers benefit from its suite of mathematical functions, enabling tasks like matrix operations, linear algebra, and Fourier transforms. NumPy's strength lies in its ability to process entire arrays at once, enhancing performance significantly over traditional loops.

Pandas:

Pandas is a powerful library for data manipulation and analysis in Python. It provides easy-to-use data structures like Data Frame and Series, which are designed to make working with structured data intuitive and efficient. With Pandas, you can read and write data from various file formats (CSV, Excel, SQL databases), clean and preprocess data, perform data exploration, and handle missing values. Pandas is especially useful for tasks such as data cleaning, transformation, aggregation, and visualization. It integrates well with other libraries like NumPy and Matplotlib, making it a go-to choice for data wrangling in data science projects.

Data scientists and analysts leverage Pandas for its efficiency in loading diverse data formats, handling missing values, and conducting insightful exploratory data analysis. This library's adaptability and efficiency make it an indispensable tool for professionals navigating the complexities of real-world data.

Matplotlib:

Matplotlib is a comprehensive library for creating static, interactive, and animated visualizations in Python. It provides a wide range of plotting functions to create line plots, scatter plots, bar plots, histograms, pie charts, and more. Matplotlib aims to make it easy to generate high-quality plots with customizable features. You can control every aspect of a plot, including axes, labels, colours, and styles. Matplotlib is often used for data visualization in exploratory data analysis (EDA), scientific plotting, presentation graphics, and creating publication-ready figures. It also serves as the foundation for other plotting libraries and tools in the Python ecosystem. Researchers, scientists, and analysts benefit from its precise control over plot aesthetics, ensuring publication-ready figures.

Scikit-learn:

Scikit-learn is a popular machine learning library in Python that provides simple and efficient tools for data mining and data analysis. It features various algorithms for classification, regression, clustering, dimensionality reduction, model selection, and preprocessing. Scikit-learn is built on NumPy, SciPy, and Matplotlib, making it easy to integrate with these libraries. It offers a consistent interface and a well-documented API, making it accessible for both beginners and experts in machine learning. With scikit-learn, you can build and train machine learning models, evaluate their performance, tune hyperparameters, and deploy models in production environments. Data preprocessing, feature extraction, and model evaluation utilities further streamline the machine learning workflow. The library's interoperability with NumPy and Pandas ensures seamless integration with data processing pipelines. Machine learning projects, whether simple or intricate, benefit from scikit-learn's versatility and performance, making it a go-to choice for tackling diverse machine learning tasks.

mlxtend:

mlxtend is a library built on top of scikit-learn and provides additional functionality for machine learning tasks. It includes various tools and extensions to complement scikit-learn's capabilities, offering implementations of popular algorithms, convenience functions, and helper utilities. Mlxtend extends scikit-learn with functionalities like association rule mining, feature selection, stacking classifiers, and visualization tools. For example, it provides an easy-to-use implementation of the FP-Growth algorithm for frequent pattern mining, which is particularly useful for Market Basket Analysis. Mlxtend is designed to be compatible with scikit-learn, enabling users to seamlessly combine its functionalities with scikit-learn's machine learning workflows. Additionally, it offers utilities for stacking classifiers, creating ensemble models, and visualizing model performance. Experimentation with various machine learning techniques becomes seamless with mlxtend, thanks to its compatibility with scikit-learn. The library adds an extra layer of functionality to machine learning pipelines, enabling users to delve into association rule exploration, ensemble modelling, and model visualization with ease and efficiency.

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

The system architecture provides a blueprint for the entire market basket analysis solution, detailing its structural components and their interactions. It typically includes:

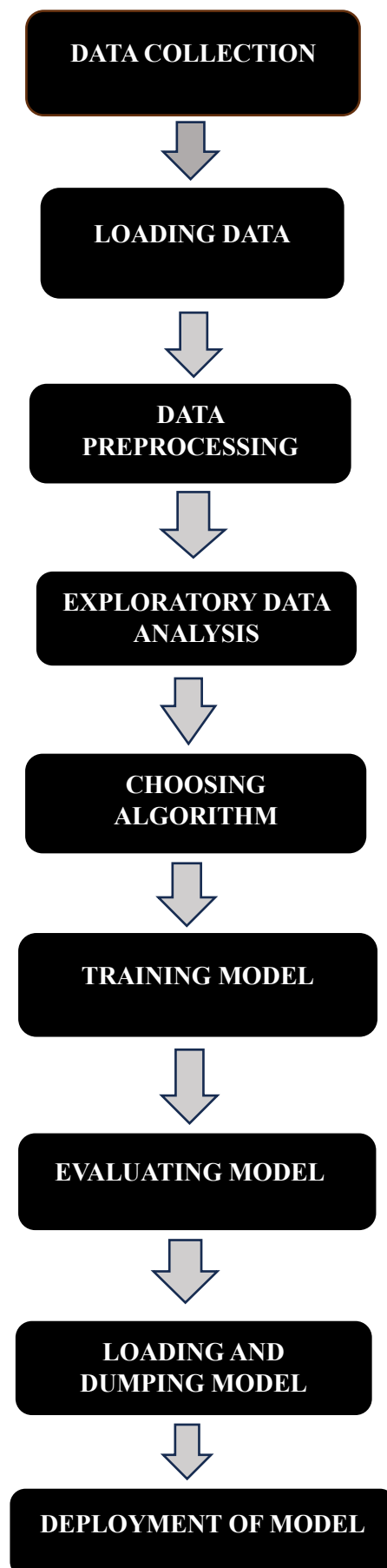
- **Components:** Identifies the major components of the system, such as data sources, processing engines, analytics modules, and user interfaces.
- **Interactions:** Describes how these components communicate and collaborate to fulfil the system's objectives. This may involve synchronous or asynchronous communication protocols, message queues, APIs, or event-driven architectures.
- **Deployment Environment:** Specifies the deployment topology, including hardware infrastructure, cloud services, and network configurations. It addresses concerns related to scalability, reliability, security, and performance optimization.
- **Scalability and Fault Tolerance:** Discusses strategies for scaling the system horizontally or vertically to handle increasing data volumes and user loads. It also considers fault tolerance mechanisms to ensure uninterrupted operation in the event of system failures or disruptions.

3.2 MODULE DESCRIPTION

The module description provides detailed insights into the functional modules or components that constitute the market basket analysis system. It includes:

- **Module Name and Purpose:** Clearly identifies the name and purpose of each module, highlighting its role in the system architecture.
- **Functionality and Operations:** Describes the core functionalities and operations performed by each module, including data processing, analysis algorithms, and user interactions.
- **Input and Output:** Specifies the inputs required by each module to perform its tasks and the outputs generated as a result. This may include raw data, parameters, configuration settings, and result summaries.

3.3 PROJECT FLOW DIAGRAM



CHAPTER 4

REQUIREMENTS

4.1 SOFTWARE REQUIREMENTS

This section specifies the software tools and technologies required to support the proposed market basket analysis solution. It includes both off-the-shelf software products and custom-developed applications. Common software requirements may include data mining and machine learning platforms for analysing transaction data, database management systems for storing and retrieving large volumes of data, visualization tools for presenting insights in a user-friendly manner, and possibly integration frameworks for connecting disparate data sources. The selection of software should be based on factors such as functionality, scalability, ease of use, and compatibility with existing IT infrastructure.

1. Data Mining & ML Platforms:

scikit-learn: Python's versatile ML library, perfect for FP-Growth implementation.

mlxtend: Enhances scikit-learn with FP-Growth for association rule mining.

2. Database Management Systems (DBMS):

MySQL/PostgreSQL: Reliable RDBMS for scalable storage of transactional data.

SQLite: Lightweight option for smaller projects.

MongoDB: NoSQL flexibility for unstructured data handling.

3. Visualization Tools:

Matplotlib & Seaborn: Python's go-to for visualizing frequent item sets and rules.

Tableau/Power BI: Dynamic dashboards to present MBA insights interactively.

4. Integration Frameworks:

Apache Kafka: Real-time data processing for streaming transaction data.

Apache NiFi: Streamlines data flows, useful for preprocessing transactions.

Apache Spark: MLlib for scalable ML on large datasets.

5. Custom-Developed Apps:

Python Scripts: Preprocessing, FP-Growth implementation, rule mining.

Jupyter Notebooks: Interactive data exploration and result visualization.

Web Apps (Flask/Django): User interfaces for exploring MBA results.

6. Version Control & IDEs:

Git: Essential for collaboration and managing code versions.

IDEs (PyCharm/JupyterLab/VS Code): Writing, testing, and debugging Python code.

8. Other Tools & Considerations:

Docker: Containerization for consistent deployment across environments.

Virtual Environments (virtualenv/conda): Managing dependencies for reproducibility.

Documentation (Sphinx/MkDocs): Comprehensive project documentation.

CI/CD (Jenkins/GitHub Actions): Automated testing

4.2 HARDWARE REQUIREMENTS

Hardware requirements detail the computing infrastructure needed to support the software components of the market basket analysis solution. This includes servers, workstations, storage devices, and networking equipment. The hardware specifications should be determined based on factors such as the volume of data to be processed, the complexity of analytical algorithms, and the expected workload. Scalability and reliability considerations are also crucial, especially for organizations dealing with large-scale data analytics. Additionally, factors like budget constraints and future growth projections may influence hardware selection decisions.

In addition to software tools, the hardware requirements for Market Basket Analysis (MBA) must be considered to support the solution's computing needs. The computing infrastructure should be tailored to handle the processing demands of the software components efficiently. This includes servers with sufficient processing power and memory to execute complex analytical algorithms, especially when dealing with large volumes of transactional data. Workstations for data analysts should have ample resources to run the software tools smoothly, enabling interactive data exploration and model development. Storage devices such as high-capacity drives or cloud storage solutions are essential for storing transactional data and model outputs. Networking equipment is crucial for facilitating data transfer between components, especially in distributed computing environments. Scalability is paramount, with hardware capable of accommodating potential growth in data volume and analysis complexity. Reliability is also critical, ensuring minimal downtime for continuous analysis.

4.3 FUNCTIONAL REQUIREMENTS

The Market Basket Analysis project's functional requirements encompass various critical aspects of the system's capabilities. First, the system must efficiently preprocess transactional data, including handling missing values and transforming raw data into a suitable format for the FP-Growth algorithm. Implementing the FP-Growth algorithm itself is essential, allowing the system to mine frequent item sets accurately. Users should have the flexibility to set support thresholds and adjust parameters for the algorithm as needed. The system should then be capable of extracting association rules from these item sets, providing options to filter rules based on metrics like support, confidence, and lift. Visualizing these rules and item sets is crucial, requiring the system to create interactive dashboards with various plot options for data exploration. Users should be able to customize these visualizations according to their preferences. Reporting and export functionalities are necessary for summarizing results and exporting them in commonly used formats like CSV or Excel, with the ability to schedule automated reports for convenience.

User management features are also vital, including defining roles with different access levels and ensuring secure authentication for data privacy. Scalability and performance are critical, with the system needing to handle large datasets efficiently and respond promptly to user queries. Integration capabilities, such as connecting to external databases and providing APIs, enhance the system's usability and interoperability. Error handling functionalities ensure the system provides informative messages for any issues encountered and logs system activities for auditing purposes. Internationalization support for multiple languages and customizable date formats caters to diverse user bases. Security measures, including data encryption and role-based access control, are essential for compliance with data protection regulations. Lastly, user assistance features like contextual help within the application and comprehensive training materials ensure users can effectively utilize the system's functionalities. These functional requirements collectively define the system's capabilities to process data, extract insights, and provide a user-friendly environment for Market Basket Analysis.

CHAPTER 5

APPROACH TO PROJECT

5.1 COLLECTION OF DATA

In the realm of Market Basket Analysis (MBA) machine learning projects, the collection of high-quality data is paramount for deriving meaningful insights into consumer behaviour and purchase patterns. The data collected typically consists of transactional records that detail the items purchased together during a single customer transaction. This data forms the foundation for identifying frequent item sets and generating association rules, which are fundamental for MBA.

Types of Data Sources:

Retail Transactions: Retailers often have transactional data readily available, capturing purchases made by customers at checkout. This data includes details such as item IDs, quantities, prices, and timestamps.

E-commerce Platforms: Online retailers have a wealth of transactional data, including information on products viewed, added to cart, and ultimately purchased.

Point-of-Sale (POS) Systems: In physical stores, POS systems record transactions, providing data on items sold together and customer purchase behaviour.

Market Research Databases: External sources like market research firms or syndicated data providers may offer purchase data across multiple retailers, allowing for broader analysis.

Challenges in Data Collection:

Data Volume: Dealing with large volumes of transactional data can strain resources and require efficient storage and processing solutions.

Data Variety: Transactional data can vary in formats and structures, especially when combining data from multiple sources.

Data Integration: Integrating data from diverse sources while maintaining consistency and accuracy can be challenging.

5.2 LOADING DATA

Loading data is the foundational step in any machine learning project, crucial for training models, testing algorithms, and deriving insights. In this comprehensive guide, we'll explore various strategies, tools, and best practices for loading data into machine learning projects. From handling diverse data formats to ensuring data quality and scalability, we'll cover everything you need to know.

Loading data into a machine learning project is a crucial step that sets the foundation for model training and analysis. Here's a step-by-step guide on how to effectively load data using Python and Pandas, one of the most popular libraries for data manipulation and analysis in machine learning.

Installing Required Libraries Before Loading The Data Is Mandatory For That Pip install Command Is Used

Importance of Data Loading:

Data is the life blood of machine learning projects, providing the fuel for training models and making predictions. Proper data loading ensures that the right information is available in the right format, setting the foundation for accurate analysis and model performance. By understanding the intricacies of data loading, data scientists and machine learning engineers can ensure that their models are built on reliable and well-prepared data.

Common Data Formats:

Data comes in various formats, each requiring specific handling methods:

CSV (Comma-Separated Values): Ideal for tabular data, commonly used in spreadsheets.

JSON (JavaScript Object Notation): Suitable for semi-structured and nested data, often found in web APIs.

Excel Spreadsheets: Widely used for small to medium-sized datasets with tabular structure.

Images (JPEG, PNG): Images need to be loaded into arrays for image classification tasks.

Text Data (TXT, PDF): NLP tasks require loading and processing text data.

Data Loading Tools and Libraries:

Pandas: A versatile library in Python for data manipulation and analysis. It excels at loading tabular data from CSV, Excel, and JSON files.

NumPy: Fundamental for numerical operations, often used in conjunction with Pandas.

requests: For making HTTP requests, useful for loading data from web APIs.

Beautiful Soup, Scrapy: For web scraping and loading web data.

5.3 DATA PREPROCESSING

Data preprocessing is a fundamental step in data analysis and machine learning, involving a series of transformations to raw data before it is used for modelling or analysis. The goal of data preprocessing is to clean, format, and organize data into a format that is suitable for machine learning algorithms. This process ensures that the data is accurate, complete, and in a form that allows models to learn effectively from it. Here's an overview of what data preprocessing entails, In General

1. Handling Missing Values:

Missing data is a common issue in datasets and can lead to biased or inaccurate results.

Strategies for handling missing values include:

Imputation: Filling missing values with the mean, median, or mode of the column.

Deletion: Removing rows or columns with missing values.

Prediction: Using machine learning algorithms to predict missing values based on other features.

2. Data Cleaning:

Data cleaning involves removing noise and inconsistencies in the data.

Tasks include:

Removing duplicates: Eliminating duplicate rows that may skew analysis.

Correcting errors: Fixing typos, inconsistencies in naming conventions, or erroneous entries.

Standardizing data: Ensuring data is in a consistent format, such as date formats or units of measurement.

3. Encoding Categorical Variables:

Machine learning algorithms typically require numerical inputs, so categorical variables need to be encoded.

Common methods include:

One-Hot Encoding: Creating binary columns for each category.

Label Encoding: Assigning a unique numerical value to each category.

4. Feature Scaling:

Features often have different scales, which can impact model performance.

Scaling ensures that all features contribute equally to the model.

Standardization: Scaling features to have a mean of 0 and a standard deviation of 1.

5. Handling Outliers:

Outliers are data points that significantly differ from the rest of the dataset.

Outliers can skew statistical analyses and model training.

Methods for handling outliers:

Trimming: Removing extreme values beyond a certain threshold.

Transformation: Applying mathematical transformations to normalize the data.

Imputation: Replacing outliers with a more typical value.

6. Data Transformation:

Transforming variables to meet the assumptions of the model.

Examples include:

Log Transformation: Converting skewed data to a more normally distributed form.

Box-Cox Transformation: A family of power transformations that includes the logarithm as a special case.

7. Handling Imbalanced Data:

In classification tasks, imbalanced classes (where one class has significantly more samples than others) can bias the model.

Techniques for dealing with imbalanced data:

Resampling: Oversampling the minority class or undersampling the majority class.

Synthetic Minority Over-sampling Technique (SMOTE): Generating synthetic samples for the minority class.

8. Feature Engineering:

Creating new features from existing ones to improve model performance.

Examples include:

Polynomial Features: Creating interaction terms or higher-order combinations of features.

Domain-specific transformations: Engineering features based on domain knowledge.

9. Data Partitioning:

Splitting the dataset into training, validation, and testing sets.

Training set: Used to train the model.

Validation set: Used for hyperparameter tuning and model selection.

Testing set: Used to evaluate the model's performance on unseen data.

Data Preprocessing in Market Basket Analysis:

In Market Basket Analysis (MBA), identifying unique items within transactional data is a fundamental step. This process involves extracting distinct items that appear in the dataset, which is essential for generating item sets and association rules.

Transaction Encoding:

Transaction encoding is a pivotal process in Market Basket Analysis (MBA), a technique used by retailers and businesses to uncover associations and patterns within customer transactions. In MBA, transaction data typically consists of records where each row represents a single transaction, and each column represents an item purchased. Encoding these transactions into a suitable format is essential for mining frequent item sets and deriving association rules. Let's delve into the details of transaction encoding in MBA:

1. Transaction Data Structure:

In MBA, transaction data is often in the form of a transaction-item matrix, where:

Rows represent transactions.

Columns represent distinct items available for purchase.

Each cell contains a binary value indicating whether an item was present in a transaction.

2. Binary Transaction Encoding:

The most common encoding method in MBA is binary encoding:

If an item is present in a transaction, it is represented as '1'.

If an item is not present, it is represented as '0'.

This binary representation simplifies the dataset and is efficient for frequent itemset mining algorithms like Apriori and FP-Growth.

3. Sparse Matrix Representation:

Since most transactions involve only a small subset of available items, the transaction-item matrix is typically sparse.

A sparse matrix is a matrix in which most entries are zero, making it memory-efficient.

Libraries like `scipy.sparse` in Python are used to handle such sparse matrices.

Benefits of Transaction Encoding:

Efficient Storage: Binary encoding reduces memory usage, especially for large transaction datasets.

Algorithm Compatibility: Many frequent itemset mining algorithms like Apriori and FP-Growth require data in this binary format.

Ease of Interpretation: The resulting matrix is straightforward to interpret, with '1's indicating item presence.

	Instant food products	UHT- milk	abrasive cleaner	artif. sweetener	baby cosmetics	baby food	bags	baking powder	bathroom cleaner	beef	...	turkey	vinegar	waffles	whipped cream	whisky	white bread	white wine	whole milk	yogurt	zwieback
0	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	True	False
2	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	False	False
3	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	True	False
4	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	True	False	False
...
9830	False	False	False	False	False	False	False	False	False	True	...	False	False	False	True	False	False	False	True	False	False
9831	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
9832	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	True	False
9833	False	False	False	False	False	False	False	False	False	False	...	False	False	False	False	False	False	False	False	False	False
9834	False	False	False	False	False	False	False	False	False	False	...	False	True	False	False	False	False	False	False	False	False
9835 rows × 169 columns																					

5.4 EXPLORATORY DATA ANALYSIS

Exploratory Data Analysis (EDA) is a crucial initial step in any data analysis project, providing a comprehensive overview of the dataset's characteristics and uncovering patterns, anomalies, and relationships within the data. It involves a series of techniques and visualizations to understand the structure, distribution, and quality of the data before diving into modelling or hypothesis testing. Here's an in-depth look at EDA and its significance in extracting meaningful insights from data:

1. Understanding the Dataset:

EDA begins with loading the dataset and getting a sense of its size, dimensions, and basic statistics.

Key steps include:

Displaying the first few rows to understand the data format and column names.

Checking data types (numeric, categorical, datetime) of each column.

Describing the dataset to get summary statistics such as mean, median, min, max, etc.

2. Exploring Data Distributions:

Understanding the distribution of data helps in selecting appropriate analysis methods.

Techniques for distribution exploration:

Histograms: To visualize the distribution of numerical variables.

Boxplots: Showing the median, quartiles, and outliers in the data.

Kernel Density Estimates (KDE): Providing a smoothed representation of the distribution.

3. Analysing Relationships:

EDA investigates relationships between variables to identify patterns and correlations.

Techniques include:

Scatter plots: Showing the relationship between two continuous variables.

Pair plots: Visualizing pairwise relationships in a grid of scatterplots.

Correlation matrices: Quantifying the strength and direction of linear relationships.

4. Feature Engineering Insights:

EDA can inspire the creation of new features to enhance model performance.

Insights for feature engineering may include:

Identifying interactions between variables that may be predictive.

Transforming variables to achieve linearity or normality.

5. Outlier Detection:

EDA uncovers outliers, data points that significantly differ from the rest of the dataset.

Techniques for outlier detection:

Boxplots and scatter plots: Identifying data points far from the main cluster.

Z-score or IQR methods: Quantifying the deviation of a data point from the mean or median.

6. Visualizing Trends and Patterns:

EDA uses visualizations to reveal trends, seasonality, and patterns in the data.

Common visualization techniques include:

Time series plots: To observe trends over time in temporal data.

Line plots: Showing trends and changes in data over different categories.

Heatmaps: Visualizing correlations or patterns in tabular data.

7. Detecting Skewness and Transformation:

Skewed data distributions can impact model performance, so EDA identifies and corrects skewness.

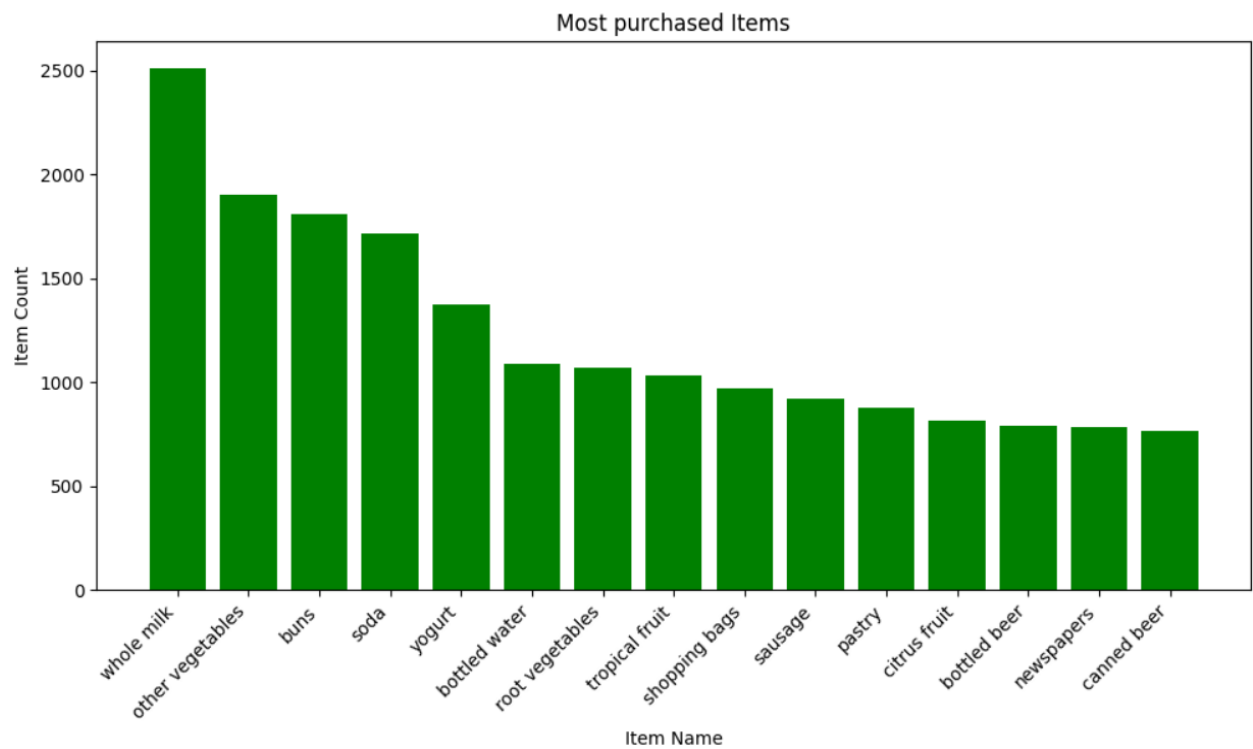
Techniques for handling skewness:

Logarithmic or Box-Cox transformations to normalize skewed data.

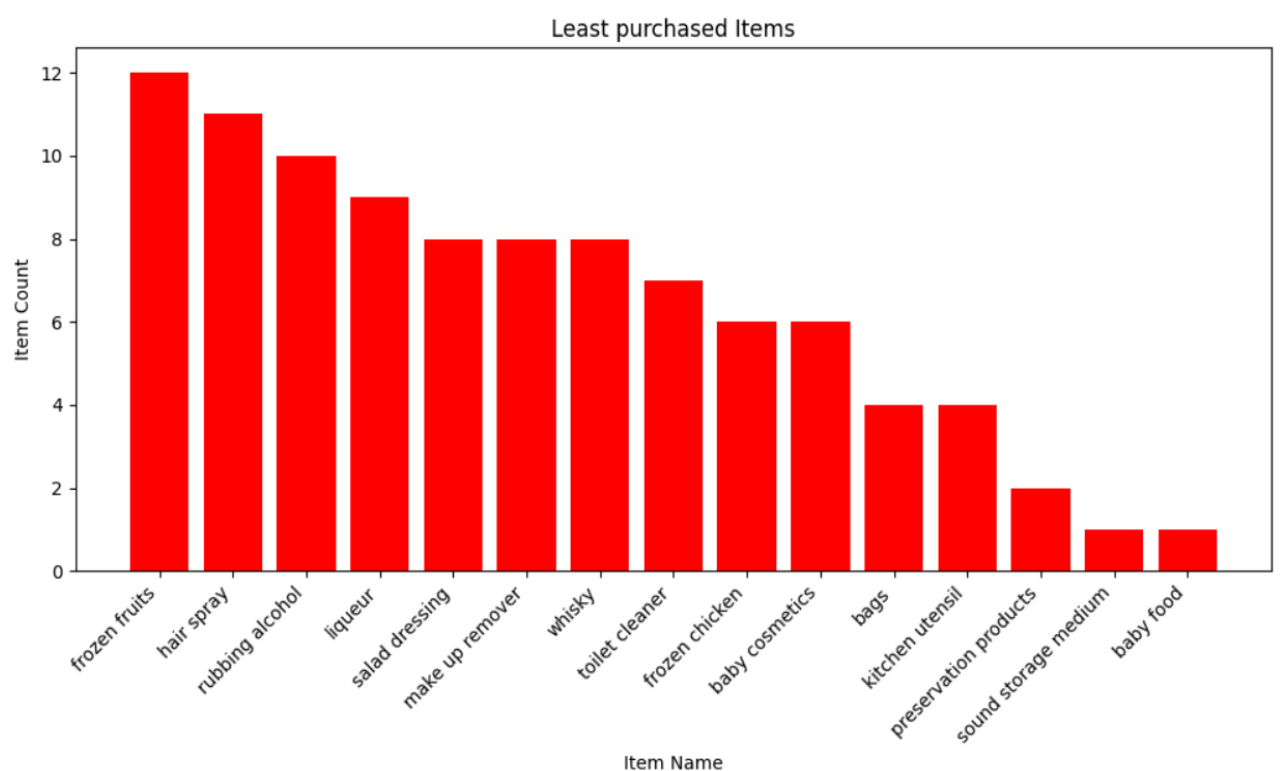
Visualizing transformed data to assess improvements in distribution.

EXPLORATORY DATA ANALYSIS IN MARKET BASKET ANALYSIS

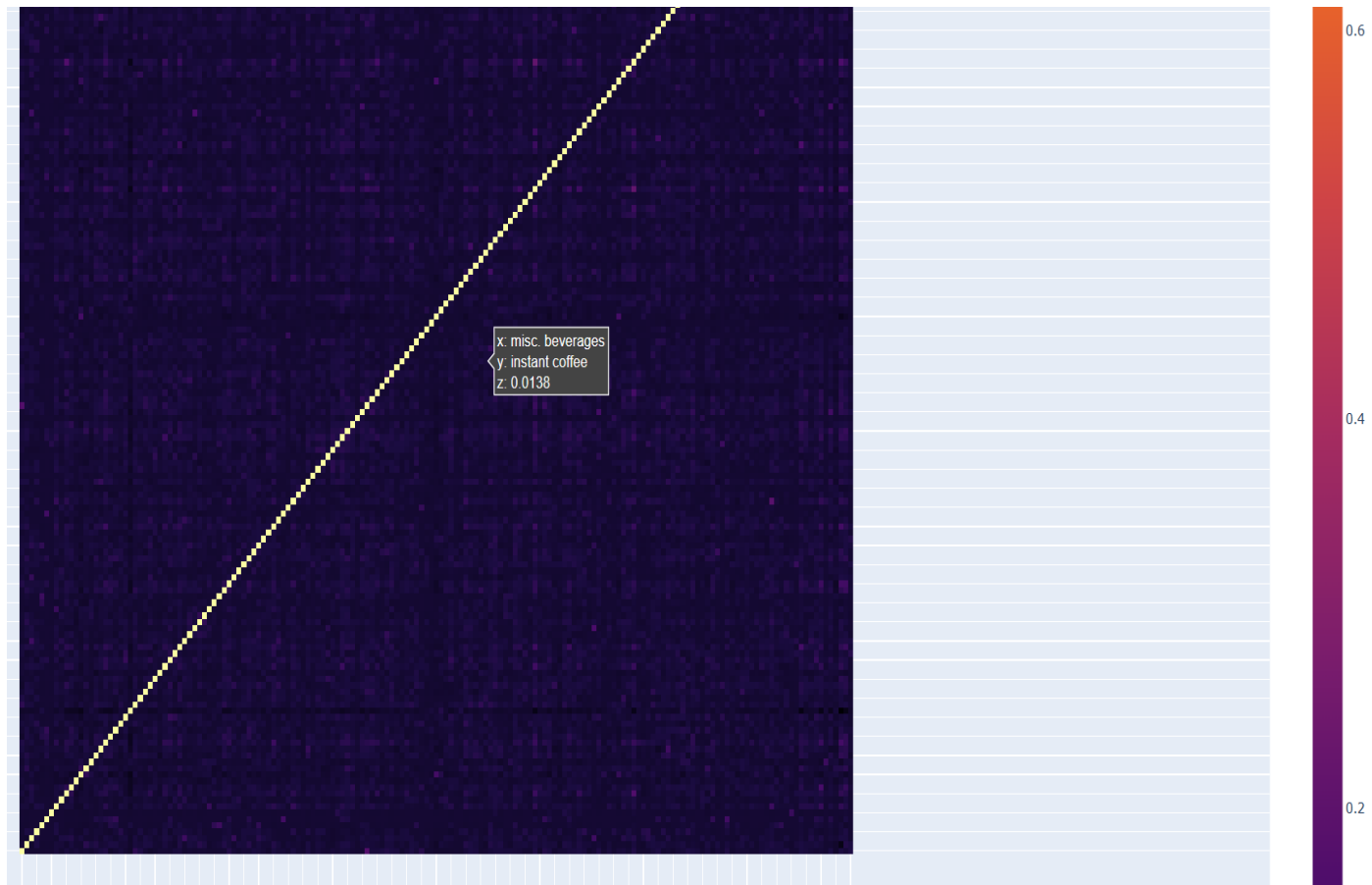
Analysing Frequently Purchased Items



Analysing Least Purchased Items



Visualizing Correlation Between Items Using Interactive Heat Map



5.5 TRAINING MODEL

Association Rule Mining

Association Rule Mining is a data mining technique focused on discovering interesting relationships and patterns within large datasets. It is commonly used in Market Basket Analysis (MBA) to uncover which items tend to be purchased together. The core idea is to find associations between items based on their co-occurrence in transactions, leading to actionable insights for businesses.

The process begins with identifying frequent item sets, which are combinations of items that appear together frequently in the dataset. Algorithms like Apriori and FP-Growth are used to efficiently generate these frequent itemsets. The Apriori algorithm, for instance, works by gradually extending itemsets from single items to larger sets, pruning those that do not meet a minimum support threshold. On the other hand, FP-Growth constructs a compact tree structure (FP-tree) to mine frequent item sets directly.

Once frequent item sets are identified, association rules are derived from them. These rules take the form "If {A}, then {B}," indicating that if a customer buys item A, they are likely to buy item B as well. These rules are evaluated using metrics such as support, confidence, and lift. Support measures the frequency of occurrence of the items together, confidence indicates the reliability of the rule, and lift compares the observed support with that expected if the items were independent.

The insights gained from Association Rule Mining are invaluable for businesses. In retail, it helps optimize product placement on shelves, design effective cross-selling strategies, and tailor promotions based on customer behaviour. In healthcare, it aids in identifying patterns in patient data for better diagnosis and treatment plans. Web usage mining utilizes association rules to recommend products or content based on user behaviour, enhancing the user experience.

Despite its challenges such as dealing with large search spaces and selecting appropriate thresholds, Association Rule Mining remains a fundamental technique in data mining. Its ability to uncover hidden relationships and patterns allows businesses to make informed decisions, improve customer satisfaction, and drive strategic growth.

Association Rule Mining

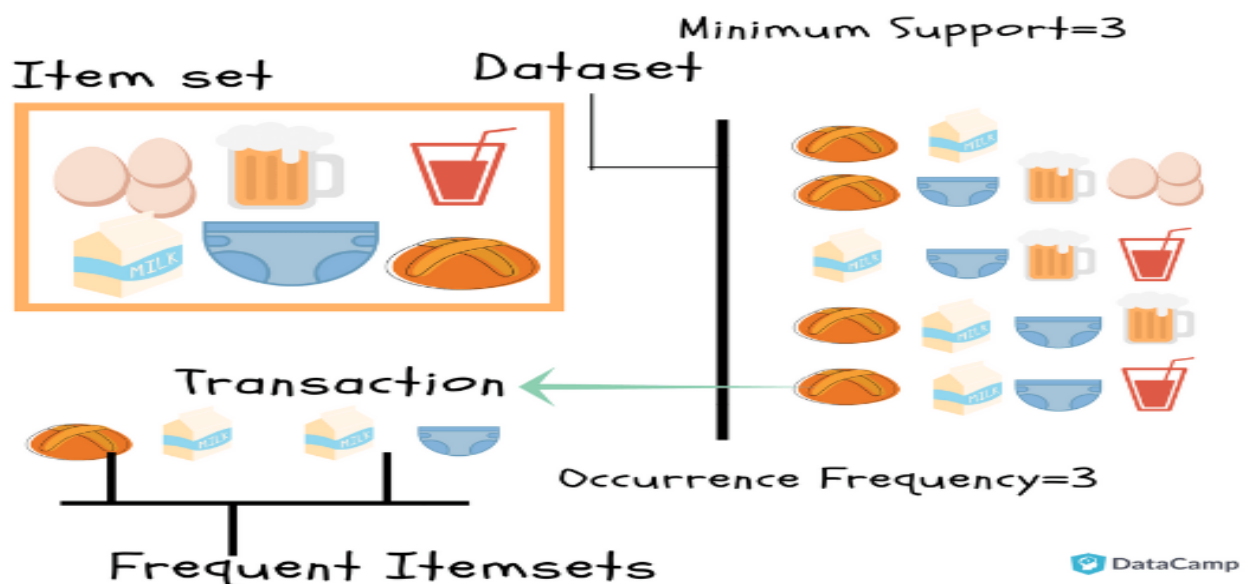
- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{\text{Diaper}\} \rightarrow \{\text{Beer}\},$
 $\{\text{Milk, Bread}\} \rightarrow \{\text{Eggs, Coke}\},$
 $\{\text{Beer, Bread}\} \rightarrow \{\text{Milk}\},$



Frequent Pattern-Growth (FP-Growth) Algorithm

The FP-Growth (Frequent Pattern Growth) algorithm is a key method in Market Basket Analysis (MBA), designed to efficiently mine frequent itemsets from transactional data. It addresses the limitations of traditional algorithms like Apriori, especially when dealing with large datasets containing numerous items. At the heart of FP-Growth is the concept of the FP-tree (Frequent Pattern tree), a compact and memory-efficient data structure. The algorithm operates in two passes over the dataset: first, constructing a header table to store item frequencies, and then building the FP-tree by inserting transactions based on the ordered frequency of items. This tree structure allows for faster mining of frequent itemsets without the need to generate candidate itemsets as in Apriori.

In practice, FP-Growth's efficiency becomes apparent during the mining of frequent itemsets. It starts by considering each item as a single-item set and recursively grows larger itemsets by merging paths in the FP-tree and creating conditional FP-trees. By doing so, it efficiently generates all frequent itemsets without the exponential complexity of generating candidate sets. Once the frequent itemsets are obtained, association rules are derived, typically using metrics like support, confidence, and lift. These rules provide insights into which items tend to be purchased together, guiding businesses in product placement, cross-selling strategies, and targeted marketing efforts.

For businesses in retail, e-commerce, and other industries reliant on transactional data analysis, FP-Growth is invaluable. It allows them to uncover meaningful patterns in customer purchasing behaviour from large datasets efficiently. This knowledge empowers businesses to make data-driven decisions, offering personalized product recommendations, optimizing inventory levels, and enhancing customer satisfaction. The FP-Growth algorithm's ability to handle large-scale datasets and efficiently mine frequent itemsets makes it a cornerstone of Market Basket Analysis, driving strategic business decisions based on discovered associations within transactional data.

Understanding FP-Growth Algorithm

These two properties inevitably make the algorithm slower. To overcome these redundant steps, a new association-rule mining algorithm was developed named Frequent Pattern Growth Algorithm. It overcomes the disadvantages of the Apriori algorithm by storing all the transactions in a Trie Data Structure. Consider the following data:

Association rules

1.Association rule

Contains antecedent and consequent

$\{\text{health}\} \rightarrow \{\text{cooking}\}$

2.Multi-antecedent rule

$\{\text{humour, travel}\} \rightarrow \{\text{language}\}$

3.Multi-consequent rule

$\{\text{biography}\} \rightarrow \{\text{history, language}\}$

4.Multi-antecedent and consequent rule

$\{\text{biography, non-fiction}\} \rightarrow \{\text{history, language}\}$

Difficulty of selecting rules

- Finding useful rules is difficult.
- Set of all possible rules is large.
- Most rules are not useful.
- Must discard most rules.

Metrics and pruning

1.Support: Support measures the frequency of occurrence of an itemset in the dataset, indicating how often the items appear together.

A metric is a measure of performance for rules.

{humour} → {poetry}

0.81

{fiction} → {travel}

0.23

Pruning is the use of metrics to discard rules.

Retain: {humour} → {poetry}

Discard: {fiction} → {travel}

The simplest metric

The support metric measures the share of transactions that contain an itemset.

number of transactions with items(s)/number of transactions

2.Confidence: Confidence is the conditional probability of B given A, representing the reliability or strength of an association rule.

When support is misleading Confidence Comes Into Picture

Milk and bread frequently purchased together.

Support: {Milk} → {Bread}

Rule is not informative for marketing.

Milk and bread are both independently popular items.

The confidence metric

Can improve over support with additional metrics.

Adding confidence provides a more complete picture.

Confidence gives us the probability we will purchase Y
given we have purchased X

.

Confidence=Support(X&Y)/Support(X)

Interpreting Confidence Metric

TID	Transaction
1	Coffee, Milk
2	Bread, Milk, Orange
3	Bread, Milk
4	Bread, Milk, Sugar
5	Bread, Jam, Milk

TID	Transaction
1	Coffee, Milk
2	Bread, Milk, Orange
3	Bread, Milk
4	Bread, Milk, Sugar
5	Bread, Jam, Milk

Support (Milk & Coffee) = 0.20

Support (Milk) = 1.00

Support (Milk & Coffee) / Support (Milk) = 0.20 / 1.00 = 0.20

The probability of purchasing both milk and coffee does not change if we condition on purchasing milk. Purchasing milk tells us nothing about purchasing coffee.

3. Lift

Lift provides another metric for evaluating the relationship between items.

Numerator: Proportion of transactions that contain X And Y

Denominator: Proportion if X and Y are assigned randomly and independently to transactions.

Lift = Support(X & Y) / Support(X) * Support(Y)

- ❖ Lift > 1 items occur in transactions together more often than we would expect based on their individual support values. This means the relationship is unlikely to be explained by random chance. This natural threshold is convenient for filtering purposes.
- ❖ Lift < 1 tells us the items are paired together less frequently in transactions than we would expect if the pairings occurred by random chance.

4.Leverage

Leverage also builds on support.

$$\text{Leverage}(X \rightarrow Y) = \text{Support}(X \& Y) - \text{Support}(X)\text{Support}(Y)$$

Leverage is similar to lift, but easier to interpret.

Leverage ranges from -1 to 1

Lift ranges from 0 to infinity.

5.Conviction

Conviction is also built using support.

More complicated and less intuitive than leverage.

$$\text{Conviction}(X \rightarrow Y) = \text{Support}(X)\text{Support}(Y^c) / \text{Support}(X \& Y^c)$$

Association And Dissociation

Association: In data mining, association refers to the relationship or connection between two or more items or variables. It signifies how often items occur together in a dataset, leading to the discovery of patterns and associations, such as "customers who buy product A also tend to buy product B."

Disassociation: Disassociation, on the other hand, represents the absence of a relationship or connection between items or variables. In the context of association rules, disassociation signifies that certain items do not occur together frequently, indicating that their presence or absence is independent of each other within the dataset.

6.Zhang's metric

Takes values between -1 and +1

Value of +1 indicates perfect association

Value of -1 indicates perfect dissociation

Comprehensive and interpretable

Constructed using support

$$\text{Zhang}(A \rightarrow B) = \text{Confidence}(A \rightarrow B) - \text{Confidence}(A^- \rightarrow B) / \max(\text{Confidence}(A \rightarrow B), \text{Confidence}(A^- \rightarrow B))$$

$$\text{Confidence} = \frac{\text{Support}(A \& B)}{\text{Support}(A)}$$

Using only support

$$\text{Zhang}(A \rightarrow B) = \frac{\text{Support}(A \& B) - \text{Support}(A) * \text{Support}(B)}{\max(\text{Support}(A) * (1 - \text{Support}(B)), \text{Support}(B) * (1 - \text{Support}(A)))}$$

Frequent Item sets

frequent_itemsets		
	support	itemsets
0	0.082766	(citrus fruit)
1	0.058566	(margarine)
2	0.017692	(semi-finished bread)
3	0.139502	(yogurt)
4	0.104931	(tropical fruit)
...
2955	0.003355	(whole milk, mayonnaise)
2956	0.002542	(root vegetables, mayonnaise)
2957	0.003559	(other vegetables, mayonnaise)
2958	0.002745	(whole milk, kitchen towels)
2959	0.002949	(jam, whole milk)
2960 rows × 2 columns		

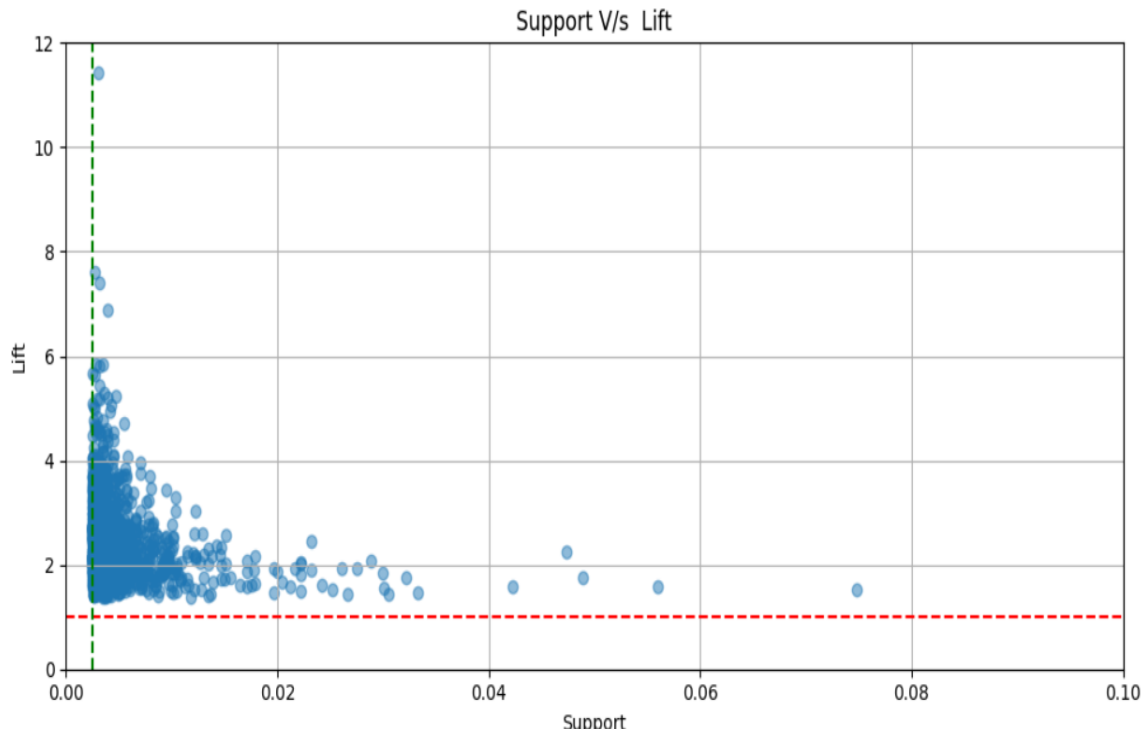
Rule Set Derived from Applying FP-Growth Algorithm

rule_set										
	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction	zhangs_metric
0	(citrus fruit)	(whole milk)	0.082766	0.255516	0.030503	0.368550	1.442377	0.009355	1.179008	0.334375
1	(citrus fruit, yogurt)	(whole milk)	0.021657	0.255516	0.010269	0.474178	1.855768	0.004736	1.415849	0.471348
2	(citrus fruit, yogurt)	(other vegetables)	0.021657	0.193493	0.007626	0.352113	1.819773	0.003435	1.244827	0.460453
3	(citrus fruit, yogurt, other vegetables)	(whole milk)	0.007626	0.255516	0.004779	0.626667	2.452553	0.002830	1.994154	0.596813
4	(citrus fruit, yogurt, whole milk)	(other vegetables)	0.010269	0.193493	0.004779	0.465347	2.404983	0.002792	1.508467	0.590258
...
1544	(roll products)	(whole milk)	0.010269	0.255516	0.004677	0.455446	1.782454	0.002053	1.367143	0.443531
1545	(mayonnaise)	(whole milk)	0.009151	0.255516	0.003355	0.366667	1.435005	0.001017	1.175501	0.305938
1546	(mayonnaise)	(other vegetables)	0.009151	0.193493	0.003559	0.388889	2.009838	0.001788	1.319739	0.507088
1547	(kitchen towels)	(whole milk)	0.005999	0.255516	0.002745	0.457627	1.790992	0.001212	1.372642	0.444316
1548	(jam)	(whole milk)	0.005389	0.255516	0.002949	0.547170	2.141431	0.001572	1.644069	0.535910

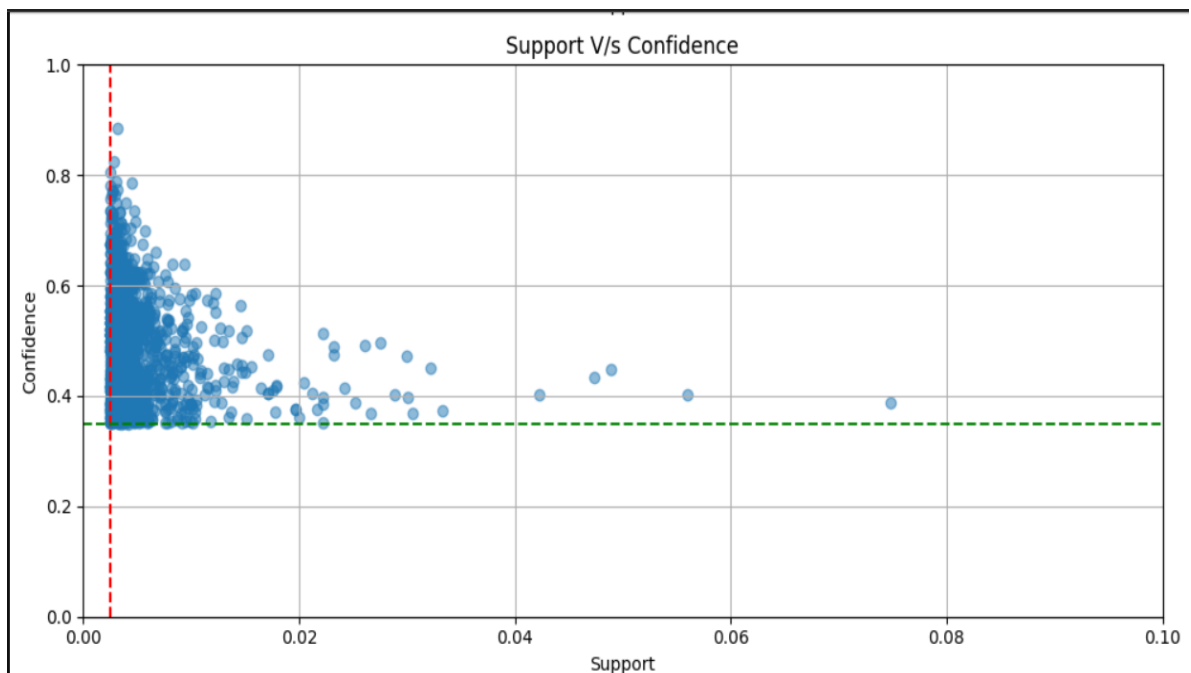
1549 rows × 10 columns

EXPLORATORY DATA ANALYSIS IN METRICS

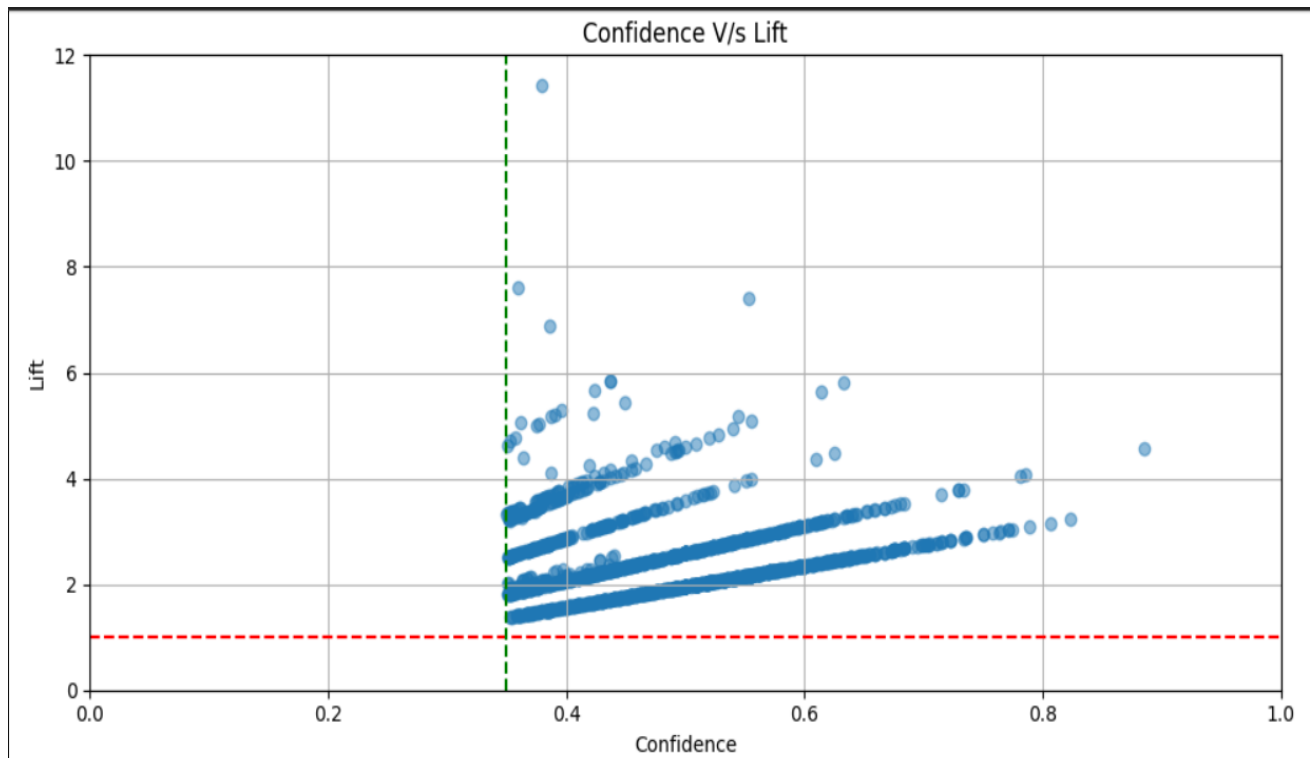
SUPPORT V/S LIFT



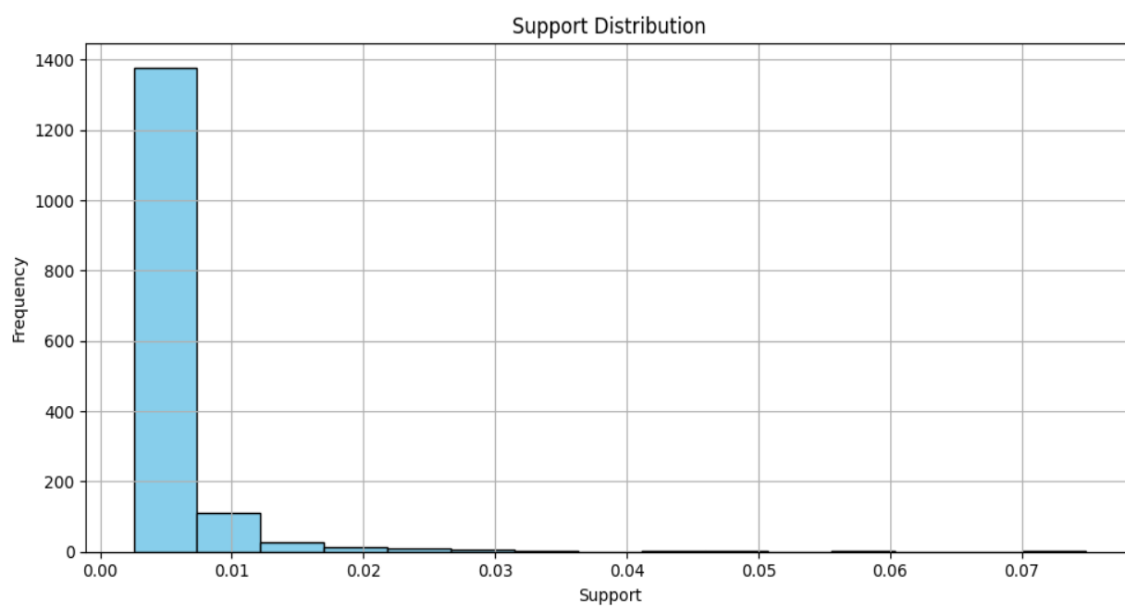
SUPPORT V/S CONFIDENCE

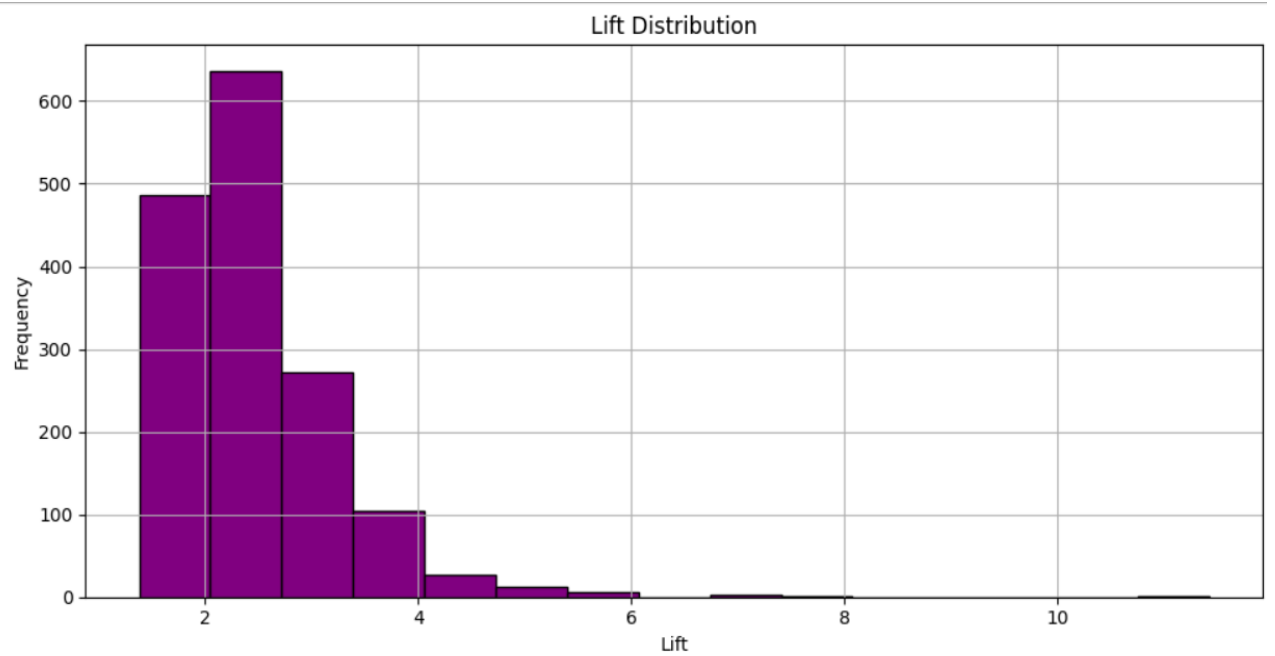
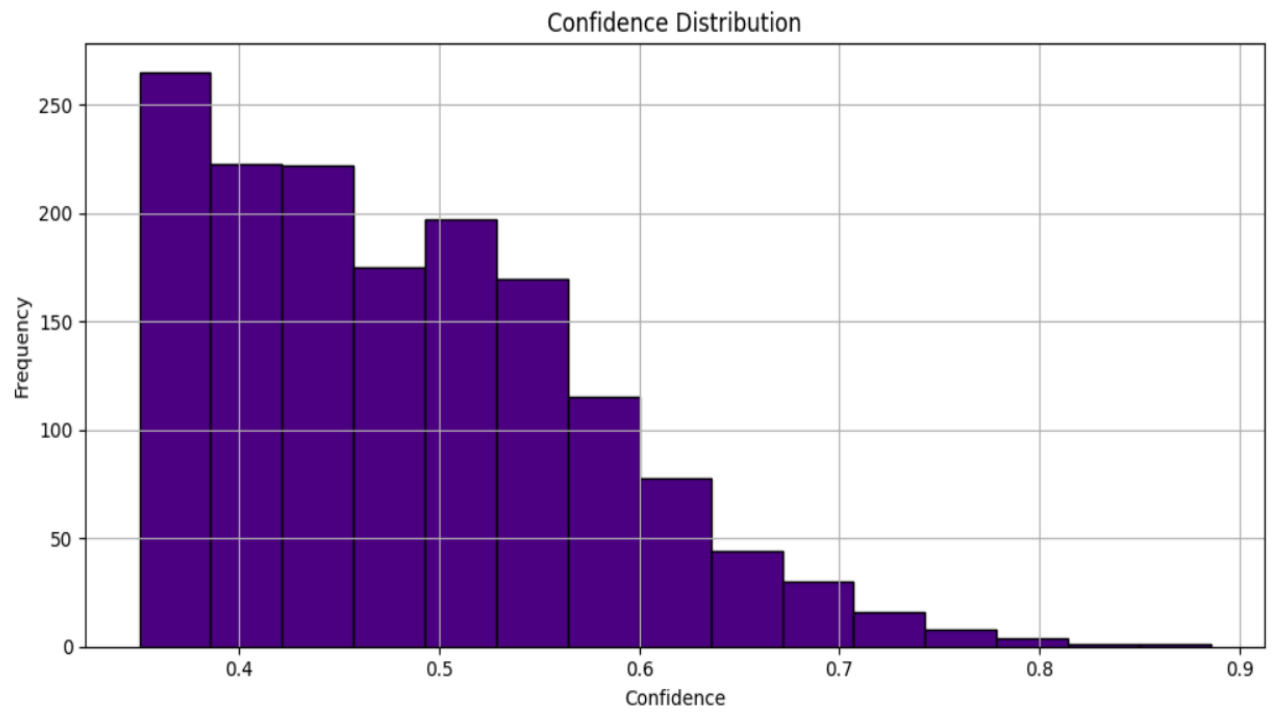


CONFIDENCE V/S LIFT



DISTRIBUTION OF METRICS





5.6 LOADING AND DUMPING MODEL

Loading and dumping models in Machine Learning using the pickle library provides a convenient way to serialize and deserialize trained models, allowing for easy storage and retrieval of model objects. When a model is trained and ready for use, it can be serialized into a binary format using `pickle.dump()`. This serialized representation captures the model's parameters, architecture, and learned patterns, essentially converting it into a file that can be saved to disk. This process is crucial for preserving the model's state, enabling it to be deployed in production environments or shared with others.

On the other end, loading a model from disk involves using `pickle.load()` to deserialize the saved binary file back into a usable model object in memory. This step is essential for reusing the trained model for making predictions on new data without the need to retrain it from scratch. Additionally, pickle allows for efficient storage of complex Python objects, such as machine learning models, making it a popular choice for saving and loading models.

However, it's essential to note that the pickle module is specific to Python and may have compatibility issues between different Python versions. It's generally recommended to use the same version of Python for both saving and loading models to ensure compatibility. Overall, loading and dumping models using pickle provides a straightforward and efficient way to persist trained models, making them portable and readily available for deployment in various applications.

5.7 DEPLOYING THE MODEL

Deploying a machine learning model using Streamlit offers a seamless and interactive way to share and showcase the model's capabilities through a web application. Streamlit is a popular Python library that simplifies the creation of web applications for data science and machine learning projects. The deployment process typically involves several steps to create an intuitive and user-friendly interface for users to interact with the model.

Firstly, the model needs to be trained and saved. Once the model is trained and ready for deployment, it can be serialized using libraries like pickle or joblib. This serialized model file is then loaded within the Streamlit application, ensuring that the model's predictions can be made in real-time.

Next, the Streamlit application is developed. This involves creating a Python script that defines the layout and functionality of the web app. Streamlit's simple yet powerful syntax allows developers to add interactive widgets such as sliders, dropdowns, and text inputs to customize user inputs. These widgets can be used to input data to the model for prediction.

The Streamlit script includes code to load the serialized model, preprocess input data (if necessary), and display the results. For example, a user might upload an image for classification, and the Streamlit app would preprocess the image, pass it through the loaded model, and display the predicted class along with a confidence score.

Additionally, Streamlit offers built-in components for displaying visualizations such as plots, charts, and tables, making it easy to showcase the model's output in a visually appealing manner. Users can interact with the app in real-time, adjusting inputs and observing how the model's predictions change accordingly.

Once the Streamlit script is finalized, the web application is launched with a single command, creating a local server that hosts the application. Users can then access the app through a web browser, providing them with an intuitive interface to interact with the model without needing to write any code.

Overall, deploying a machine learning model with Streamlit streamlines the process of creating interactive and user-friendly web applications. It allows data scientists and developers to share their models with stakeholders or the public, enabling easy exploration and understanding of the model's predictions and insights.

CHAPTER 6

DEVELOPMENT TOOLS

Features of Python:

- **Versatile Language:** Python's versatility allows for easy integration with various data analysis and machine learning libraries such as Pandas, NumPy, and scikit-learn, essential for Market Basket Analysis (MBA)
- **Rich Ecosystem:** Python's extensive library ecosystem offers specialized tools like mlxtend for association rule mining, making it ideal for implementing MBA algorithms.
- **Readable Syntax:** Python's clean and readable syntax simplifies code development and maintenance, crucial for complex data processing tasks in MBA projects.

Jupyter Lab:

- **Interactive Environment:** Jupyter Lab provides an interactive development environment, enabling data exploration, visualization, and model prototyping within a single platform.
- **Notebook Interface:** Its notebook interface allows for the creation of executable documents, integrating code, visualizations, and explanatory text, aiding in documenting and sharing MBA workflows.
- **Kernel Support:** Jupyter Lab supports multiple programming languages' kernels, including Python, R, and Julia, offering flexibility for diverse analysis needs in MBA projects.

Visual Studio Code:

- **Feature-Rich IDE:** Visual Studio Code (VS Code) offers a feature-rich integrated development environment (IDE) with robust debugging capabilities, facilitating efficient Python script development for MBA implementations
- **Extensions Ecosystem:** VS Code's vast extensions ecosystem includes plugins for Python linting, Git integration, and Jupyter Notebooks, enhancing productivity and workflow customization.
- **Version Control:** Built-in Git support in VS Code simplifies version control tasks, essential for collaborative MBA projects, ensuring team members can track changes and collaborate seamlessly.

Streamlit (A Python Framework):

- **Interactive Web Apps:** Streamlit allows for rapid development of interactive web applications with Python, enabling deployment of MBA models for user interaction and visualization.
- **Customizable Widgets:** Its easy-to-use API provides customizable widgets like sliders and dropdowns, facilitating user input and real-time model predictions in MBA applications.
- **Fast Prototyping:** Streamlit's simplicity and quick setup enable fast prototyping of MBA projects, offering a straightforward path from data exploration to interactive app deployment without complex web development.

Git and GitHub:

- **Version Control:** Git provides robust version control capabilities, allowing for tracking changes to code and data in MBA projects, essential for maintaining project history and collaborating effectively.
- **Collaboration:** GitHub, a Git repository hosting service, enables seamless collaboration among team members by providing a centralized platform for sharing code, documentation, and project milestones in MBA projects.
- **Issue Tracking:** GitHub's issue tracking features streamline project management by allowing teams to create, assign, and track tasks and bugs, enhancing the efficiency of MBA project development.

CHAPTER 7

IMPLEMENTATION

7.1 SOURCE CODE

Installing Libraries

In [2]:

```
!pip install numpy
!pip install pandas
!pip install matplotlib
!pip install mlxtend
!pip install plotly
```

Importing Libraries

In [3]:

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

Loading Dataset

In [4]:

```
basket=pd.read_csv("Data\\groceries.csv",header=None)
basket.head()

basket.replace({
    'cling film/bags': 'cling bags',
    'flower soil/fertilizer': 'flower soil',
    'fruit/vegetable juice': 'fruit juice',
    'nuts/prunes': 'nuts',
    'packaged fruit/vegetables': 'packed fruits and vegetables',
    'photo/film': 'photo film',
    'red/blush wine': 'red wine',
    'rolls/buns': 'buns',
    'whipped/sour cream': 'whipped cream'
}, inplace=True)

basket.head()

basket.shape
```

Data Preprocessing

In [7]:

```
basket=basket.astype(str)
groceries_list=np.unique(basket.values)
groceries_unique_list=groceries_list[groceries_list!='nan']
print(groceries_unique_list)
print()
print("Total Number Of Unique Items In Market Is
:",groceries_unique_list.size)

from mlxtend.preprocessing import TransactionEncoder
groceries = basket.values
encoder = TransactionEncoder()
onehot = encoder.fit_transform(groceries)
onehot_df = pd.DataFrame(onehot,
columns=encoder.columns_).drop('nan',axis=1)
onehot_df

purchased_items=onehot_df.sum().sort_values(ascending=False).reset_index()
purchased_items.columns=["Item_Name","Item_Count"]
print(purchased_items.head())
print()
total_item_count=purchased_items["Item_Count"].sum()
print("Total items sold :",total_item_count)
```

Exploratory Data Analysis

In [10]:

```
plt.figure(figsize=(10,6))
plt.bar(purchased_items['Item_Name'].head(15),purchased_items['Item_Count']
.head(15),color="green")
plt.title('Most Purchased Items')
plt.xlabel('Item Name')
plt.ylabel('Item Count')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

plt.figure(figsize=(10,6))
plt.bar(purchased_items['Item_Name'].tail(15),purchased_items['Item_Count']
.tail(15),color="red")
plt.title('Less Purchased Items')
plt.xlabel('Item Name')
plt.ylabel('Item Count')
```

```
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
corr=onehot_df.corr()
corr
```

Visualizing Correlation Among Items

In [13]:

```
import plotly.graph_objects as go
```

In [14]:

```
fig=go.Figure(data=go.Heatmap(
    z=np.around(corr.values, decimals=4),
    x=corr.index,
    y=corr.columns,
    colorscale='Inferno'
))
fig.update_layout(
    title='Correlation Heatmap ',
    xaxis=dict(title='Features', tickangle=-45),
    yaxis=dict(title='Features'),
    autosize=False,
    width=2000,
    height=2000,
    margin=dict(l=100, r=100, t=100, b=100),
)
fig.update_layout(coloraxis_colorbar=dict(title='Correlation'))
fig.write_html('correlation_heatmap_plotly.html', auto_open=True)
```

In [15]:

```
pos_corr=corr.where(corr>0,0)
```

Training Model With FP-Growth Algorithm

In [16]:

```
from mlxtend.frequent_patterns import fpgrowth, association_rules
```

In [17]:

```
frequent_itemsets = fpgrowth(onehot_df, min_support=0.0025,
    use_colnames=True)
rule_set = association_rules(frequent_itemsets, metric="confidence",
    min_threshold=0.35)
```

In [18]:

```
frequent_itemsets
```

Deriving Rule Set For Recommendations

In [19]:

```
rule_set
```

Visualizing Relationships Between Support, Confidence & Lift

In [20]:

```
fig,axs=plt.subplots(3,1,figsize=(10,15))
# Scatter Plot for Support vs Lift
axs[0].scatter(rule_set['support'],rule_set['lift'],alpha=0.5)
axs[0].set_xlabel('Support')
axs[0].set_ylabel('Lift')
axs[0].set_title('Support V/s Lift')
axs[0].axhline(y=1,c='r',ls='--')
axs[0].axvline(x=0.0025,c='g',ls='--')
axs[0].set_xlim(0, 0.1)
axs[0].set_ylim(0,12)
axs[0].grid(True)

# Scatter Plot for Support vs Confidence
axs[1].scatter(rule_set['support'], rule_set['confidence'], alpha=0.5)
axs[1].set_xlabel('Support')
axs[1].set_ylabel('Confidence')
axs[1].set_title('Support V/s Confidence')
axs[1].axvline(x=0.0025,c='r',ls='--')
axs[1].axhline(y=0.35,c='g',ls='--')
axs[1].set_xlim(0,0.1)
axs[1].set_ylim(0,1)
axs[1].grid(True)

# Scatter Plot for Confidence vs Lift
axs[2].scatter(rule_set['confidence'],rule_set['lift'], alpha=0.5)
axs[2].set_xlabel('Confidence')
axs[2].set_ylabel('Lift')
axs[2].set_title('Confidence V/s Lift')
axs[2].axhline(y=1,c='r',ls='--')
axs[2].axvline(x=0.35,c='g',ls='--')
axs[2].set_xlim(0,1)
axs[2].set_ylim(0,12)
axs[2].grid(True)

plt.tight_layout()
plt.show()
```

Distribution Of Metrics

In [21]:

```
fig, axs = plt.subplots(3, 1, figsize=(10,15))
# Support Distribution
axs[0].hist(rule_set['support'],bins=15,color='skyblue',edgecolor='black')
axs[0].set_xlabel('Support')
axs[0].set_ylabel('Frequency')
axs[0].set_title('Support Distribution')
axs[0].grid(True)

#Confidence Distribution
axs[1].hist(rule_set['confidence'],bins=15,color='indigo',edgecolor='black')
)
axs[1].set_xlabel('Confidence')
```

```

axs[1].set_ylabel('Frequency')
axs[1].set_title('Confidence Distribution')
axs[1].grid(True)

# Lift Distribution
axs[2].hist(rule_set['lift'],bins=15,color='purple',edgecolor='black')
axs[2].set_xlabel('Lift')
axs[2].set_ylabel('Frequency')
axs[2].set_title('Lift Distribution')
axs[2].grid(True)

plt.tight_layout()
plt.show()

```

Loading and Dumping the Model into Pickle File

In [22]:

```

import pickle
pickle_file_path = 'association_rules.pkl'
with open(pickle_file_path, 'wb') as f:
    pickle.dump(rule_set, f)
print("DataFrame saved to:", pickle_file_path)

```


STREAMLIT CODE IMPLEMENTATION

```
import streamlit as st
import pandas as pd
import numpy as np
import pickle
import os
from PIL import Image

# Load the data
basket = pd.read_csv("groceries.csv", header=None)
basket = basket.astype(str)

basket.rename({
    'cling film/bags': 'cling bags',
    'flower soil/fertilizer': 'flower soil',
    'fruit/vegetable juice': 'fruit juice',
    'nuts/prunes': 'nuts',
    'packaged fruit/vegetables': 'packed fruits and vegetables',
    'photo/film': 'photo film',
    'red/blush wine': 'red wine',
    'rolls/buns': 'buns',
    'whipped/sour cream': 'whipped cream'
}, inplace=True)

# Get unique items
groceries_list = np.unique(basket.values)
groceries_unique_list = groceries_list[groceries_list != 'nan']

# Background image path
```

```

background_image_path="C:\\Users\\srisa\\Desktop\\goodluck\\Designer (1).png"

# Path to the folder containing item images
item_images_folder="C:\\Users\\srisa\\Desktop\\goodluck\\Images"

# Load the association rules from pickle file
pickle_file_path = 'association_rules.pkl'
try:
    with open(pickle_file_path, 'rb') as f:
        rules_set = pickle.load(f)
except FileNotFoundError:
    st.error("Association rules pickle file not found. Please run the analysis script first.")
    rules_set = None

def main():
    # Streamlit app title with scrolling
    st.markdown("""
    <style>
        @keyframes scroll {
            0% {
                transform: translateX(100%);
            }
            100% {
                transform: translateX(-100%);
            }
        }
        .scrolling-title {
            white-space: nowrap;
            overflow: hidden;
            animation: scroll 10s linear infinite;
    """)

```

```

    }

</style>

"""', unsafe_allow_html=True)

st.markdown('<h1 class="scrolling-title">Welcome to Market Basket Analysis</h1>',
unsafe_allow_html=True)

# Adding background image
image = Image.open(background_image_path)

# Field For Selecting Items
st.sidebar.header('Select Items')
selected_items = st.sidebar.multiselect('Choose items:', groceries_unique_list)
show_background = True
if st.sidebar.button('Get Recommendations'):
    if len(selected_items) == 0:
        st.warning("Please select at least one item.")
    elif rules_set is not None:
        selected_rules = rules_set[
            rules_set['antecedents'].apply(lambda x: all(item in x for item in selected_items))
        ]

    if not selected_rules.empty:
        recommended_items = get_recommendations(selected_rules)
        if recommended_items:
            st.header('Recommended Items:')
            display_images(recommended_items)
            show_background = False
        else:
            st.write('No recommendations found for selected items.')

```

```

        else:
            st.write('No recommendations found for selected items.')
    else:
        st.error("No association rules to display. Please run the analysis script first.")

# Display the background image only if show_background is True
if show_background:
    # Increased Width Of The Background Image
    st.image(image, caption='Background', width=800)

# Function to get recommended items with image paths based on selected rules
def get_recommendations(selected_rules):
    recommended_items = {}
    for i, row in selected_rules.iterrows():
        consequents = row['consequents']
        for item in consequents:
            item_image_path = os.path.join(item_images_folder, f'{item}.jpeg')
            if os.path.exists(item_image_path):
                recommended_items[item] = item_image_path
            else:
                # If image does not exist, add a placeholder image
                recommended_items[item] = "path_to_placeholder_image.jpg"
    return recommended_items

# Function to display images
def display_images(recommended_items):
    num_columns = 3
    num_items = len(recommended_items)
    rows = num_items // num_columns + (num_items % num_columns > 0)
    columns = st.columns(num_columns)

```

```

image_width = 200
image_height = 150

item_index = 0
for row in range(rows):
    for col in range(num_columns):
        if item_index >= num_items:
            break
        item, image_path = list(recommended_items.items())[item_index]
        if image_path and os.path.exists(image_path):
            image = Image.open(image_path)
            image = image.resize((image_width, image_height))
            with columns[col]:
                st.image(image, caption=item, use_column_width=True)
        else:
            with columns[col]:
                st.write(item) # Show item name if no image available
        item_index += 1

if __name__ == "__main__":
    main()

```

CHAPTER 8

SNAP SHOTS

8.1 VARIOUS SNAPSHOTS

Select Items

Choose items:

Choose an option

Instant food products

UHT-milk

abrasive cleaner

artif. sweetener


baby cosmetics

baby food

bags

baking powder

Welcome to Market Basket Analysis

A digital interface for market basket analysis. On the left, a dark sidebar contains a 'Select Items' section with a dropdown menu and a list of product categories. The main area features a large illustration of a man in a blue cardigan pushing a shopping cart through a brightly lit supermarket aisle. The cart is filled with various items including vegetables, a milk jug, and a bag of chips. The shelves are stocked with colorful products like detergents and baby products.

Select Items

Choose items:

chicken x



Get Recommendations

Welcome to Market Basket Analysis

Recommended Items:



other vegetables



whole milk



root vegetables



citrus fruit



yogurt



buns

CHAPTER 9

FUTURE ENHANCEMENT

In the section on future enhancements, potential areas for further research and development are explored. This may include:

Advanced Algorithms: Investigate advanced data mining techniques or machine learning algorithms to improve the accuracy and efficiency of market basket analysis. This could involve exploring deep learning approaches, ensemble methods, or hybrid models.

Dynamic Analysis: Develop capabilities for dynamic and adaptive market basket analysis that can continuously learn and evolve over time. This may involve incorporating feedback mechanisms or reinforcement learning techniques.

Contextual Analysis: Enhance the analysis by incorporating contextual information such as customer demographics, purchase history, or external factors like seasonality or promotions. This could lead to more personalized and targeted recommendations.

Real-time Processing: Implement real-time or near-real-time analysis capabilities to enable timely decision-making and responsiveness to changing market conditions. This may involve leveraging streaming data processing technologies and event-driven architectures.

Visualization and Interpretability: Improve the visualization techniques and interpretability of the analysis results to make them more accessible and actionable for stakeholders. This could involve developing interactive dashboards, visual analytics tools, or storytelling techniques.

Integration with Business Processes: Integrate market basket analysis insights into existing business processes and decision-making workflows. This may involve developing APIs or connectors to seamlessly integrate analysis results with enterprise systems such as CRM, ERP, or marketing automation platforms.

By outlining potential avenues for future research and development, the section on future enhancements sets the stage for ongoing innovation and improvement in market basket analysis. It demonstrates a commitment to continuous learning and adaptation in response to evolving business needs and technological advancements.

CHAPTER 10

CONCLUSION AND REFERENCES

10.1 CONCLUSION

In the conclusion, a comprehensive summary of the entire market basket analysis project is provided. This section typically includes the following elements:

Summary of Findings: Summarize the main findings and insights obtained from the analysis. This may include significant association rules discovered, patterns identified in customer purchasing behaviour, and key metrics such as support, confidence, and lift.

Achievement of Objectives: Reflect on whether the objectives outlined in the problem statement were successfully achieved. Discuss how the analysis addressed the identified problem and contributed to meeting the project's goals.

Implications for Business: Discuss the practical implications of the findings for businesses. Highlight potential opportunities for improving marketing strategies, optimizing product placements, enhancing customer experience, or increasing revenue through cross-selling and upselling.

Limitations: Acknowledge any limitations or constraints encountered during the project. This may include data quality issues, constraints in the analysis methodology, or challenges in interpreting the results.

Recommendations: Provide recommendations for future actions based on the findings of the analysis. This could include suggestions for further research, improvements to the analysis methodology, or strategies for implementing the insights gained into business operations.

Final Thoughts: Conclude with final thoughts on the significance of the analysis and its potential impact on business decision-making. Reflect on the value added by the project and the importance of continuing to leverage data analytics for driving business success.

10.2 REFERENCES

Geeks for Geeks: https://www.geeksforgeeks.org/association-rule/?ref=header_search

Streamlit: https://youtu.be/RjiqbTLW9_E?si=gK98_UR6W2GoDzhs

IIT Kharagpur Digital Library: ndl.iitkgp.ac.in.