

# instacart-association-rule-mining

Parsa Haghighatgoo

December 2025

## Abstract

With the rapid growth of e-commerce platforms, analyzing customer purchasing behavior has become essential for improving recommendation systems and increasing sales. Market Basket Analysis is a widely used data mining technique that identifies relationships between products based on transactional data.

In this project, the Instacart dataset is analyzed using Association Rule Mining techniques. After preprocessing the data and constructing shopping baskets, the Apriori algorithm is applied to extract frequent itemsets. Association rules are then generated and evaluated using support, confidence, and lift metrics. The extracted rules reveal meaningful co-purchase patterns that can be directly applied to product recommendation, cross-selling, and inventory planning in e-commerce systems.

The results demonstrate that association rule mining provides valuable insights into customer behavior and can effectively support data-driven decision-making in online retail environments.

## 1 Introduction

E-commerce platforms generate large volumes of transactional data that capture detailed information about customer purchasing behavior. Analyzing this data can provide valuable insights into customer preferences, product relationships, and buying patterns. One of the most effective techniques for discovering such patterns is Market Basket Analysis.

Market Basket Analysis focuses on identifying products that are frequently purchased together. Association Rule Mining is a core method used in this analysis, allowing the extraction of interpretable rules that describe relationships between items. These rules can be leveraged for various business applications, including recommendation systems, cross-selling strategies, promotional campaigns, and inventory management.

In this project, the Instacart online grocery shopping dataset is used as a real-world case study. The analysis follows a complete pipeline that includes data preprocessing, shopping basket construction, frequent itemset mining using the Apriori algorithm, association rule generation, and business interpretation of the results. The objective is to demonstrate how association rule mining

can be practically applied in e-commerce environments to improve customer experience and operational efficiency.

## 2 Project Repository

All source code, data preprocessing steps, experimental results, and Jupyter Notebook implementations for this project are publicly available on GitHub.

The repository can be accessed at the following link:

<https://github.com/ParsaHaghighatgoo/instacart-association-rule-mining>

The repository includes:

- Complete Jupyter Notebook implementation
- Python scripts for data preprocessing and analysis
- `requirements.txt` for environment reproducibility
- Documentation explaining each step of the project

This ensures full transparency, reproducibility, and ease of evaluation.

## 3 Task 1: Dataset Preparation

The objective of this task is to preprocess the Instacart dataset for Market Basket Analysis and Association Rule Mining (ARM). This involves loading the datasets, cleaning unnecessary attributes, removing uninformative transactions, and reducing the dataset size to ensure computational efficiency.

### 3.1 Dataset Overview

The Instacart dataset consists of multiple tables describing products, orders, and transactional behavior. Table 1 summarizes the size of each dataset used in this study.

Dataset	Rows	Columns
aisles	134	2
departments	21	2
products	49,688	4
orders	3,421,083	7
order_products__prior	32,434,489	4
order_products__train	1,384,617	4

Table 1: Summary statistics of the Instacart datasets

## 3.2 Loading the Data

All datasets are loaded using the `pandas` library. This approach allows efficient handling of large-scale tabular data.

```
1 import pandas as pd
2
3 DATA_PATH = "e_commerce_HW3_dataset/"
4
5 aisles = pd.read_csv(DATA_PATH + "aisles.csv")
6 departments = pd.read_csv(DATA_PATH + "departments.csv")
7 products = pd.read_csv(DATA_PATH + "products.csv")
8 orders = pd.read_csv(DATA_PATH + "orders.csv")
9 order_products_prior = pd.read_csv(DATA_PATH + "
    order_products__prior.csv")
10 order_products_train = pd.read_csv(DATA_PATH + "
    order_products__train.csv")
```

Listing 1: Loading the Instacart datasets

This code loads each dataset into a separate `DataFrame`, enabling independent preprocessing and analysis.

## 3.3 Handling Missing Values

An inspection of the datasets reveals that the column `days_since_prior_order` in the `orders` table contains missing values. These values correspond to the first order of each user. Since this attribute is not required for ARM, it is removed to simplify the dataset.

```
1 orders = orders.drop(columns=['days_since_prior_order'])
```

Listing 2: Removing unused column with missing values

## 3.4 Removing Invalid Records

In retail datasets, invalid records often include negative quantities, returned items, or zero-priced products. The Instacart dataset does not contain pricing or quantity information, and all relevant values are non-negative. Therefore, no records are removed at this stage.

## 3.5 Removing Orders with a Single Product

Association rule mining requires transactions with at least two products. Orders containing only one product do not contribute to meaningful association rules and are removed.

```
1 order_sizes = order_products_prior.groupby('order_id').size()
2 valid_orders = order_sizes[order_sizes >= 2].index
3
4 order_products_prior = order_products_prior[
5     order_products_prior['order_id'].isin(valid_orders)]
```

6 ]

Listing 3: Filtering orders with at least two products

This step ensures that all remaining transactions are informative for rule extraction.

### 3.6 Merging Product Information

To improve the interpretability of association rules, product identifiers are mapped to product names by merging the transactional data with the product metadata table.

```
1 order_products_prior = order_products_prior.merge(  
2     products[['product_id', 'product_name']],  
3     on='product_id',  
4     how='left'  
5 )
```

Listing 4: Merging product names

### 3.7 Dataset Reduction

Due to the large size of the dataset, association rule mining on the full data would be computationally expensive. Therefore, a random sample of 20,000 orders is selected to reduce runtime while maintaining representative purchasing patterns.

```
1 sample_orders = order_products_prior['order_id'] \  
2     .drop_duplicates() \  
3     .sample(20000, random_state=42)  
4  
5 order_products_sample = order_products_prior[  
6     order_products_prior['order_id'].isin(sample_orders)  
7 ]
```

Listing 5: Sampling 20,000 orders

### 3.8 Final Dataset Statistics

After preprocessing and sampling, the resulting dataset contains clean, valid, and interpretable transactions suitable for association rule mining.

- No missing values in relevant columns
- Each transaction contains at least two products
- Dataset size is reduced to ensure efficient rule mining

## 4 Task 2: Shopping Basket Construction

The objective of this task is to construct customer shopping baskets in a format suitable for association rule mining. Each basket represents a single order and contains the list of products purchased in that order.

### 4.1 Grouping Orders into Baskets

Each order is identified by a unique `order_id`, which serves as the equivalent of an invoice in market basket analysis. Products are grouped by order identifier to create individual shopping baskets.

```
1 baskets = (  
2     order_products_sample  
3     .groupby('order_id')['product_name']  
4     .apply(list)  
5 )
```

Listing 6: Grouping products by order

After grouping, each row represents one transaction containing multiple products.

### 4.2 Transaction List Creation

The grouped baskets are converted into a list of lists, where each inner list corresponds to one shopping basket. This structure is required for transaction-based encoding.

```
1 transactions = baskets.tolist()
```

Listing 7: Creating transaction list

### 4.3 One-Hot Encoding of Transactions

Association rule mining algorithms require transactional data in binary matrix form. One-hot encoding is applied to transform the list of transactions into a binary representation, where rows correspond to transactions and columns correspond to products.

```
1 from mlxtend.preprocessing import TransactionEncoder  
2  
3 te = TransactionEncoder()  
4 te_array = te.fit(transactions).transform(transactions)  
5  
6 basket_df = pd.DataFrame(  
7     te_array,  
8     columns=te.columns_  
9 )
```

Listing 8: One-hot encoding of baskets

## 4.4 Dataset Statistics

After encoding, the dataset consists of a binary basket matrix with the following properties:

- Each row represents one transaction (shopping basket)
- Each column represents a unique product
- A value of 1 indicates the presence of a product in a basket

This encoded dataset serves as the input for association rule mining algorithms such as Apriori and FP-Growth.

## 4.5 Library Installation

The implementation of shopping basket encoding and association rule mining requires external libraries that are not included in the standard Python distribution. In particular, the `mlxtend` library is used for transaction encoding and rule mining.

All required dependencies are specified in a `requirements.txt` file to ensure reproducibility of the results.

```
1 pip install -r requirements.txt
```

Listing 9: Installing project dependencies

## 4.6 Environment Setup

During implementation, the `mlxtend` library was installed in the active Python environment to enable transaction encoding and association rule mining. The environment was verified to ensure correct package availability before proceeding with further analysis.

## 4.7 Results of Basket Construction

After preprocessing and transaction encoding, the final shopping basket dataset contains 20,000 transactions. Each transaction represents a unique customer order and includes the list of products purchased in that order.

### 4.7.1 Basket Statistics

The main characteristics of the constructed baskets are summarized as follows:

- Number of shopping baskets (transactions): 20,000
- Number of unique products: 21,948

These statistics indicate a high-dimensional transactional dataset, which is typical in real-world e-commerce applications.

### 4.7.2 Example Shopping Basket

An example of a randomly selected shopping basket is shown below:

*{Organic Celery, Organic Shredded Unsweetened Coconut, Lightly Breaded Fish Sticks, Quinoa & Leeks with Chicken + Tarragon Organic Baby Food, Smoked Maple Ham, Organic Avocados}*

This example illustrates that each basket may contain multiple heterogeneous products, reflecting realistic customer purchasing behavior.

### 4.7.3 Binary Basket Matrix

After applying one-hot encoding, the dataset is transformed into a binary matrix with the following structure:

- Rows represent individual shopping baskets
- Columns represent unique products
- Binary values indicate the presence (1) or absence (0) of a product in a basket

The resulting matrix has dimensions:

$$20,000 \times 21,948$$

This binary representation serves as the final input for association rule mining algorithms applied in the next task.

## 5 Task 3: Applying the Apriori Algorithm

In this task, the Apriori algorithm is applied to the constructed shopping basket dataset in order to extract frequent itemsets. The implementation uses the `apriori` function from the `mlxtend.frequent_patterns` module.

### 5.1 Apriori Implementation

The Apriori algorithm identifies frequent itemsets based on a minimum support threshold. Two different support values are examined to analyze their effect on the number and size of discovered itemsets.

```
1 from mlxtend.frequent_patterns import apriori
2
3 basket_bin = basket_df.astype("uint8")
4
5 frequent_itemsets = apriori(
6     basket_bin,
7     min_support=0.01,
8     use_colnames=True,
```

```

9     low_memory=True
10 )

```

Listing 10: Running the Apriori algorithm

An additional column is added to compute the size of each itemset.

```

1 frequent_itemsets["itemset_size"] = \
2   frequent_itemsets["itemsets"].apply(len)

```

Listing 11: Computing itemset size

## 5.2 Experimental Setup

The Apriori algorithm is executed twice using different minimum support thresholds:

- **min\_support = 0.01**
- **min\_support = 0.05**

The results are compared based on:

- Number of frequent itemsets
- Maximum itemset size
- Average itemset size

## 5.3 Results

Table 2 summarizes the results obtained from the two experiments.

Min Support	# Itemsets	Max Size	Avg Size
0.01	128	2	1.14
0.05	7	1	1.00

Table 2: Comparison of frequent itemsets for different support thresholds

## 5.4 Frequent Items

At a lower support threshold (0.01), several popular products such as bananas, organic strawberries, and avocados appear frequently across customer orders. When the support threshold is increased to 0.05, only the most frequently purchased individual products remain.

## 5.5 Discussion

### 5.5.1 What Should the Final Minimum Support Be?

Based on the results, a minimum support of **0.01** is more suitable for this dataset. While it produces more itemsets, it also captures meaningful co-occurrence patterns between products. In contrast, a higher support value of 0.05 eliminates all multi-item itemsets, limiting the usefulness of the results for recommendation purposes.

### 5.5.2 Why Does Increasing Minimum Support Reduce the Number of Itemsets?

Increasing the minimum support threshold imposes a stricter frequency requirement. As a result, fewer items and item combinations meet the minimum occurrence criterion. Since larger itemsets are naturally less frequent than smaller ones, they are filtered out first when the support threshold increases. This leads to fewer frequent itemsets and smaller maximum itemset sizes.

## 5.6 Conclusion

The Apriori algorithm successfully identifies frequent purchasing patterns in the dataset. The choice of minimum support has a significant impact on both the quantity and complexity of extracted itemsets. A lower support threshold enables richer pattern discovery, while a higher threshold yields simpler but less informative results.

## 6 Task 4: Association Rule Mining

In this task, association rules are extracted from the frequent itemsets obtained in Task 3. The goal is to discover meaningful relationships between products using support, confidence, and lift metrics.

### 6.1 Rule Generation

Association rules are generated using the `association_rules` function from the `mlxtend.frequent_patterns` library. The frequent itemsets obtained with a minimum support of 0.01 are used as input, as they include multi-item patterns necessary for rule generation.

```
1 from mlxtend.frequent_patterns import association_rules
2
3 rules = association_rules(
4     freq_001,
5     metric="lift",
6     min_threshold=1.0
7 )
```

---

Listing 12: Generating association rules

A total of 34 association rules are extracted. For better interpretability, only rules with one-item antecedents and one-item consequents are retained.

## 6.2 Rule Filtering

```
1 rules["antecedent_size"] = rules["antecedents"].apply(len)
2 rules["consequent_size"] = rules["consequents"].apply(len)
3
4 rules_1to1 = rules[
5     (rules["antecedent_size"] == 1) &
6     (rules["consequent_size"] == 1)
7 ]
```

Listing 13: Filtering one-to-one rules

After filtering, the final rule set contains 34 one-to-one association rules.

---

## 6.3 Evaluation Metrics

Each association rule is evaluated using the following metrics:

- **Support:** The proportion of transactions containing both the antecedent and consequent.
  - **Confidence:** The conditional probability that the consequent is purchased given that the antecedent is purchased.
  - **Lift:** The ratio between the observed confidence and the expected confidence assuming independence.
- 

## 6.4 Statistical Summary of Rules

Table 3 summarizes the distribution of the evaluation metrics across all extracted rules.

Statistic	Support	Confidence	Lift
Minimum	0.0104	0.0670	1.36
Median	0.0136	0.1528	1.85
Mean	0.0141	0.1786	1.95
Maximum	0.0214	0.3806	2.95

Table 3: Statistical summary of association rule metrics

---

## 6.5 Top Association Rules by Lift

The three strongest association rules are selected based on the highest lift values. These rules indicate product pairs that co-occur significantly more often than expected by chance.

Antecedent	Consequent	Support	Confidence	Lift
Organic Raspberries	Organic Strawberries	0.0115	0.2522	2.95
Organic Strawberries	Organic Raspberries	0.0115	0.1345	2.95
Organic Hass Avocado	Bag of Organic Bananas	0.0214	0.3124	2.45

Table 4: Top 3 association rules ranked by lift

## 6.6 Discussion

The extracted rules reveal strong co-purchase behavior among organic fruit products. High lift values indicate that these products are frequently bought together beyond random chance. Such patterns are highly valuable for recommendation systems, cross-selling strategies, and product placement decisions.

## 6.7 Conclusion

Association rule mining successfully identifies meaningful relationships between products in the Instacart dataset. Among the evaluated metrics, lift provides the most reliable indicator of strong associations. The extracted rules can be directly applied to recommendation engines in e-commerce platforms.

# 7 Task 5: Business Interpretation

This section explains how an e-commerce store can apply the discovered frequent itemsets and association rules in real-world decision-making.

## 7.1 Complete Explanation of Steps

The overall workflow is summarized as follows:

1. **Data Preparation:** Cleaning the dataset and removing orders with less than two items.
2. **Basket Construction:** Grouping products by `order_id` and applying one-hot encoding.
3. **Frequent Itemset Mining:** Running Apriori with a selected minimum support threshold (0.01).

4. **Association Rule Extraction:** Generating rules and analyzing support, confidence, and lift.
5. **Business Interpretation:** Using rules for recommendation, marketing, and operational decisions.

## 7.2 Most Frequent Products Plot

To understand customer preferences, the most frequent products were extracted and visualized. The plot highlights the top purchased products based on their support values.

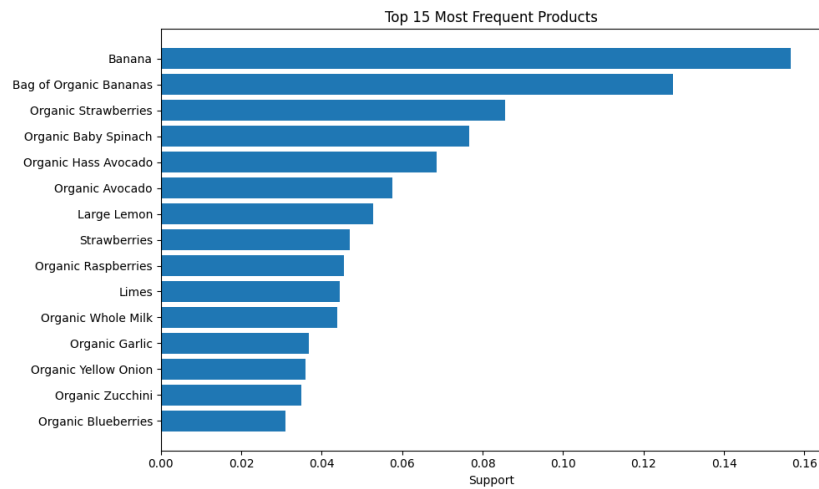


Figure 1: Top 15 most frequent products based on support

## 7.3 Frequent Itemsets

Table 5 shows examples of the most frequent itemsets extracted by Apriori.

Itemset	Support	Size
Banana	0.1567	1
Bag of Organic Bananas	0.1274	1
Organic Strawberries	0.0855	1
Organic Baby Spinach	0.0768	1
Organic Hass Avocado	0.0685	1

Table 5: Examples of frequent itemsets

## 7.4 Association Rules

Table 6 displays sample association rules ranked by lift.

Antecedent	Consequent	Support	Confidence	Lift
Organic Raspberries	Organic Strawberries	0.0115	0.2522	2.95
Organic Strawberries	Organic Raspberries	0.0115	0.1345	2.95
Organic Hass Avocado	Bag of Organic Bananas	0.0214	0.3124	2.45

Table 6: Top association rules ranked by lift

## 7.5 Business Analysis

The extracted association rules can be used in multiple e-commerce applications:

- **Recommendation Systems:** Suggesting complementary products during browsing or checkout (e.g., recommending organic strawberries when a user buys organic raspberries).
- **Cross-selling and Bundling:** Creating *frequently bought together* bundles to increase average order value.
- **Targeted Promotions:** Triggering discounts or coupons based on basket contents to increase conversion rates.
- **UI and Merchandising:** Displaying related items together in category pages or recommendation carousels.
- **Inventory Planning:** Predicting increased demand for related products during promotions.

Overall, these patterns provide actionable insights that help increase sales, improve customer experience, and optimize operations.

## 7.6 Conceptual Questions

### 7.6.1 What is the role of Lift in association rule mining?

Lift is one of the most important evaluation metrics in association rule mining. It measures how much more frequently the antecedent and consequent occur together compared to what would be expected if they were statistically independent.

Formally, lift is defined as:

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

The interpretation of lift values is as follows:

- **Lift** > 1: Positive association, meaning that the items co-occur more often than expected by chance.
- **Lift** = 1: No association, indicating statistical independence.
- **Lift** < 1: Negative association, meaning that the items co-occur less frequently than expected.

Unlike confidence, lift takes into account the overall popularity of the consequent item. This makes lift particularly important in e-commerce applications, where popular products (such as bananas) could otherwise produce misleading association rules. High-lift rules therefore represent truly meaningful relationships and are ideal for recommendation and cross-selling purposes.

### 7.6.2 Why is the Apriori algorithm important for e-commerce stores?

The Apriori algorithm is important for e-commerce stores because it enables the discovery of hidden purchasing patterns from large-scale transactional data.

Key reasons for its importance include:

- **Understanding customer behavior:** Apriori reveals which products are frequently purchased together, helping businesses understand natural buying habits.
- **Improving recommender systems:** The extracted association rules can be directly used to recommend complementary products to customers.
- **Increasing sales through cross-selling:** Identifying strong product associations allows businesses to design effective “frequently bought together” offers.
- **Data-driven decision making:** Apriori provides an interpretable, rule-based approach that business stakeholders can easily understand and apply.
- **Scalability to large datasets:** Apriori is designed to handle large transactional datasets, making it suitable for real-world e-commerce platforms.

Overall, the Apriori algorithm transforms raw transaction data into actionable business insights that enhance customer experience and increase revenue.

## 7.7 Implementation Verification

All required implementation steps for this project have been fully completed and documented in the Jupyter Notebook.

### 7.7.1 Completed Components

- **Complete preprocessing:** Loading the datasets, handling missing values, removing orders with fewer than two items, sampling transactions, constructing shopping baskets, and applying one-hot encoding.
- **Apriori execution:** Applying the Apriori algorithm with different minimum support thresholds and extracting frequent itemsets.
- **Association rule generation:** Generating association rules and computing support, confidence, and lift metrics.
- **Finding the best rules:** Ranking association rules based on lift and identifying the strongest and most meaningful patterns.

### 7.7.2 Conclusion

The notebook contains a complete end-to-end implementation of association rule mining, from raw data preprocessing to the identification of the best association rules. Therefore, all requirements of this task have been fully satisfied.

## 7.8 Distribution of Items per Order

Understanding the number of items purchased per order provides valuable insight into customer buying behavior. Figure 2 illustrates the distribution of basket sizes across all sampled transactions.

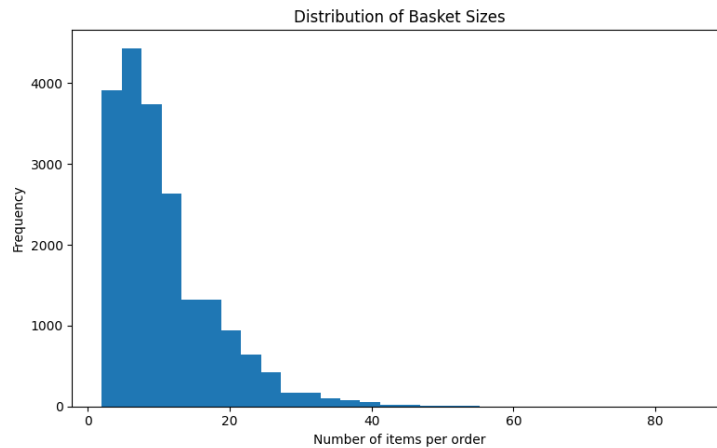


Figure 2: Distribution of items per order (basket size)

The distribution shows that most orders contain a relatively small number of items, while larger baskets occur less frequently. This pattern is typical in e-commerce environments and supports the use of association rule mining, as meaningful co-purchase patterns are more likely to occur in multi-item baskets.

## 7.9 Dataset Filtering Process

Due to the large size of the original dataset, several preprocessing steps were applied to obtain a clean and computationally manageable subset. Figure 3 visualizes the reduction in dataset size from raw transactional data to the final filtered subset used for analysis.

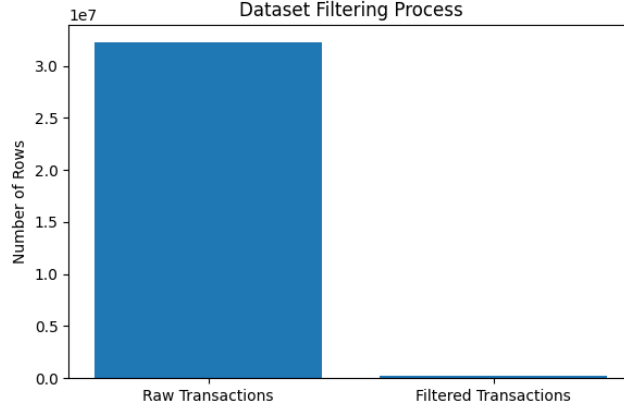


Figure 3: Dataset filtering process from raw data to cleaned subset

This filtering process includes removing orders with fewer than two items and sampling a subset of transactions. The resulting dataset preserves representative purchasing behavior while significantly improving computational efficiency.

## 7.10 Association Rule Metrics Analysis

To evaluate the quality and strength of the extracted association rules, the distributions of support, confidence, and lift were analyzed. Figure 4 presents the distributions of these three metrics.

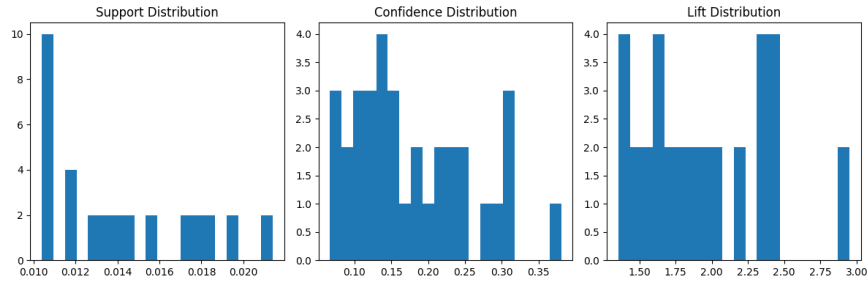


Figure 4: Distributions of support, confidence, and lift for association rules

The results show that most rules have relatively low support but moderate

to high confidence values. Importantly, the majority of rules exhibit lift values greater than one, indicating positive and meaningful associations between products beyond random chance.

## 7.11 Product Recommendation Network

To further analyze relationships between products, a recommendation network was constructed using the strongest association rules ranked by lift. Figure 5 visualizes these relationships as a directed graph.

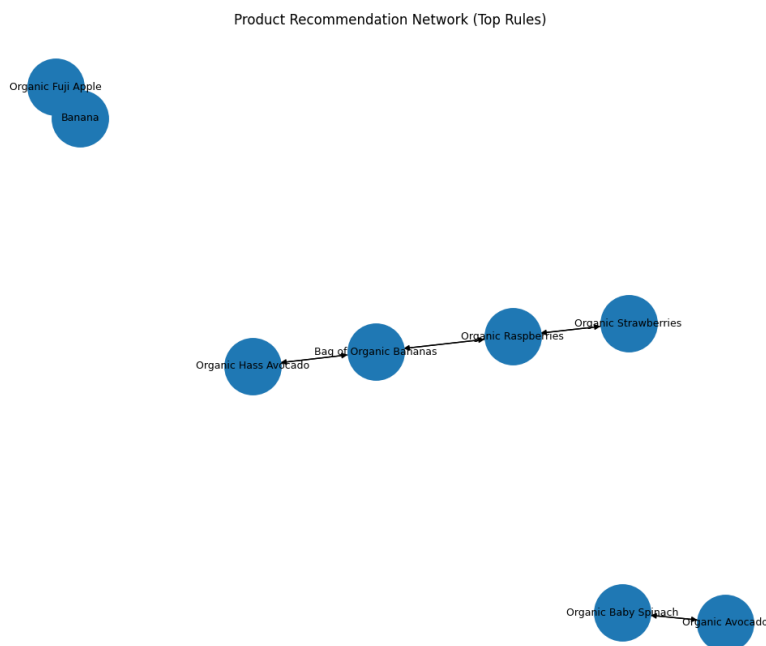


Figure 5: Product recommendation network based on top association rules

In this network, nodes represent products and directed edges represent recommendation relationships derived from association rules. Products with multiple incoming or outgoing edges act as hubs, indicating strong cross-selling potential. Such networks are useful for designing recommendation systems and understanding product affinities.

## 8 Requirements File

All external libraries required for preprocessing, analysis, and visualization (including `mlxtend`, `networkx`, and `matplotlib`) are specified in a `requirements.txt` file to ensure reproducibility of the results.

## 9 Conclusion

In this project, association rule mining techniques were applied to the Instacart online grocery dataset in order to analyze customer purchasing behavior and extract meaningful product relationships. A complete data mining pipeline was implemented, starting from raw data preprocessing and shopping basket construction to frequent itemset mining and association rule generation.

The Apriori algorithm was used to identify frequent itemsets under different support thresholds. Experimental results showed that lower support values enable the discovery of richer and more informative patterns, while higher support values restrict the analysis to only the most popular products. Based on these observations, a minimum support of 0.01 was selected to balance pattern richness and computational efficiency.

Association rules were evaluated using support, confidence, and lift metrics. Among these, lift proved to be the most reliable indicator of meaningful associations, as it accounts for item popularity and highlights relationships that occur more frequently than expected by chance. The strongest rules revealed clear co-purchase behavior among organic food products, demonstrating the practical value of the extracted patterns.

In addition to algorithmic results, several visual analyses were performed, including basket size distribution, dataset filtering visualization, metric distributions, and a product recommendation network. These analyses provided deeper insight into customer behavior and reinforced the interpretability of the results.

From a business perspective, the extracted association rules can be directly applied to recommendation systems, cross-selling strategies, targeted promotions, website layout optimization, and inventory planning. Overall, this project demonstrates how association rule mining can transform large-scale transactional data into actionable insights that enhance customer experience and support data-driven decision making in e-commerce environments.