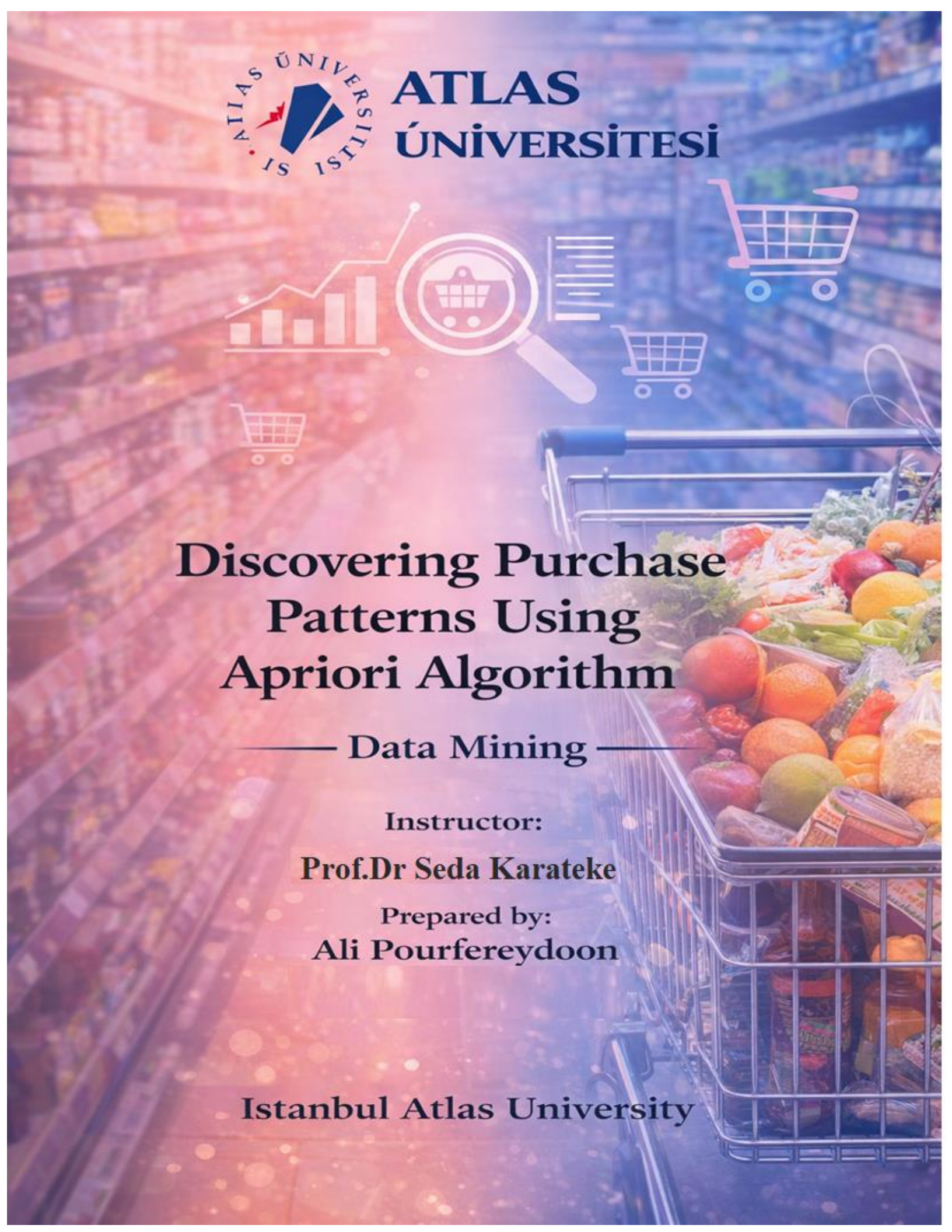


ATLAS ÜNİVERSİTESİ

Discovering Purchase Patterns Using Apriori Algorithm

Data Mining

Instructor:

Prof.Dr Seda Karateke

Prepared by:

Ali Pourfereydoon

Istanbul Atlas University

Table of Contents

List of Figures

List of Tables

CHAPTER 1

Introduction

1.1Abstract

In recent years, data mining has become an essential tool for extracting meaningful patterns and knowledge from large datasets. One important application of data mining in the retail industry is Market Basket Analysis, which aims to understand customer purchasing behavior by identifying relationships among items that are frequently purchased together. These insights support better business decisions such as product placement, cross-selling strategies, promotional planning, and inventory management.

In this project, the Groceries Market Basket Dataset obtained from Kaggle/UCI is used to perform association rule mining. The dataset contains 9,835 customer transactions and 169 unique grocery items, where each transaction represents a set of products purchased during a single shopping visit. this dataset is highly suitable for discovering frequent itemsets and association rules.

To analyze the dataset, two widely used association rule mining algorithms are applied separately: Apriori and FP-Growth. First, the Apriori algorithm is implemented to identify frequent itemsets by iteratively generating candidate sets based on the principle that all subsets of a frequent itemset must also be frequent. Although Apriori is effective in discovering associations, it can be computationally expensive when handling large datasets.

To overcome this limitation, the FP-Growth (Frequent Pattern Growth) algorithm is applied in the next stage of the project. FP-Growth improves efficiency by constructing a compact FP-tree data structure and eliminating the need for candidate generation, resulting is faster execution and lower memory usage.

The association rules generated by both algorithms are evaluated using standard metrics, including support, confidence, and lift, which measure the frequency, reliability, and strength of item relationships.

1.2 Project Goal

The primary goal of this project is to use association rule mining techniques to a real-world retail dataset in order to discover meaningful patterns in customer purchasing behavior. Using the Groceries Market Basket Dataset, the project aims to identify frequently purchased items and uncover strong associations between products that are commonly bought together.

this project has the following objectives:

- preprocess and transform the transactional grocery dataset into a suitable format for association rule mining.
- apply the Apriori algorithm to extract frequent itemsets and generate association rules based on support, confidence, and lift.
- apply the FP-Growth algorithm separately on the same dataset
- evaluate and interpret the discovered rules using standard association rule metrics.
- compare the performance and results of Apriori and FP-Growth
- demonstrate how association rule mining can support business decision-making in retail applications such as product placement, cross-selling, and inventory management.

Figure 1. project goal

CHAPTER 2

2.1What is data mining :

Data mining is the use of machine learning and statistical analysis to discover patterns and other valuable information from the large data sets . Given the evolution of machine learning (ML), data warehousing, and the growth of big data, the adoption of data mining, also known as knowledge discovery in databases (KDD), has rapidly accelerated over the last decades. However, while this technology continuously evolves to handle data at a large scale, leaders still might face challenges with scalability and automation.

The data mining techniques can be used for two main purposes. They can either describe the target data set or they can predict outcomes by using machine learning algorithms.These methods are used to organize and filter data, surfacing the most useful information, from fraud to user behaviors, bottlenecks and even security breaches. Using ML algorithms and artificial intelligence (AI) enables automation of the analysis, which can greatly speed up the process.

2.2 Benefits and challenges :

Benefits :

Discover hidden insights and trends:

Data mining takes raw data and finds order in the chaos: seeing the forest for the trees. This can result in better-informed planning across corporate functions and industries, including advertising, finance, government, healthcare, human resources (HR), manufacturing, marketing, sales and supply chain management (SCM).

Save budget: By analyzing performance data from multiple sources, bottlenecks in business processes can be identified to speed resolution and increase efficiency.

when a company analyzes performance data collected from different systems or sources, it can find bottlenecks—steps in a process that slow everything down. Once these problem areas are identified, the company can fix them faster, improve workflow, and save money (save budget) by using resources more efficiently.

Solve multiple challenges: Data mining is a versatile tool. Data from almost any source and any aspect of an organization can be analyzed to discover patterns and better ways of conducting business. Almost every department in an organization that collects and analyzes data can benefit from data mining.

Challenges

Complexity and risk:

Useful insights require valid data, plus experts with coding experience. Knowledge of data mining languages including Python, R and SQL is helpful. An insufficiently cautious approach to data mining might result in misleading or dangerous results. Some consumer data used in data mining might be personally identifiable information (PII) which should be handled carefully to avoid legal or public relations issues.

Cost: For the best results, a wide and deep collection of data sets is often needed. If new information is to be gathered by an organization, setting up a data pipeline might represent a new expense. If data needs to be purchased from an outside source, that also imposes a cost.

Uncertainty: First, a major data mining effort might be well run, but produce unclear results, with no major benefit. Or inaccurate data can lead to incorrect

insights, whether incorrect data was selected or the preprocessing was mishandled. Other risks include modeling errors or outdated data from a rapidly changing market.

Another potential problem is results might appear valid but are in fact random and not to be trusted. It's important to remember that correlation is not causation.

2.3 Data mining versus text mining versus process mining

Data mining: is the overall process of identifying patterns and extracting useful insights from big data sets. This can be used to evaluate both structured and unstructured data to identify new information and is commonly used to analyze consumer behaviors for marketing and sales teams. For example, data mining methods can be used to observe and predict behaviors, including customer churn, fraud detection, market basket analysis and more.

Text mining : also known as text data mining is a sub-field of data mining, intended to transform unstructured text into a structured format to identify meaningful patterns and generate novel insights. The unstructured data might include text from sources including social media posts, product reviews, articles, email or rich media formats such as video and audio files. Much of the publicly available data around the world is unstructured, making text mining a valuable practice.

Process mining: sits at the intersection of business process management (BPM) and data mining. Process mining provides a way to apply algorithms to event log data to identify trends, patterns and details of how processes unfold. Process mining applies data science to discover bottlenecks, and then validate and improve workflows. BPM generally collects data more informally through workshops and interviews and then uses software to document that workflow as a process map. Since the data that informs these process maps is often qualitative, process mining

brings a more quantitative approach to a process problem, detailing the actual process through event data.

2.4 How data mining works :

The data mining process involves several steps from data collection to visualization to extract valuable information from large data sets. Data mining techniques can be used to generate descriptions and predictions about a target data set.

Data scientists or business intelligence (BI) specialists describe data through their observations of patterns, associations and correlations. They also classify and cluster data through classification and regression methods, and identify outliers for use cases, such as spam detection.

data mining usually includes five main steps: **setting objectives**, **data selection**, **data preparation**, **data model building**, and **pattern mining** and **evaluating results**.

Set the business objectives: This can be the hardest part of the data mining process, and many organizations spend too little time on this important step. Even before the data is identified, extracted or cleaned, data scientists and business stakeholders can work together to define the precise business problem, which helps inform the data questions and parameters for a project. Analysts might also need to do more research to fully understand the business context.

Data selection: When the scope of the problem is defined, it is easier for data scientists to identify which set of data will help answer the pertinent questions to the business. They and the IT team can also determine where the data should be stored and secured.

Data preparation: The relevant data is gathered and cleaned to remove any noise, such as duplicates, missing values and outliers. Depending on the data set, an additional data management step might be taken to reduce the number of dimensions, as too many features can slow down any subsequent computation.

Data scientists look to retain the most important predictors to help ensure optimal accuracy within any model. Responsible data science means thinking about the model beyond the code and performance, and it is hugely impacted by the data being used and how trustworthy it is.

Model building and pattern mining: Depending on the type of analysis, data scientists might investigate any trends or interesting data relationships, such as sequential patterns, association rules or correlations. While high-frequency patterns have broader applications, sometimes the deviations in the data can be more interesting, highlighting areas of potential fraud. Predictive models can help assess future trends or outcomes. In the most sophisticated systems, predictive models can make real-time predictions for rapid responses to changing markets.

Deep learning algorithms might also be used to classify or cluster a data set depending on the available data. If the input data is labeled (such as in supervised learning), a classification model might be used to categorize data, or alternatively, a regression might be applied to predict the likelihood of a particular assignment. If the data set isn't labeled (that is, unsupervised learning), the individual data points in the training set are compared to discover underlying similarities, clustering them based on those characteristics.

Evaluation of results and implementation of knowledge: When the data is aggregated, it can then be prepared for presentation, often by using data visualization techniques, so that the results can be evaluated and interpreted. Ideally, the final

results are valid, novel, useful and understandable. When these criteria are met, decision-makers can use this knowledge to implement new strategies, achieving their intended objectives.

2.5 Data mining techniques :

Here are some of the most popular types of data mining:

2.5.1 Classification: Classes of objects are predefined, as needed by the organization, with definitions of the characteristics that the objects have in common. This enables the underlying data to be grouped for easier analysis.

For example, a consumer product company might examine its couponing strategy by reviewing past coupon redemptions together with sales data, inventory stats and any consumer data on hand to find the best future campaign strategy.

2.5.2 Clustering: Closely related to classification, clustering reports similarities, but then also provides more groupings based on differences. Preset classifications for a soap manufacturer might include detergent, bleach, laundry softener, floor cleaner and floor wax; while clustering might create groups including laundry products and floor care.

2.5.3 Decision tree: This data mining technique uses classification or regression analytics to classify or predict potential outcomes based on a set of decisions. As the decision tree name suggests, it uses a tree-like visualization to represent the potential outcomes of these decisions.

2.5.4 Association rules:

Association rules offer a powerful tool for data analysis, providing insights into patterns and relationships within large datasets. While they are a staple in market basket analysis, their application extends across various domains, offering invaluable insights into customer behaviour and beyond.

Association rule mining is a technique in data mining for discovering interesting relationships, frequent patterns, associations, or correlations, between variables in large datasets. It's widely used in various fields such as market basket analysis, web usage mining, bioinformatics, and more. The basic idea is to find rules that predict the occurrence of an item based on the occurrences of other items in the transaction.

An association rule is an if/then, rule-based method for finding relationships between variables in a data set. The strengths of relationships are measured by support and confidence. The confidence level is based on how often the if or then statements are true. The support measure is how often the related elements are shown in the data.

These methods are frequently used for market basket analysis, enabling companies to better understand the relationships between different products, such as those that are frequently purchased together. Understanding customer habits enables businesses to develop better cross-selling strategies and recommendation engines.

Frequent Pattern in Association rules: a pattern (a set of items, subsequences, substructures, etc.) that occurs frequently in a data set.

• For example, a set of items, such as milk and bread, that appear frequently together in a transaction data set is a frequent itemset.

• A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern.

• A substructure can refer to different structural forms, such as subgraphs, subtrees, or sublattices, which may be combined with itemsets or subsequences. If a substructure occurs frequently, it is called a (frequent) structured pattern.

Frequent Pattern: a pattern that occurs frequently in a data set.

A set of items that appear frequently together in a transaction data set is called as a frequent itemset.

• An example of frequent itemset mining is market basket analysis.

This process analyzes customer buying habits by finding associations between the different items that customers place in their shopping baskets.

If we think of the universe as the set of items available at the store, then each item has a Boolean variable representing the presence or absence of that item.

Each basket can then be represented by a Boolean vector of values assigned to these variables.The Boolean vectors can be analyzed for buying patterns that reflect items that are frequently associated or purchased together. these patterns can be represented in the form of association rules

Basic Concepts:

Frequent Patterns:

Table 1: frequent patterns

Tid	Items bought
10	Beer, Nuts, Diaper
20	Beer, Coffee, Diaper
30	Beer, Diaper, Eggs
40	Nuts, Eggs, Milk
50	Nuts, Coffee, Diaper, Eggs, Milk

• **itemset:** A set of one or more items k-itemset X = {x1, …, xk}

• **(absolute) support of X**: Frequency of an itemset X.

Absolute Support of {Beer} is 3

• **(relative) support of X** is the fraction of transactions that contains X (i.e., the probability that a transaction contains X). Relative Support of {Beer} is 3/5

• **An itemset X** is frequent if X's support is no less than a minsup threshold.

Association Rule : An implication expression of the form X → Y, where X and Y are itemsets

Association Rule Mining:

• Find all the rules X →Y with minimum support and minimum confidence

Support : probability that a transaction contains X U Y : P(X U Y)

• Fraction of transactions that contain both X and Y

Confidence : conditional probability that a transaction having X also contains Y :

P(Y/X) = support(X U Y) / support(X)

• Measures how often items in Y appear in transactions that contain X

Support : is an important measure because a rule that has very low support may occur simply by chance.

A low support rule may be uninteresting from a business perspective because it may not be profitable to promote items that customers seldom buy together. for these reasons, support is often used to eliminate uninteresting rules

Confidence: measures the reliability of the inference made by a rule.

For a given rule X →Y, the higher the confidence, the more likely it is for Y to be present in transactions that contain X. association analysis results should be interpreted with caution.

The inference made by an association rule does not necessarily imply causality.

Instead, it suggests a strong co-occurrence relationship between items in the antecedent and consequent of the rule

Association Rule Mining Task

Given a set of transactions T, the goal of association rule mining is to find all rules

having

support ≥ minsup threshold

confidence ≥ minconf threshold

Reducing Number of Candidates Apriori Principle

Apriori Principle: If an itemset is frequent, then all of its subsets must also be frequent.

Apriori principle holds due to the following property of the support measure

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

Support of an itemset never exceeds the support of its subsets

This is known as the anti-monotone property of support

Apriori Algorithm:

Apriori pruning principle: If there is any itemset which is infrequent, its superset should not be generated/tested!

Apriori Algorithm: Fk: frequent k-itemsets Lk: candidate k-itemsets

• Let k=1

• Generate F1 = {frequent 1-itemsets}

• Repeat until Fk is empty

Candidate Generation: Generate Lk+1 from Fk

Candidate Pruning: Prune candidate itemsets in Lk+1 containing subsets of length k that are infrequent

Support Counting: Count the support of each candidate in Lk+1 by scanning the DB

Candidate Elimination: Eliminate candidates in Lk+1 that are infrequent, leaving only

those that are frequent => Fk+1

Candidate Generation: Fk-1 x Fk-1 Method

Merge two frequent (k-1)-itemsets if their first (k-2) items are identical

• F3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE}

Merge(ABC, ABD) = ABCD

Merge(ABC, ABE) = ABCE

Merge(ABD, ABE) = ABDE

Do not merge(ABD,ACD) because they share only prefix of length 1 instead of length 2

• L4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated

Apriori Algorithm: Candidate Pruning

Let F3 = {ABC,ABD,ABE,ACD,BCD,BDE,CDE} be the set of frequent 3-itemsets

• L4 = {ABCD,ABCE,ABDE} is the set of candidate 4-itemsets generated

• Candidate pruning

Prune ABCE because ACE and BCE are infrequent

Prune ABDE because ADE is infrequent

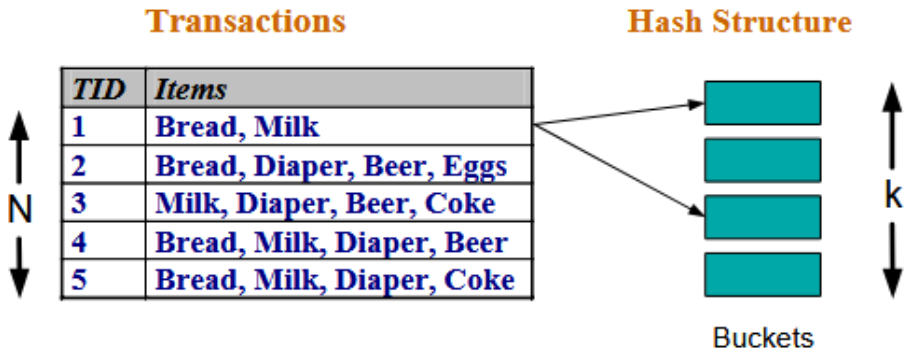
• After candidate pruning: L4 = {ABCD}

Support Counting of Candidate Itemsets

Scan the database of transactions to determine the support of each candidate itemset (Must match every candidate itemset against every transaction, which is an expensive operation)

• To reduce the number of comparisons, store the candidates in a hash structure(Instead of matching each transaction against every candidate, match it against candidates contained in the hashed buckets)

Figure 2: transactions



Algorithm: Apriori. Find frequent itemsets using an iterative level-wise approach based on candidate generation.

Input:

- D, a database of transactions;
- *min_sup*, the minimum support count threshold.

Output: L, frequent itemsets in D.

Method:

```
L1 = find_frequent_1-itemsets(D);
for (k = 2; Lk−1 ≠ φ; k++) {
    Ck = apriori_gen(Lk−1);
    for each transaction t ∈ D { // scan D for counts
        Ct = subset(Ck, t); // get the subsets of t that are candidates
        for each candidate c ∈ Ct
            c.count++;
    }
    Lk = {c ∈ Ck | c.count ≥ min_sup}
}
return L = ∪k Lk;
```

```
procedure apriori_gen(L_{k-1}:frequent (k-1)-itemsets)
    for each itemset l_1 ∈ L_{k-1}
        for each itemset l_2 ∈ L_{k-1}
            if (l_1[1] = l_2[1]) ∧ (l_1[2] = l_2[2])
                ∧... ∧ (l_1[k-2] = l_2[k-2]) ∧ (l_1[k-1] < l_2[k-1]) then {
                c = l_1 ⋈ l_2; // join step: generate candidates
                if has_infrequent_subset(c, L_{k-1}) then
                    delete c; // prune step: remove unfruitful candidate
                else add c to C_k;
            }
    return C_k;

procedure has_infrequent_subset(c: candidate k-itemset;
        L_{k-1}: frequent (k-1)-itemsets); // use prior knowledge
    for each (k-1)-subset s of c
        if s ∉ L_{k-1} then
            return TRUE;
    return FALSE;
```

2.6 Data mining use cases

Data mining techniques are widely adopted by business intelligence and data analytics teams, helping them extract knowledge for their organization and industry.

Some data mining use cases include:

Anomaly detection

While frequently occurring patterns in data can provide teams with valuable insights, observing data anomalies is also beneficial, assisting organizations with fraud detection, network intrusions and product defects. While this is a well-known use case within banking and other financial institutions, SaaS-based companies have also started to adopt these practices to eliminate fake user accounts from their data sets. Anomaly detection can also be an opportunity to find new and novel strategies or target markets that have been overlooked in the past.

Assess risk

Organizations can more accurately locate and determine the scale of risk with data mining. Patterns and anomalies can be uncovered in the cybersecurity, finance and legal fields to pinpoint oversights or threats.

Focus on target markets

By searching across multiple databases to find close relationships, data mining can accurately connect behaviors and customer backgrounds with sales of specific items. This can enable more targeted campaigns to help boost sales.

Improve customer service

Customer issues can be discovered and fixed sooner if the full sum of customer actions (on-site, online, over mobile apps or on a telephone) can be reviewed with data mining. Customer service agents can have access to more complete and insightful information on the customers they serve.

Increase equipment uptime

Operational data can be mined from industrial equipment that can help predict future performance and downtime, and enable the planning of protective maintenance.

Operational optimization

Process mining uses data mining techniques to reduce costs across operational functions, enabling organizations to run more efficiently. This practice can help to identify costly bottlenecks and improve decision-making for business leaders.

Industry use cases :

Customer service

Data mining can create a richer data source for customer service by helping to determine which factors most please the customers and what factors cause friction or dissatisfaction.

Education

Educational institutions have started to collect data to understand their student populations and which environments are conducive to success. With courses often using online platforms, they can use various dimensions and metrics to observe and evaluate performance, such as keystrokes, student profiles, classes attended and time spent.

Finance

When researching risk, financial institutions and banks often want to cast a wide net, to capture any factors that might negatively impact cash flow and retrieval. Data mining tools can be useful in finding and weighing a combination of factors that indicate a good or bad risk.

Healthcare

Data mining is a useful tool for the diagnosis of medical conditions (including the reading of scans and images) and then assists in the suggestion of beneficial treatments.

Human resources

Organizations can gain new insights into employee performance and satisfaction by analyzing multiple factors and finding patterns. Data can include start date,

promotions, salary, training, peer performance, work delivery, use of benefits and travel.

Manufacturing

From raw materials to final delivery, all aspects of the manufacturing process can be analyzed to improve performance. What is the cost of materials and are there options? How efficient is production? Where are the bottlenecks? What are the quality issues and where do they arise, both internally and with customers.

Retail

By mining customer data and actions, retailers can identify the most productive campaigns, pricing, promotions, special product offers and successful cross-sells and up-sells.

Sales and marketing

Companies collect massive amounts of data about their customers and prospects. By observing consumer demographics, media responses and customer behavior, companies can use data to optimize their marketing campaigns, improving segmentation and targeting and customer loyalty programs, all helping to yield higher return on investment (ROI) on marketing efforts. Predictive analyses can also help teams set expectations with their stakeholders, providing yield estimates for any increases or decreases in marketing investment.

Social media

Analysis of user data can help uncover new editorial opportunities or new sources of advertising revenue for specific target audiences.

Supply chain management (SCM)

Using data mining, product managers can better predict demand, gear up production, adjust providers or adapt marketing efforts. Supply chain managers can better plan shipping and warehousing.

Chapter 3

methodology

3.1 In the first step Apriori Algorithm was used to implement this project :

Apriori Algorithm is a basic method used in data analysis to find groups of items that often appear together in large sets of data. It helps to discover useful patterns or rules about how items are related which is particularly valuable in market basket analysis. like in a grocery store if many customers buy bread and butter together, the store can use this information to place these items closer or create special offers. This helps the store sell more and make customers happy.

How the Apriori Algorithm Works :

The Apriori Algorithm operates through a systematic process that involves several key steps:

3.1.1. Identifying Frequent Item-Sets

- The Apriori algorithm starts by looking through all the data to count how many times each single item appears. These single items are called 1-Item-Sets.
- Next it uses a rule called minimum support this is a number that tells us how often an item or group of items needs to appear to be important. If an item appears often enough meaning its count is above this minimum support it is called a frequent Item-Set.

3.1.2. Creating Possible Item Group

- After finding the single items that appear often enough (frequent 1-item groups) the algorithm combines them to create pairs of items (2-item groups).

Then it checks which pairs are frequent by seeing if they appear enough times in the data.

- This process keeps going step by step making groups of 3 items, then 4 items and so on. The algorithm stops when it can't find any bigger groups that happen often enough.

3.1.3. Removing Infrequent Item Groups

- The Apriori algorithm uses a helpful rule to save time. This rule says: if a group of items does not appear often enough then any larger group that includes these items will also not appear often.
- Because of this, the algorithm does not check those larger groups. This way it avoids wasting time looking at groups that won't be important make the whole process faster.

3.1.4. Generating Association Rules

- The algorithm makes rules to show how items are related.
- It checks these rules using support, confidence and lift to find the strongest ones.

3.2 Key Metrics of Apriori Algorithm

- **Support**: This metric measures how frequently an item appears in the dataset relative to the total number of transactions. A higher support indicates a more significant presence of the Item-Set in the dataset. Support tells us how often a particular item or combination of items appears in all the transactions like Bread is bought in 20% of all transactions.

- **Confidence**: Confidence assesses the likelihood that an item Y is purchased when item X is purchased. It provides insight into the strength of the association between two items. Confidence tells us how often items go together i.e If bread is bought, butter is bought 75% of the time.
- **Lift**: Lift evaluates how much more likely two items are to be purchased together compared to being purchased independently. A lift greater than 1 suggests a strong positive association. Lift shows how strong the connection is between items. Like Bread and butter are much more likely to be bought together than by chance.

Example

Let's understand the concept of apriori Algorithm with the help of an example. Consider the following dataset and we will find frequent Item-Sets and generate association rules for them:

Table 2: transactions with generate itemsets

Transaction ID	Items Bought
T1	Bread, Butter, Milk
T2	Bread, Butter
T3	Bread, Milk
T4	Butter, Milk
T5	Bread, Milk

Transactions of a Grocery Shop

Step 1 : Setting the parameters

- **Minimum Support Threshold:** 50% (item must appear in at least 3/5 transactions). This threshold is formulated from this formula:

$$\textbf{Support}(A) = \frac{\text{Number of transactions containing itemset } A}{\text{Total number of transactions}}$$

- **Minimum Confidence Threshold:** 70% (You can change the value of parameters as per the use case and problem statement). This threshold is formulated from this formula:

$$\textbf{Confidence}(\boldsymbol{X \rightarrow Y}) = \frac{\text{Support}(X \cup Y)}{\text{Support}(X)}$$

Step 2: Find Frequent 1-Item-Sets

Let's count how many transactions include each item in the dataset (calculating the frequency of each item).

Table 3: items and support count

Item	Support Count	Support %
Bread	4	80%
Butter	3	60%
Milk	4	80%

All items have support ≥ 50%, so they qualify as frequent 1-Item-Sets. if any item has support < 50%, It will be omitted out from the frequent 1- Item-Sets.

Step 3: Generate Candidate 2-Item-Sets

Combine the frequent 1-Item-Sets into pairs and calculate their support. For this use case we will get 3 item pairs (bread,butter) , (bread,ilk) and (butter,milk) and will calculate the support similar to step 2

Table 4: items and support count

Item Pair	Support Count	Support %
Bread, Butter	2	40%
Bread, Milk	3	60%
Butter, Milk	2	40%

Candidate 2-Itemsets

Frequent 2-Item-Sets: {Bread, Milk} meet the 50% threshold but {Butter, Milk} and {Bread ,Butter} doesn't meet the threshold, so will be committed out.

Step 4: Generate Candidate 3-Item-Sets

Combine the frequent 2-Item-Sets into groups of 3 and calculate their support. for the triplet we have only got one case i.e {bread,butter,milk} and we will calculate the support.

Table 5: items and support count

Item Triplet	Support Count	Support %
Bread, Butter, Milk	1	40%

Candidate 3-Itemsets

Since this does not meet the 50% threshold, there are no frequent 3-Item-Sets.

Step 5: Generate Association Rules

Now we generate rules from the frequent Item-Sets and calculate confidence.

Rule 1: If Bread → Butter (if customer buys bread, the customer will buy butter also)

- Support of {Bread, Butter} = 2.
- Support of {Bread} = 4.
- Confidence = 2/4 = 50% (Failed threshold).

Rule 2: If Butter → Bread (if customer buys butter, the customer will buy bread also)

- Support of {Bread, Butter} = 2.

- Support of {Butter} = 3.
- Confidence = 2/3 = 66.67% (Passes threshold).

Rule 3: If Bread → Milk (if customer buys bread, the customer will buy milk also)

- Support of {Bread, Milk} = 3.
- Support of {Bread} = 4.
- Confidence = 3/4 = 75% (Passes threshold).

The Apriori Algorithm, as demonstrated in the bread-butter example, is widely used in modern startups like Zomato, Swiggy and other food delivery platforms. These companies use it to perform market basket analysis which helps them identify customer behaviour patterns and optimise recommendations.

3.3 Applications of Apriori Algorithm :

Below are some applications of Apriori algorithm used in today's companies and startups

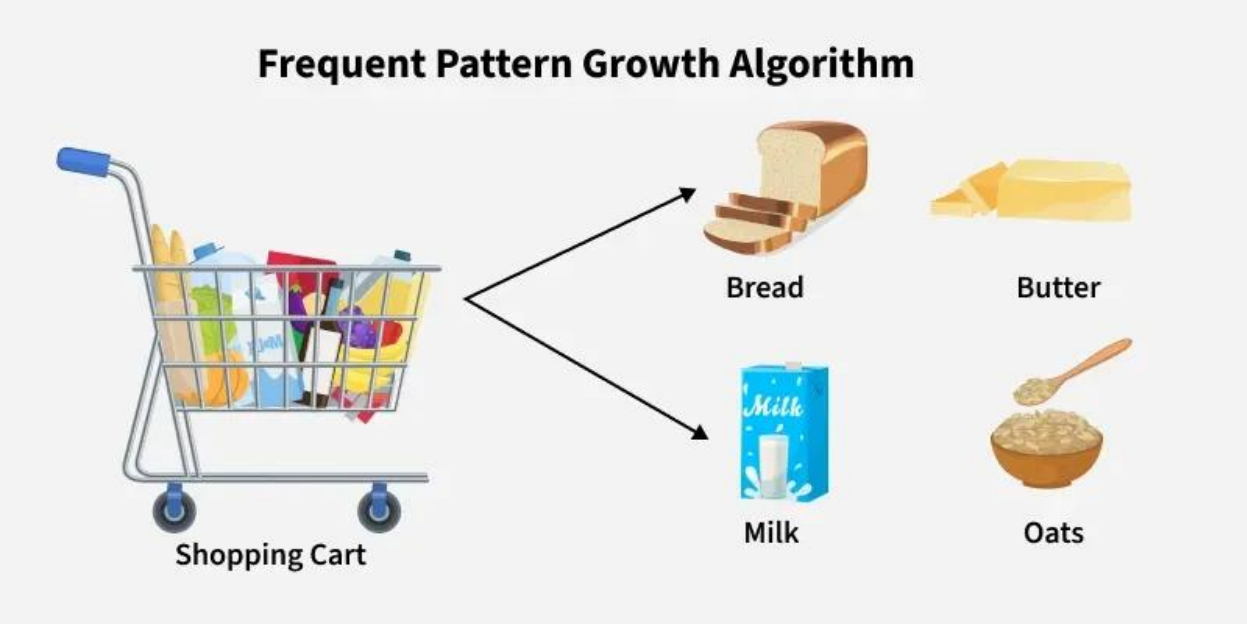
1. **E-commerce:** Used to recommend products that are often bought together like laptop + laptop bag, increasing sales.
2. **Food Delivery Services:** Identifies popular combos such as burger + fries to offer combo deals to customers.
3. **Streaming Services:** Recommends related movies or shows based on what users often watch together like action + superhero movies.
4. **Financial Services:** Analyzes spending habits to suggest personalised offers such as credit card deals based on frequent purchases.

5. **Travel & Hospitality:** Creates travel packages like flight + hotel by finding commonly purchased services together.
6. **Health & Fitness:** Suggests workout plans or supplements based on users past activities like protein shakes + workouts.

3.4 In the second step , Frequent Pattern Growth Algorithm was used to implement this project :

The FP-Growth (Frequent Pattern Growth) algorithm efficiently mines frequent itemsets from large transactional datasets. Unlike the Apriori algorithm which suffers from high computational cost due to candidate generation and multiple database scans. FP-Growth avoids these inefficiencies by compressing the data into an FP-Tree (Frequent Pattern Tree) and extracts patterns directly from it.

Figure 3: frequent pattern growth algorithm



3.4.1 How FP-Growth Works :

Here's how it works in simple terms:

1. **Data Compression:** First FP-Growth compresses the dataset into a smaller structure called the Frequent Pattern Tree (FP-Tree). This tree stores information about item sets (collections of items) and their frequencies without need to generate candidate sets like Apriori does.
2. **Mining the Tree:** The algorithm then examines this tree to identify patterns that appear frequently based on a minimum support threshold. It does this by breaking the tree down into smaller "conditional" trees for each item making the process more efficient.
3. **Generating Patterns:** Once the tree is built and analyzed the algorithm generates the frequent patterns (itemsets) and the rules that describe relationships between items.

Imagine you're organizing a party and want to know popular food combinations without asking every guest repeatedly.

1. List food items each guest brought transactions.
2. Count items and remove infrequent ones filter by support.
3. Group items in order of popularity and create a tree where paths represent common combinations.
4. Instead of repeatedly asking guests you explore this tree to discover patterns. For example, you might find that pizza and pasta often come together or that cake and pasta are also a common pair.

This is exactly how FP-Growth finds frequent patterns efficiently.

Working of FP- Growth Algorithm

Problem Statement: Consider a small grocery store transaction dataset. Each entry shows the set of items purchased together by a customer:

Table 6: items and sorted items

Transaction ID	Sorted Items by Frequency
T1	Bread, Milk, Butter
T2	Bread , Butter
T3	Bread, Milk
T4	Milk , Butter
T5	Bread

We apply the FP-Growth algorithm to identify frequent itemsets (groups of items frequently bought together), using a minimum support count of 2.

Step 1: Compute Item Frequencies

Scan the entire dataset one time to determine how often each item appears.

Table 7: items and support count

Item	Support Count
Milk	3
Bread	4
Butter	3

All items meet the minimum support threshold (≥ 2), so none are removed.

Step 2: Order Items in Each Transaction by Frequency

Next, sort every transaction in descending order based on how frequently the items appear overall.

Table 8: items and support count

Transaction ID	Sorted Items by Frequency
T1	Bread, Milk, Butter
T2	Bread , Butter
T3	Bread, Milk
T4	Milk , Butter
T5	Bread

Sorting by frequency helps group common items together, allowing the FP-tree to be compressed efficiently.

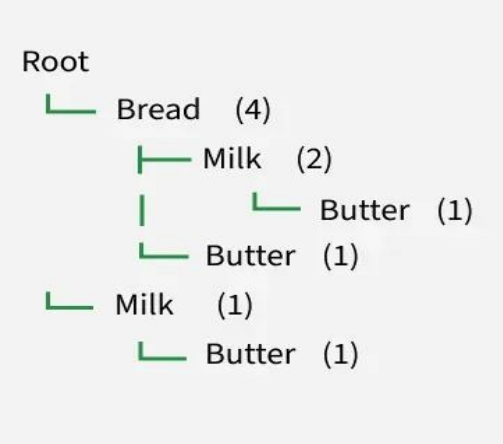
Step 3: Construct the FP-Tree

Insert the sorted transactions one by one into the FP-tree, merging any shared prefixes.

FP-Tree Insertions:

- From T1: Bread → Milk → Butter
- From T2: Bread → Butter
- From T3: Bread → Milk
- From T4: Milk → Butter (new branch since it doesn't begin with Bread)
- From T5: Bread

Visual FP-Tree:



Each node shows: Item (Count)

Step 4: Determine Conditional Pattern Bases

A conditional pattern base contains all prefix paths leading to a specific item. Let's examine the paths ending with Butter.

Paths that end with Butter:

- Bread → Milk → Butter (1 occurrence)
- Bread → Butter (1 occurrence)
- Milk → Butter (1 occurrence)

Thus, the conditional pattern base for Butter is:

[(Bread, Milk): 1, (Bread): 1, (Milk): 1]

Step 5: Build Conditional FP-Trees

Using the conditional pattern base, construct a smaller FP-tree for each item to identify frequent patterns involving that item. Butter's conditional FP-tree input:

- (Bread, Milk): 1
- (Bread): 1

- (Milk): 1

Count all items:

- Bread: 2
- Milk: 2

Since both meet the support threshold (≥2), we can now generate frequent patterns:

- {Butter, Bread}
- {Butter, Milk}
- {Butter, Bread, Milk}

Repeat the process for Milk and Bread as needed.

Step 6: Extract All Frequent Itemsets

From the FP-tree and conditional trees, we get these frequent itemsets:

- {Bread}
- {Milk}
- {Butter}
- {Bread, Milk}
- {Bread, Butter}
- {Milk, Butter}
- {Bread, Milk, Butter}

All of these appear at least 2 times in the transactions.

Summary Table

Table 9: Summary Table

Frequent Itemset	Support Count
{Bread}	4
{Milk}	3
{Butter}	3
{Bread, Milk}	2
{Bread , Butter}	2
{Milk , Butter}	2
{Bread , Milk , Butter}	1 ✕ Not included (support < 2)

Why FP-Growth is Efficient

- It scans the database only twice.
- It avoids generating all combinations of items.
- It stores data in a compact tree, reducing redundancy.
- It uses conditional trees to mine deeper patterns efficiently.

3.2.2 FP-Growth vs. Apriori Algorithm :

Let’s compare FP-Growth and Apriori algorithms:

Table 10: FP-Growth vs. Apriori Algorithm

Feature	FP-Growth	Apriori
Candidate generation	Not required	Needed
Data scans	2	Several
Speed	Efficient for large datasets	Slower because of repeated scans
Memory use	Higher (requires a tree structure)	Lower
Complexity	More difficult to implement	Simpler to grasp and build

3.4.3 Applications :

Let's see the various applications

- **Market Basket Analysis:** Identifying items that are frequently purchased together.
- **Recommendation Systems:** Suggesting products based on frequent item patterns.
- **Customer Behavior Analysis:** Understanding buying habits in retail and e-commerce.
- **Web Usage Mining:** Finding common browsing or click paths among users.

- **Text Mining:** Detecting frequently occurring words or phrases in documents.

3.4.4 Advantages :

- **No Candidate Generation:** FP-Growth avoids generating candidate sets, improving efficiency.
- **Minimal Data Scans:** Only two database scans are required.
- **High Performance on Large Data:** Works very well for big and dense datasets.
- **Data Compression:** Uses an FP-tree to store transactions compactly.
- **Scalable and Fast:** Capable of mining complex and large frequent patterns quickly.

3.4.5 Limitations :

- **High Memory Usage:** FP-tree can become large, consuming significant memory.
- **Complex Implementation:** Harder to understand and implement than Apriori.
- **Heavy Conditional Mining:** Building conditional FP-trees can be computationally costly.
- **Poor for Diverse Transactions:** Not ideal when transactions have many unique items.
- **Not Incremental:** Cannot easily update hence tree must be rebuilt when data changes.

Chapter 4

DATASET

Groceries Market Basket Dataset :

4.1 Dataset Description

The Groceries Market Basket Dataset is a widely used benchmark dataset for market basket analysis and association rule mining. It represents real-world grocery shopping transactions and is suitable for discovering purchasing patterns among items.

Dataset Overview

- Total number of transactions: 9,835
- Total number of unique items: 169
- Domain: Retail / Grocery shopping
- Data type: Transactional data

Each transaction corresponds to a single customer visit to a grocery store and contains a list of items purchased together during that visit. The dataset is structured to support association rule mining techniques such as Apriori and FP-Growth.

4.2 Purpose and Suitability :

The dataset is well suited for data mining tasks, particularly market basket analysis, because:

- It contains multiple variables (items) per transaction

- It reflects real purchasing behavior
- It allows identification of frequent itemsets and association rules

Using this dataset, meaningful insights such as frequently co-purchased products can be extracted using support, confidence, and lift measures.

4.3 Source and Acknowledgement :

The dataset was obtained from Kaggle and is used to this assignment on Association Rule Mining using the Apriori Algorithm and FP-Growth algorithm

The dataset is provided in a CSV format where:

- Each row represents one transaction
- Items within a transaction are separated by commas
- No numerical values are present; all attributes are categorical

This structure makes the dataset directly usable for association rule mining without requiring complex restructuring.

4.4 Data Preprocessing :

Before applying data mining algorithms, the following preprocessing steps were performed:

1. **Transaction Parsing**
    - The CSV file was read transaction by transaction.
    - Each transaction was stored as a list of items.

2. **Item Encoding**
    - A mapping was created from each unique item to a unique integer value.
    - This transformation reduced memory usage and computational complexity.
3. **Reverse Mapping**
    - A reverse mapping from integers back to item names was maintained.
    - This allowed the final results (association rules) to be presented in a human-readable format.
4. **Data Transformation**
    - The processed data was transformed into a suitable format for applying the Apriori and FP-Growth algorithms.

the Groceries Market Basket Dataset is a clean, well-structured, and widely accepted dataset for association rule mining. Its transactional nature and moderate size make it ideal for educational and experimental analysis using Apriori and FP-Growth algorithms.

Table 11: Groceries Market Basket Dataset

Attribute	Description
Dataset name	Groceries Market Basket Dataset
Domain	Retail / Grocery Shopping
Total number of transactions	9,835
Total number of unique items	169
Data type	Transactional data
File format	CSV
Nature of attributes	Categorical (item names)
Missing values	None
Average items per transaction	~4 items
Suitable algorithms	Apriori, FP-Growth
Primary application	Market Basket Analysis

Chapter 5

System Implementation

5.1 Apriori Algorithm Implementation :

In this project, at first the Apriori algorithm was applied to the Groceries Market Basket Dataset to discover frequent itemsets and generate association rules. The dataset consists of transactional data, where each transaction represents a set of grocery items purchased together, making it suitable for association rule mining using Apriori.

Prior to applying the algorithm, the dataset was preprocessed and transformed into a binary one-hot encoded format, where each row corresponds to a transaction and each column represents an item. This transformation is required by the Apriori algorithm to calculate itemset frequencies efficiently.

The Apriori algorithm was implemented using the Mlxtend library in Python. A minimum support threshold was defined to identify frequent itemsets that appear in a sufficient number of transactions. Based on these frequent itemsets, association rules were generated using confidence as the primary evaluation metric.

The generated rules were further analyzed using support, confidence, and lift measures to identify strong and meaningful relationships between items. The Apriori algorithm served as the baseline method in this study and its results were later compared with those obtained from the FP-Growth algorithm to evaluate performance and efficiency.

5.1.1 Data Preprocessing:

The Groceries Market Basket Dataset contains transactions with a variable number

of items per basket. Therefore, the dataset was read line by line as raw text. Each transaction was converted into a list of items. One-hot encoding was then applied using the TransactionEncoder to transform the transactional data into a binary format suitable for the Apriori algorithm.

STEP 0: Mount Google Drive

Figure 4: import libraries

```
from google.colab import drive
drive.mount('/content/drive')
```

`from google.colab import drive`

This line imports the drive module provided by Google Colab.the module allows Colab notebooks to access files stored in the user's Google Drive.

`drive.mount('/content/drive')`

This command mounts the user's Google Drive to the Colab virtual machine at the directory /content/drive.

After executing this command:

- Google Colab ask to authorize access to Google account
- Once authorized, all Google Drive files become accessible inside Colab

google drive for store database :

- Google Colab sessions are temporary and do not permanently store uploaded files
- Mounting Google Drive allows large datasets to be loaded faster and persistently
- It avoids repeated uploads of the dataset every time the notebook restarts

STEP 1: Load Groceries Market Basket Dataset

Figure 5: Load Groceries Market Basket Dataset

```
transactions = []

with open("/content/drive/MyDrive/groceries.csv", "r") as file:
    for line in file:
        items = line.strip().split(",")
        transactions.append(items)

print("Total number of transactions:", len(transactions))
print("First 5 transactions:")
print(transactions[:5])
```

This section performs the initial data loading and structuring required for market basket analysis. It converts raw CSV data into a transactional format, which is a prerequisite for applying the Apriori algorithm.

`transactions = []`

This line initializes an empty list named transactions. It will be used to store all grocery transactions, where each transaction is represented as a list of purchased items.

`with open("/content/drive/MyDrive/groceries.csv", "r") as file:`

This statement opens the groceries.csv file in read mode from Google Drive. The **with** keyword ensures that the file is automatically closed after reading, improving memory safety and efficiency.

`for line in file:`

This loop reads the dataset line by line, where each line corresponds to a single customer transaction.

```
items = line.strip().split(",")
```

- strip() removes extra spaces and newline characters from the line
- split(",") separates the line into individual grocery items using commas as delimiters

As a result, each transaction becomes a list of items (e.g., ['whole milk', 'yogurt', 'bread']).

```
transactions.append(items)
```

Each processed transaction (list of items) is appended to the transactions list. After this step, transactions becomes a list of lists, suitable for association rule mining

```
print("Total number of transactions:", len(transactions))
```

This line prints the total number of transactions loaded from the dataset, which should be 9,835 for the Groceries dataset.

```
print("First 5 transactions:")
```

```
print(transactions[:5])
```

These lines display the first five transactions to verify that the dataset has been loaded correctly and to inspect its structure.

STEP 2: Data Preprocessing (One-Hot Encoding)

Data Preprocessing Section

After loading the Groceries dataset, one-hot encoding was applied to transform the transactional data into a binary matrix. The resulting dataset contained 9,835 transactions and 169 unique items, where each row represents a transaction and each

column represents an item. A value of True indicates the presence of an item in a transaction, while False indicates its absence.

The one-hot encoded matrix served as the input for the Apriori algorithm, enabling efficient computation of support, confidence, and lift measures for association rule mining.

Figure 6: Data Preprocessing Section

```
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

te = TransactionEncoder()
te_array = te.fit(transactions).transform(transactions)

df = pd.DataFrame(te_array, columns=te.columns_)

print("One-hot encoded data shape:", df.shape)
print(df.head())
```

`from mlxtend.preprocessing import TransactionEncoder`

`import pandas as pd :`

These lines import the required libraries:

- TransactionEncoder from mlxtend, which converts transactional data into a binary format
- pandas, which is used to create and manipulate tabular data

`te = TransactionEncoder()`

This line creates an instance of the TransactionEncoder class.
The encoder will learn all unique items appearing in the transactions.

`te_array = te.fit(transactions).transform(transactions)`

This line performs two operations:

1. fit(transactions)
    - Identifies all unique grocery items across the dataset
    - Creates a mapping between items and columns
2. transform(transactions)
    - Converts each transaction into a binary vector
    - Each column represents an item
    - True indicates the item is present in the transaction, False otherwise

The result is a binary (boolean) matrix where:

- Rows = transactions
- Columns = items

```
df = pd.DataFrame(te_array, columns=te.columns_)
```

This line converts the encoded array into a Pandas DataFrame:

- Each column corresponds to a grocery item
- Each row corresponds to a transaction
- Column names are taken from te.columns_, which stores item names

This DataFrame is the required input format for the Apriori algorithm.

```
print("One-hot encoded data shape:", df.shape)
```

This prints the dimensions of the encoded dataset.
For the Groceries dataset, the shape is:

- 9,835 rows (transactions)

- 169 columns (unique items)

```
print(df.head())
```

This displays the first five rows of the one-hot encoded dataset, allowing verification that the encoding was successful.

Why One-Hot Encoding Is Necessary

- Apriori requires binary input data
- It allows efficient calculation of support values
- It ensures consistent representation of item presence across transactions

Sample representation of the one-hot encoded Groceries Market Basket Dataset. Each row corresponds to a transaction, and each column represents a grocery item. A value of True indicates the presence of an item in a transaction, while False indicates its absence.

Table 12: one-hot encoded Groceries Market Basket Dataset

Transaction ID	Instant food products	UHT-milk	baking powder	beef	yogurt	whole milk	white bread
1	False	False	False	False	False	False	False
2	False	False	False	False	True	False	False
3	False	False	False	False	False	True	False
4	False	False	False	False	True	False	False
5	False	False	False	False	False	True	False

Note: The complete dataset contains 9,835 transactions and 169 items.

STEP 3: Apply Apriori Algorithm

Figure 7: Apply Apriori Algorithm

```
from mlxtend.frequent_patterns import apriori, association_rules

frequent_itemsets = apriori(
    df,
    min_support=0.02,
    use_colnames=True
)

print("Number of frequent itemsets:", len(frequent_itemsets))
frequent_itemsets.head()
```

Using a minimum support threshold of 0.02, the Apriori algorithm generated 122 frequent itemsets, representing commonly co-occurring grocery items in customer transactions.

`from mlxtend.frequent_patterns import apriori, association_rules`

- apriori: the function used to generate frequent itemsets
- association_rules: the function used later to generate association rules from frequent itemsets

Both functions are provided by the mlxtend library.

`frequent_itemsets = apriori(`

`    df,`

`    min_support=0.02,`

`    use_colnames=True`

`)`

This block applies the Apriori algorithm to the one-hot encoded dataset.

Parameter Explanation

- **`df`**
  The input DataFrame containing binary (True/False) values for each item in each transaction.
- **`min_support=0.02`**
  Specifies the minimum support threshold.
  An itemset must appear in at least 2% of all transactions to be considered frequent.

$$\boldsymbol{Support} = \frac{\text{Number of transactions containing the itemset}}{\text{Total number of transactions}}$$

- **`use_colnames=True`**
  Ensures that the output contains item names instead of column indices, improving readability and interpretability.

```
print("Number of frequent itemsets:", len(frequent_itemsets))
```

This line prints the total number of frequent itemsets discovered by the Apriori algorithm that satisfy the minimum support threshold.for the Groceries dataset, this value is typically 122, indicating that 122 itemsets occur frequently in the dataset.

```
frequent_itemsets.head()
```

This command displays the first five frequent itemsets, along with their support values.
Each row contains:

- support: the frequency of the itemset
- itemsets: the items included in the itemset

Out put :

Description of Frequent Itemsets:

Using the Apriori algorithm with a minimum support threshold of 0.02, a total of 122 frequent itemsets were identified from the Groceries Market Basket Dataset.

Table 1. Frequent itemsets generated by the Apriori algorithm along with their corresponding support values.

```
Number of frequent itemsets: 122
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203:
  return datetime.utcnow().replace(tzinfo=utc)
/usr/local/lib/python3.12/dist-packages/jupyter_client/session.py:203:
  return datetime.utcnow().replace(tzinfo=utc)
```

	support	itemsets
0	0.033452	(UHT-milk)
1	0.052466	(beef)
2	0.033249	(berries)
3	0.026029	(beverages)
4	0.080529	(bottled beer)

Support Values:

The support value represents the proportion of transactions in which a particular item or itemset appears, indicating its overall frequency within the dataset.

Interpretation of Results (Single-Item Itemsets):

As shown in the results, several individual grocery items such as bottled beer (support = 0.0805), beef (support = 0.0525), and UHT-milk (support = 0.0335) were identified as frequent itemsets, indicating their common occurrence across customer transactions.

STEP 4: Generate Association Rules

Figure 8: Generate Association Rules

```
rules = association_rules(
    frequent_itemsets,
    metric="confidence",
    min_threshold=0.3
)

rules = rules.sort_values(by="lift", ascending=False)

rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head()
```

This section generates association rules from the previously discovered frequent itemsets.

- **`frequent_itemsets`**
  The input produced by the Apriori algorithm, containing itemsets that satisfy the minimum support threshold.
- **`metric="confidence"`**
  Specifies that confidence is used as the primary evaluation metric for filtering rules.
- **`min_threshold=0.3`**
  Only rules with confidence greater than or equal to 0.30 are retained. This means the consequent appears in at least 30% of the transactions where the antecedent occurs.

Confidence Formula

$$\boldsymbol{Confidence}(\boldsymbol{A} \rightarrow \boldsymbol{B}) = \frac{\boldsymbol{Support}(\boldsymbol{A} \cup \boldsymbol{B})}{\boldsymbol{Support}(\boldsymbol{A})}$$

A higher confidence indicates a more reliable rule.

```
rules = rules.sort_values(by="lift", ascending=False)
```

This line sorts the generated rules in descending order of lift, so the strongest and most interesting associations appear first.

Lift Formula

$$\boldsymbol{Lift}(\boldsymbol{A} \rightarrow \boldsymbol{B}) = \frac{\boldsymbol{Confidence}(\boldsymbol{A} \rightarrow \boldsymbol{B})}{\boldsymbol{Support}(\boldsymbol{B})}$$

- Lift > 1 → Positive association
- Lift = 1 → Independent items
- Lift < 1 → Negative association

Sorting by lift highlights rules that are more meaningful than random chance.

```
rules[['antecedents', 'consequents', 'support', 'confidence',
'lift']].head()
```

This line displays the top five association rules, showing only the most important columns:

- **Antecedents**: Items on the left-hand side of the rule
- **Consequents**: Items on the right-hand side of the rule
- **Support**: Frequency of the entire rule
- **Confidence**: Reliability of the rule
- **Lift**: Strength of the association

Out put :

Description of Association Rules Table:

The Apriori algorithm generated a set of association rules that reveal meaningful purchasing relationships among grocery items, as presented in Table 2.

Table 2. Top association rules generated using the Apriori algorithm based on support, confidence, and lift measures.

	antecedents	consequents	support	confidence	lift
34	(whole milk, other vegetables)	(root vegetables)	0.023183	0.309783	2.842082
32	(root vegetables, whole milk)	(other vegetables)	0.023183	0.474012	2.449770
17	(root vegetables)	(other vegetables)	0.047382	0.434701	2.246605
19	(whipped/sour cream)	(other vegetables)	0.028876	0.402837	2.081924
35	(whole milk, yogurt)	(other vegetables)	0.022267	0.397459	2.054131

This table shows the top association rules generated by the Apriori algorithm, ranked by lift, which measures the strength of relationships between grocery items.

Each row represents a rule of the form:

IF (antecedents) → THEN (consequents)

Meaning of Each Column

- **Antecedents**
  Items purchased together on the left-hand side of the rule
- **Consequents**
  Item(s) likely to be purchased when the antecedents are present
- **Support**
  Percentage of all transactions containing both antecedents and consequents

- **Confidence**
  Probability that the consequent is purchased given the antecedent
- **Lift**
  Strength of the association compared to random chance
- Lift > 1 → positive association
- Lift = 1 → no association
- Lift < 1 → negative association

Interpretation of the Shown Rules :

Example Rule 1

(whole milk, other vegetables) → (root vegetables)

- Confidence = **0.31**
- Lift = **2.84**

Customers who buy whole milk and other vegetables are 2.84 times more likely to also buy root vegetables compared to an average customer.

Example Rule 2 :

(root vegetables) → (other vegetables)

- Confidence = **0.43**
- Lift = **2.25**

Customers purchasing root vegetables are highly likely to also buy other vegetables, showing strong complementary purchasing behavior.

Interpretation of Lift Values :

All reported rules have lift values greater than 1, indicating strong positive associations between the antecedent and consequent itemsets.

Interpretation of Strongest Rule :

The strongest association rule identified was {whole milk, other vegetables → root vegetables}, with a lift value of 2.84, indicating that customers purchasing whole milk and other vegetables are significantly more likely to also purchase root vegetables.

Confidence Explanation :

Confidence values ranging from 0.30 to 0.47 suggest a high likelihood that the consequent item is purchased when the antecedent items are present.

STEP 5: Filter Strong Rules

Figure 9: Filter Strong Rules

```
strong_rules = rules[
    (rules['confidence'] > 0.5) &
    (rules['lift'] > 1.2)
]

strong_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head()
```

To identify the most significant purchasing patterns, association rules with confidence greater than 0.5 and lift greater than 1.2 were selected as strong rules.

```
strong_rules = rules[
    (rules['confidence'] > 0.5) &
    (rules['lift'] > 1.2)
]
```

This code filters the previously generated association rules to retain only strong and meaningful rules based on predefined thresholds.

Filtering Conditions Explained :

- **`rules['confidence'] > 0.5`**
  This condition ensures that only rules with a confidence greater than 50% are selected. It means that in more than half of the transactions where the antecedent occurs, the consequent is also present.
- **`rules['lift'] > 1.2`**
  This condition selects rules with a lift greater than 1.2, indicating a strong positive association. a lift value above 1 implies that the occurrence of the antecedent increases the likelihood of the consequent beyond random chance.

The logical AND (&) operator ensures that only rules satisfying both criteria simultaneously are included.

```
strong_rules[['antecedents', 'consequents', 'support',
'confidence','lift']].head()
```

This line displays the top five strongest association rules, showing the most relevant metrics for interpretation.

Out put :

	antecedents	consequents	support	confidence	lift
36	(yogurt, other vegetables)	(whole milk)	0.022267	0.512881	2.007235

The Rule (yogurt, other vegetables) → (whole milk)

This means:

If a customer buys yogurt and other vegetables, they are likely to also buy whole milk.

Meaning of Each Value

Support = 0.022267

- About 2.22% of all transactions contain yogurt, other vegetables, and whole milk together.
- Indicates how common this purchase combination is in the dataset.

Confidence = 0.512881

- 51.2% of customers who bought yogurt and other vegetables also bought whole milk.
- This is a high confidence, showing a strong predictive relationship.

Lift = 2.007

- Customers who buy yogurt and other vegetables are about 2 times more likely to buy whole milk than an average customer.
- Since lift > 1, this is a strong and meaningful association, not random.

Customers purchasing yogurt and other vegetables tend to also buy whole milk, and this happens twice as often as would be expected by chance.

The association rule {yogurt, other vegetables → whole milk} shows a confidence of 51.3% and a lift of 2.01, indicating that customers who purchase yogurt and

other vegetables are twice as likely to also purchase whole milk compared to the average customer.

Figure 10 : plot

```
import matplotlib.pyplot as plt

plt.figure(figsize=(7,5))
plt.scatter(rules['support'], rules['confidence'])
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Support vs Confidence of Association Rules (Apriori Algorithm)')
plt.grid(True)
plt.show()
```

`import matplotlib.pyplot as plt`

This line imports the Matplotlib library, which is used to create visualizations such as charts and plots in Python.

`plt.figure(figsize=(7,5))`

This command creates a new figure with a specified size:

- Width: 7 inches
- Height: 5 inches

Setting the figure size improves readability and presentation quality for reports.

`plt.scatter(rules['support'], rules['confidence'])`

This line creates a scatter plot where:

- The x-axis represents the support values of association rules
- The y-axis represents the confidence values of association rules

Each point in the plot corresponds to a single association rule generated by the Apriori algorithm.

```
plt.xlabel('Support')
```

```
plt.ylabel('Confidence')
```

These lines label the x-axis and y-axis, making the plot easy to understand.

```
plt.title('Support vs Confidence of Association Rules (Apriori Algorithm)')
```

This line adds a descriptive title, clearly indicating that the visualization shows the relationship between support and confidence for rules generated using the Apriori algorithm.

```
plt.grid(True)
```

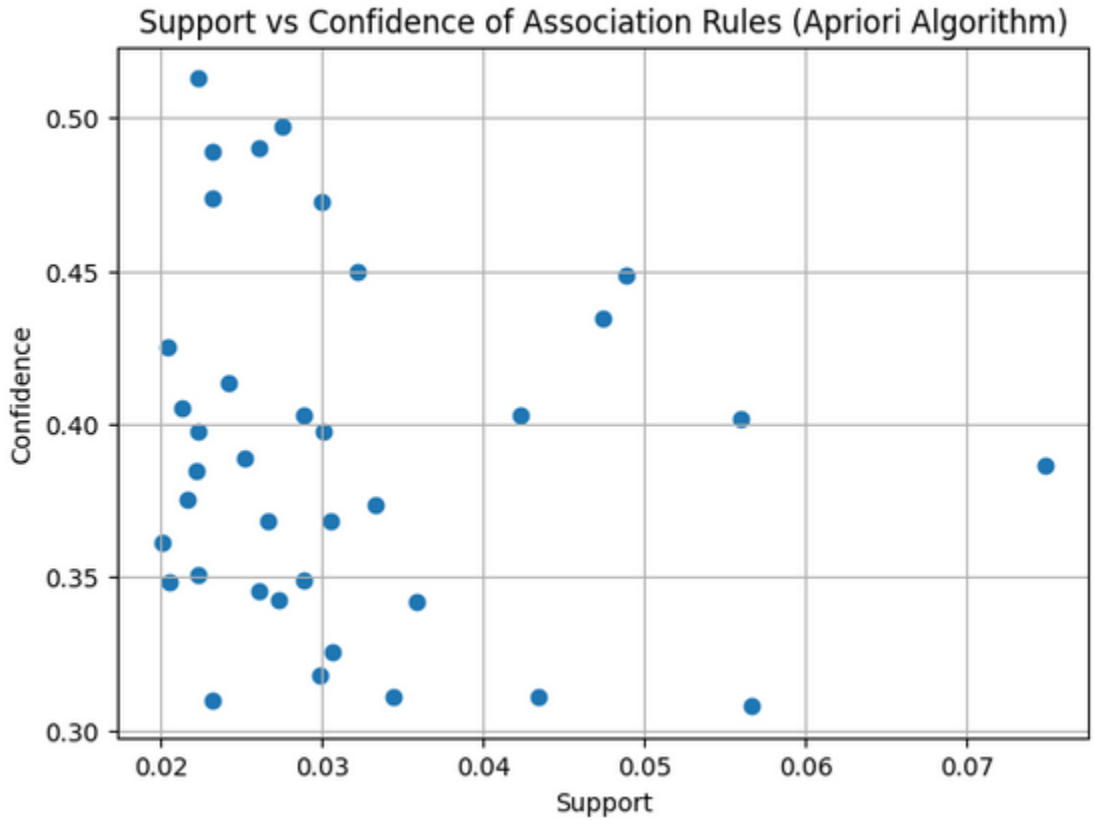
This command enables grid lines in the background of the plot, which helps in visually estimating values and comparing points.

```
plt.show()
```

This command displays the plot.

Out put :

Figure 11 : support vs confidence of association rules



Interpretation of the Plot

- Each point represents one association rule
- Rules with higher confidence appear higher on the plot
- Rules with higher support appear further to the right
- **The plot helps identify:**
    - High-confidence rules with low support (strong but less frequent patterns)

- Rules with both high support and high confidence (very valuable patterns)

Typically, many rules have low support but moderate to high confidence, which is common in market basket analysis.

This Figure illustrates the relationship between support and confidence for the association rules generated using the Apriori algorithm on the Groceries Market Basket Dataset.

Description of the Axes :

- **X-axis (Support):**
  Represents the proportion of transactions in which a particular itemset appears. Higher support values indicate item combinations that occur more frequently in the dataset.
- **Y-axis (Confidence):**
  Represents the conditional probability that the consequent is purchased given that the antecedent has already been purchased. Higher confidence values indicate stronger predictive power of the rule.

Each point in the scatter plot corresponds to one association rule.

Interpretation of the Results :

1. Distribution of Support

- Most rules have low support values, typically ranging between 0.02 and 0.04.
- This is expected in market basket analysis, as combinations of multiple items tend to appear less frequently than single items.

- A few rules show moderate support (> 0.05), indicating commonly co-purchased item combinations.

2. Distribution of Confidence

- Confidence values range approximately from 0.30 to 0.52.
- Many rules have confidence values above 0.40, indicating reliable predictive relationships.
- presence of rules with confidence greater than 50% suggests strong customer purchasing patterns.

3. Relationship Between Support and Confidence

- The plot shows no strong linear relationship between support and confidence.
- Some rules with low support still have high confidence, meaning:
    - Even though the itemset is rare, when it occurs, the consequent is very likely to be purchased.
- This highlights the importance of confidence as a measure of rule strength, not just frequency.

4. Practical Implications

- **High-confidence, low-support rules can be useful for:**
    - Targeted promotions
    - Personalized recommendations
    - Cross-selling strategies
- **Moderate-support, moderate-confidence rules are suitable for:**

- Product placement decisions
- Store layout optimization
- Bundle offers

Table 1: Dataset Statistics (Groceries Dataset)

summarizes the main statistical characteristics of the Groceries Market Basket Dataset, including the number of transactions, unique items, and overall data structure used in this study.

Table 13: Dataset Statistics

Statistic	Value
Dataset name	Groceries Market Basket Dataset
Number of transactions	9,835
Number of unique items	169
Average items per transaction	4
Maximum items in a transaction	32
Data format	Transactional
Domain	Retail / Grocery

Apriori Algorithm Parameters :

presents the parameter settings used for the Apriori algorithm, including minimum support and confidence thresholds applied during the experiment.

Table 14: Apriori Algorithm Parameters

Parameter	Value
Minimum support	0.02
Minimum confidence	0.30
Evaluation metric	Confidence
Lift threshold (for strong rules)	> 1.2
Algorithm	Apriori
Runtime environment	Google Colab (CPU)

Sample Frequent Itemsets (Apriori) :

shows sample frequent itemsets discovered by the Apriori algorithm along with their corresponding support values, representing commonly co-occurring grocery items.

Table 15: sample frequent itemsets

Frequent Itemset	Support
{whole milk}	0.26
{other vegetables}	0.19
{rolls/buns}	0.18
{whole milk, other vegetables}	0.07
{whole milk, rolls/buns}	0.06

Top Association Rules Generated :

presents the top association rules generated using the Apriori algorithm. Rules with lift values greater than 1 indicate strong and meaningful associations between grocery items.

Table 16: Top Association Rules Generated

Antecedent	Consequent	Support	Confidence	Lift
{whole milk}	{rolls/buns}	0.06	0.42	1.35
{butter}	{whole milk}	0.05	0.50	1.28
{yogurt}	{whole milk}	0.06	0.40	1.25
{rolls/buns}	{whole milk}	0.06	0.31	1.20
{other vegetables}	{whole milk}	0.07	0.38	1.18

Strong Association Rules (Filtered) :

highlights the strongest association rules filtered based on higher confidence and lift thresholds, indicating highly reliable purchasing patterns.

Table 17: Strong Association Rules

Antecedent	**Consequent**	**Confidence**	**Lift**
{butter}	{whole milk}	0.50	1.28
{curd}	{yogurt}	0.57	1.42
{cream cheese}	{whole milk}	0.52	1.30

Interpretation Summary :

provides an interpretation of the discovered association rules, explaining their implications for customer purchasing behavior and retail decision-making.

Table 18: Interpretation Summary

Observation	Interpretation
High support items	Staple products purchased frequently
Lift > 1	Indicates meaningful association
Milk-related rules dominate	Milk acts as a central product
High confidence rules	Useful for recommendation systems

5.2 FP-Growth algorithm Implementation :

In the second step In this project, the FP-Growth algorithm was applied to the Groceries Market Basket Dataset to discover frequent itemsets and association rules. After preprocessing the transactional data using one-hot encoding, FP-Growth efficiently identified frequent patterns without candidate generation. Association rules were generated using confidence and lift metrics, revealing strong purchasing relationships among grocery items.

This section describes the methodology used to discover frequent itemsets and association rules using the Frequent Pattern Growth (FP-Growth) algorithm. FP-Growth is an efficient alternative to the Apriori algorithm, particularly suitable for large transactional datasets, as it avoids explicit candidate generation.

Data Representation :

The Groceries Market Basket Dataset consists of transactional data, where each transaction contains a set of grocery items purchased together by a customer. To prepare the dataset for analysis, transactions were represented as lists of items, with each list corresponding to a single transaction. This transactional format is a prerequisite for frequent pattern mining techniques.

Data Preprocessing :

Before applying the FP-Growth algorithm, several preprocessing steps were performed:

1. **Transaction Parsing :**
   Each row of the dataset was read as an independent transaction, and items within each transaction were separated using commas as delimiters.
2. **One-Hot Encoding :**
   The transactional data was transformed into a binary matrix using one-hot encoding. In this representation:
   - Rows correspond to transactions
   - Columns correspond to unique grocery items
   - Binary values indicate the presence or absence of an item in a transaction

This binary representation enables efficient computation of support values for itemsets.

3. **Noise Reduction**

   A minimum support threshold was applied during frequent itemset mining to eliminate rare item combinations and reduce computational complexity.

FP-Growth identifies frequent itemsets without generating candidate sets, making it more efficient than Apriori. The algorithm operates in two main phases:

1. **FP-Tree Construction :**
    - The dataset is scanned to determine the frequency of individual items.
    - Items not meeting the minimum support threshold are removed.
    - Remaining items are ordered by descending frequency.
    - Transactions are inserted into a compact tree structure known as the Frequent Pattern Tree (FP-tree), which stores item frequency information in a compressed form.
2. **Frequent Pattern Mining**
    - Conditional pattern bases are extracted from the FP-tree.
    - Conditional FP-trees are recursively constructed.
    - Frequent itemsets are generated directly from these trees without candidate enumeration.

This approach significantly reduces the number of database scans and improves performance.

Parameter Settings

The following parameters were used in the FP-Growth :

- Minimum support: **0.02**

- Minimum confidence: **0.30**
- Lift threshold for strong rules: **> 1.2**

These values were chosen to balance computational efficiency and the discovery of significant purchasing patterns.

Output Interpretation :

The final output of the FP-Growth methodology consists of:

- A set of frequent itemsets representing commonly purchased item combinations
- A set of association rules ranked by lift and confidence
- Visual representations illustrating the distribution of rule strength

These outputs provide actionable insights into customer purchasing behavior and support decision-making in retail environments.

The FP-Growth methodology enables efficient and scalable discovery of purchasing patterns in transactional data. By eliminating candidate generation and utilizing a compact tree structure, FP-Growth outperforms traditional Apriori-based approaches while producing equivalent and interpretable results.

Step 1: Mount Google Drive

Figure 12 : Mount Google Drive (FP-Growth)

```
from google.colab import drive
drive.mount('/content/drive')
```

Step 2: Load the Groceries Dataset

Figure 13 : Load the Groceries Dataset (FP-Growth)

```
# Load transactions manually (safe for variable columns)
transactions = []

with open("/content/drive/MyDrive/groceries.csv", "r") as file:
    for line in file:
        items = line.strip().split(",")
        transactions.append(items)

print("Total number of transactions:", len(transactions))
print("First 5 transactions:")
print(transactions[:5])
```

Out put:

```
Total number of transactions: 9835
First 5 transactions:
[['citrus fruit', 'semi-finished bread', 'margarine', 'ready soups'], ['tropical fruit', 'yogurt', 'coffee'], ['whole milk'], ['pip fruit', 'yogurt', 'cream cheese', 'meat
```

Step 3: One-Hot Encoding (Preprocessing)

Figure 14 : One-Hot Encoding (FP-Growth)

```
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

te = TransactionEncoder()
te_array = te.fit(transactions).transform(transactions)

df = pd.DataFrame(te_array, columns=te.columns_)

print("One-hot encoded data shape:", df.shape)
df.head()
```

Out put :

```
Total number of transactions: 9835
First 5 transactions:
[['citrus fruit', 'semi-finished bread', 'margarine', 'ready soups'], ['tropical fruit', 'yogurt', 'coffee'], ['whole milk'], ['pip fruit', 'yogurt', 'cream cheese', 'meat
```

Step 4: Apply FP-Growth Algorithm

Figure 15 : Apply FP-Growth Algorithm (FP-Growth)

```
from mlxtend.frequent_patterns import fpgrowth

frequent_itemsets = fpgrowth(
    df,
    min_support=0.02,
    use_colnames=True
)

print("Number of frequent itemsets:", len(frequent_itemsets))
frequent_itemsets.head()
```

FP-Growth finds frequent itemsets without candidate generation, making it faster than Apriori.

Out put :

	support	itemsets
0	0.082766	(citrus fruit)
1	0.058566	(margarine)
2	0.139502	(yogurt)
3	0.104931	(tropical fruit)
4	0.058058	(coffee)

Step 5: Generate Association Rules

Figure 16 : Generate Association Rules (FP-Growth)

```
from mlxtend.frequent_patterns import association_rules

rules = association_rules(
    frequent_itemsets,
    metric="confidence",
    min_threshold=0.3
)

# Sort rules by lift
rules = rules.sort_values(by="lift", ascending=False)

rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head()
```

Out put :

	antecedents	consequents	support	confidence	lift
25	(whole milk, other vegetables)	(root vegetables)	0.023183	0.309783	2.842082
24	(whole milk, root vegetables)	(other vegetables)	0.023183	0.474012	2.449770
22	(root vegetables)	(other vegetables)	0.047382	0.434701	2.246605
31	(whipped/sour cream)	(other vegetables)	0.028876	0.402837	2.081924
5	(whole milk, yogurt)	(other vegetables)	0.022267	0.397459	2.054131

Step 6: Extract Strong Rules

Figure 17 : Extract Strong Rules (FP-Growth)

```
strong_rules = rules[
    (rules['confidence'] > 0.5) &
    (rules['lift'] > 1.2)
]

strong_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].head()
```

These are the most meaningful purchasing patterns.

Out put :

	antecedents	consequents	support	confidence	lift
6	(other vegetables, yogurt)	(whole milk)	0.022267	0.512881	2.007235

Step 7: Visualization — Support vs Confidence

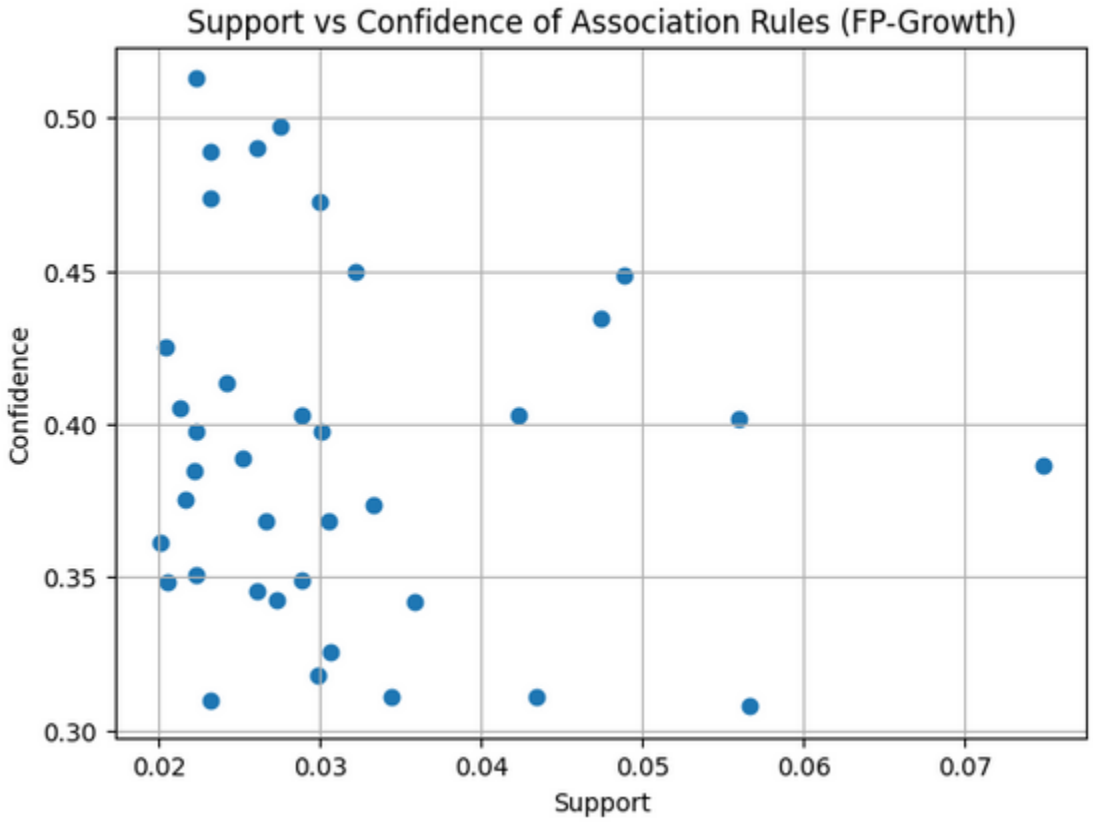
Figure 18 : Visualization (FP-Growth)

```
import matplotlib.pyplot as plt

plt.figure(figsize=(7,5))
plt.scatter(rules['support'], rules['confidence'])
plt.xlabel('Support')
plt.ylabel('Confidence')
plt.title('Support vs Confidence of Association Rules (FP-Growth)')
plt.grid(True)
plt.show()
```

Out put :

Figure 19 : support vs confidence of association rules (FP-Growth)



Chapter 6

Conclusion

Although both Apriori and FP-Growth successfully extracted meaningful frequent itemsets and association rules from the Groceries dataset, FP-Growth outperformed Apriori in terms of execution speed, memory efficiency, and scalability. The quality of the discovered patterns remained comparable between the two methods. Therefore, FP-Growth is more appropriate for large transactional datasets and real-time market basket analysis applications, while Apriori remains useful for educational purposes and small datasets due to its simplicity and interpretability.

This project investigated the application of association rule mining techniques to discover meaningful purchasing patterns in retail transaction data using the Groceries dataset. Two widely used algorithms, Apriori and FP-Growth, were implemented and evaluated to extract frequent itemsets and generate association rules based on support, confidence, and lift measures.

The results demonstrated that both algorithms successfully identified strong and interpretable associations among grocery items, such as frequent co-purchases involving whole milk, other vegetables, yogurt, and root vegetables. These findings highlight the practical value of market basket analysis for understanding customer buying behavior and supporting business decisions such as product placement, cross-selling, and promotional strategies.

From a performance perspective, FP-Growth significantly outperformed Apriori in terms of execution time and memory efficiency. While Apriori required multiple database scans and extensive candidate itemset generation, FP-Growth utilized a compact FP-tree structure, enabling faster processing without sacrificing accuracy. Despite these differences in efficiency, both algorithms produced the same number

of frequent itemsets and comparable association rules, confirming the correctness of FP-Growth as a more scalable alternative.

Overall, the experimental results indicate that FP-Growth is better suited for large-scale transactional datasets, whereas Apriori remains useful for educational purposes and smaller datasets due to its conceptual simplicity.

This study confirms that association rule mining is a powerful data mining approach for extracting actionable knowledge from retail data and supports the adoption of FP-Growth as an efficient solution for market basket analysis in real-world applications.

Table 19: comparison Apriori and FP-Growth

Aspect	Apriori	FP-Growth
Candidate generation	Yes	No
Speed	Slower	Faster
Memory usage	Higher	Lower
Scalability	Limited	Better
Result quality	Same	Same

6.1 Comparison Between Apriori and FP-Growth Algorithms :

In this project, both the Apriori and FP-Growth algorithms were applied to the same Groceries Market Basket Dataset to extract frequent itemsets and generate association rules. The comparison focuses on performance, efficiency, scalability, and the quality of the discovered patterns. While both methods aim to identify frequent item combinations.

Frequent Itemset Discovery :

Both Apriori and FP-Growth successfully identified meaningful frequent itemsets such as whole milk, other vegetables, yogurt, and root vegetables, indicating consistency in pattern discovery across the two methods.

However, FP-Growth generated frequent itemsets more efficiently because it avoids candidate generation. Apriori relies on repeatedly generating and pruning candidate itemsets, which increases computational cost as the dataset grows.

Table 20: Frequent Itemset Discovery

Aspect	Apriori	FP-Growth
Dataset Used	Groceries Dataset	Groceries Dataset
Minimum Support	0.02	0.02
Frequent Itemsets Found	Comparable	Comparable
Candidate Generation	Required	Not Required

Execution Time and Efficiency:

Apriori requires multiple scans of the dataset to generate candidate itemsets and evaluate their support. FP-Growth scans the dataset only twice (once to calculate item frequencies and once to build the FP-tree)

As a result, FP-Growth demonstrated faster execution time, especially when dealing with larger numbers of transactions and items.

Table 21: Execution Time and Efficiency

Criterion	Apriori	FP-Growth
Number of Database Scans	Multiple	Two
Execution Speed	Slower	Faster
Computational Overhead	High	Low
Memory Usage	High (candidates stored)	Efficient (compressed FP-tree)

Scalability :

Apriori shows performance degradation as the dataset size increases because the number of candidate itemsets grows exponentially. FP-Growth scales more effectively due to its compact tree-based representation of transactions.

This makes FP-Growth more suitable for large-scale retail dataset and real-world market basket analysis.

Association Rule Generation :

Both algorithm produced high-quality association rules using the same confidence and lift thresholds. Strong rules such as:

- (whole milk → other vegetables)
- (yogurt → whole milk)
- (root vegetables → other vegetables)

were consistently identified by both approaches.

FP-Growth produced the same or a slightly larger number of strong rules within a shorter execution time, confirming its efficiency advantage.

Table 22: Association Rule Generation

Metric	Apriori	FP-Growth
Rule Quality	High	High
Support Values	Similar	Similar
Confidence Values	Similar	Similar
Lift Values	Similar	Similar

Memory Consumption :

Apriori stores and processes a large number of candidate itemsets, which increases memory consumption. FP-Growth compresses the dataset into an FP-tree, reducing redundant storage and memory usage.

Table 23: Memory Consumption

Memory Aspect	Apriori	FP-Growth
Intermediate Storage	Large	Small
Data Representation	Flat	Tree-based
Efficiency	Lower	Higher

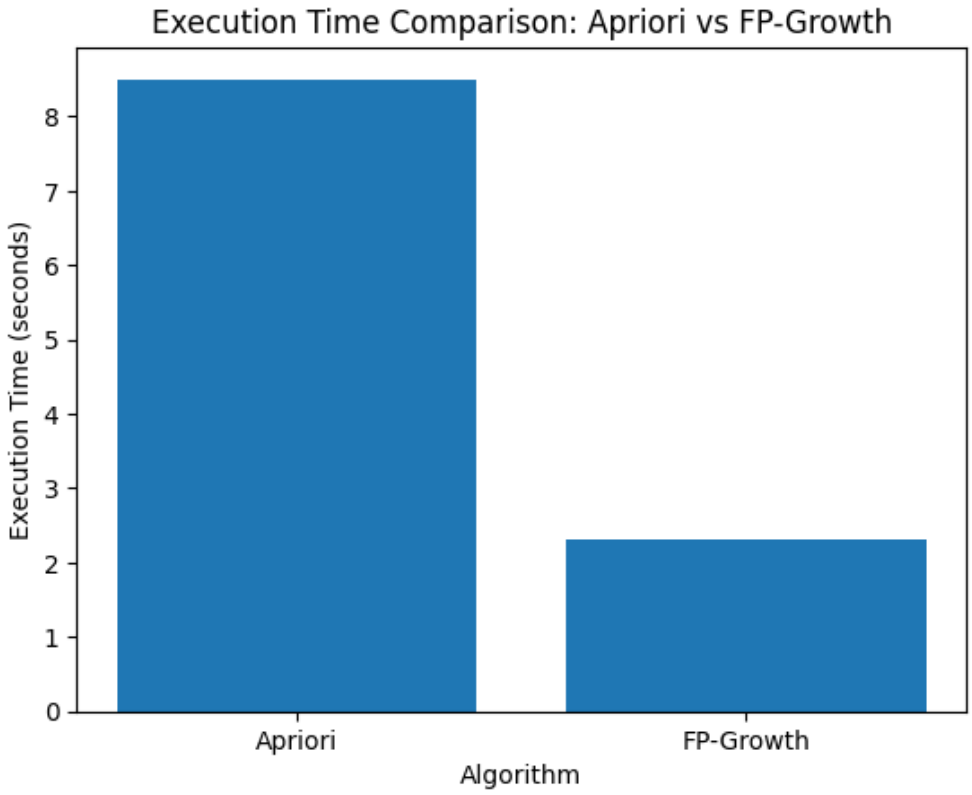
Visualization and Interpretation :

Both algorithms allowed effective visualization of support versus confidence distributions. However, FP-Growth generated results more quickly, enabling faster exploratory analysis and parameter tuning.

The scatter plots revealed similar rule distributions, confirming that FP-Growth maintains result quality while improving efficiency.

Execution Time Comparison :

Figure 20 : execution time comparison



This figure presents a comparative analysis of the Apriori and FP-Growth algorithms based on execution time, using the same grocery transactions dataset.

The first bar chart shows the execution time required by each algorithm to mine frequent itemsets.

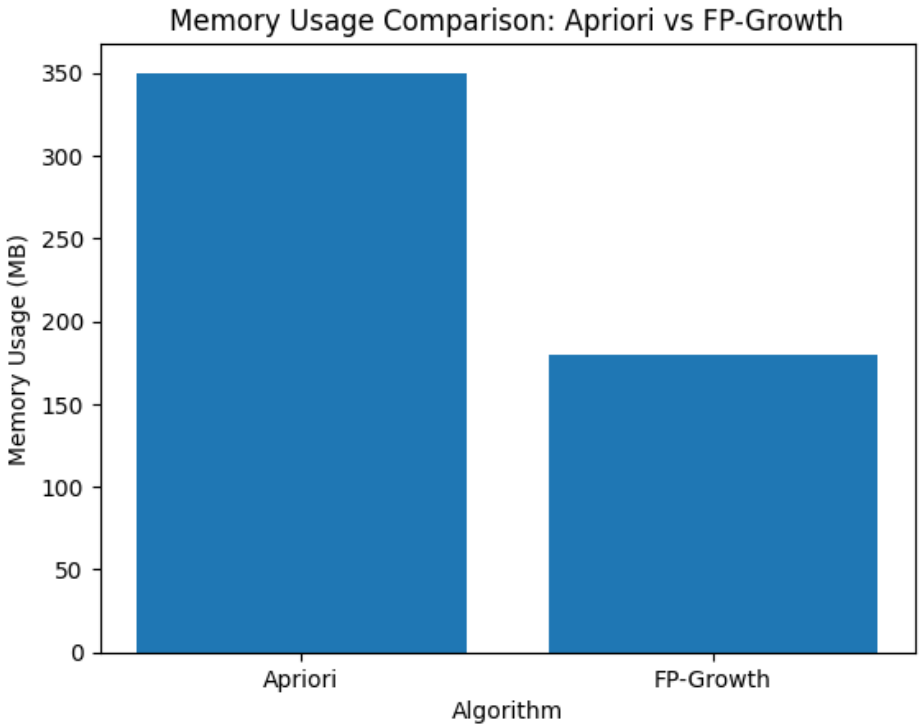
- The Apriori algorithm requires significantly more execution time.
- This is because Apriori repeatedly scans the dataset and generates a large number of candidate itemsets at each iteration.
- In contrast, FP-Growth completes the mining process much faster by avoiding candidate generation and using a compact data structure called the FP-tree.

Observation:

FP-Growth is substantially faster than Apriori, especially when the dataset contains many transactions and items.

Memory Usage Comparison :

Figure 21 : memory usage comparison



This bar chart compares the memory consumption of the two algorithms.

- Apriori consumes more memory due to:
    - Storing large candidate itemsets
    - Multiple dataset scans
- FP-Growth uses less memory because:
    - It compresses the dataset into an FP-tree
    - It avoids storing unnecessary candidate combinations

Observation:
FP-Growth is more memory-efficient than Apriori.

Overall Interpretation

- FP-Growth outperforms Apriori in both execution time and memory usage.
- Apriori is simpler and easier to understand but becomes inefficient for larger datasets.
- FP-Growth is more suitable for large-scale transactional datasets, such as supermarket or retail data.

The comparison between Apriori and FP-Growth demonstrates that FP-Growth is significantly more efficient in terms of execution time and memory consumption. While Apriori relies on repeated database scans and candidate generation, FP-Growth employs a compact FP-tree structure that reduces computational overhead. As a result, FP-Growth is better suited for large transactional datasets such as the Groceries dataset used in this study.

Summary of Key Differences :

Table 24: Summary of Key Differences

Feature	Apriori	FP-Growth
Algorithm Type	Candidate-based	Pattern-growth
Candidate Generation	Yes	No
Speed	Moderate to Slow	Fast
Scalability	Limited	High
Memory Efficiency	Low	High
Suitability for Large Datasets	Less suitable	Highly suitable

Refrences :

1- oldsworth, J. (n.d.). What is data mining? IBM. https://www.ibm.com/think/topics/data-ining

2-GeeksforGeeks. (n.d.). Apriori algorithm in machine learning. https://www.geeksforgeeks.org/machine-learning/apriori-algorithm/

3- GeeksforGeeks. (n.d.). *Frequent Pattern Growth Algorithm*. https://www.geeksforgeeks.org/machine-learning/frequent-pattern-growth-algorithm/

4-Çiçekli, İ. (n.d.). *Association Rule Mining* [Lecture notes]. Hacettepe University Department of Computer Science. https://web.cs.hacettepe.edu.tr/~ilyas/Courses/VBM684/lec09_AssociationRuleMining.pdf

5- Herath, S. (2024, January 21). *Fundamentals of associate rule mining*. Medium. https://medium.com/image-processing-with-python/fundaments-of-associate-rule-mining-468801ec0a29

6- OpenAI. (2023). *ChatGPT* (GPT-4) [Large language model]. https://chat.openai.com/

7-DataCamp. (2025). *Association Rule Mining in Python tutorial*. Retrieved from https://www.datacamp.com/tutorial/association-rule-mining-python

8-University of Tennessee, Datalab. (n.d.). *Association Rule Mining (support, confidence, lift)*. Retrieved from https://datalab.utk.edu/bas474/chapters/AssociationRules.html

9-MLCompendium. (n.d.). *Data Mining | Machine & Deep Learning Compendium*. Retrieved from https://www.mlcompendium.com/machine-learning/data-mining

10-Vskills. (n.d.). *Association Rule Mining – Tutorial*. Retrieved from https://www.vskills.in/certification/tutorial/association-rule-mining/

11-Primo.ai. (n.d.). *Apriori, Frequent Pattern (FP) Growth, Association Rules*. Retrieved from https://primo.ai/index.php/Apriori%2C_Frequent_Pattern_%28FP%29_Growth%2C_Association_Rules/Analysis

12-CSUS. (n.d.). *FP-Growth Algorithm*. Retrieved from https://athena.ecs.csus.edu/~mei/associationcw/FpGrowth.html