NANKAI UNIVERSITY

COLLEGE OF SOFTWARE

Subject: MACHINE LEARNING

Assignment-1

Basket Analysis

Professor

WANG CHAO

Submitted by

Somoeurn Virakden   2120246050

Submission Date

2025-April-08

Contents

1 Introduction

In this project, we explore **Basket Analysis**—a data mining technique used to discover associations or patterns among items purchased together in transactional data. It helps uncover which products are frequently bought together, allowing businesses to make data-driven decisions such as product bundling, cross-selling, and targeted promotions. We aim to discover meaningful patterns and customer behavior insights from transactional datasets. Our analysis is based on two approaches:

- The **Standard Apriori Algorithm**, using the `mlxtend` library this library learn from the [3]Open Source Software (JOSS), 2018.
- A **Clustering-based** approach, inspired by the paper [1]"*Fast Discovery of association rules*" by research at the University of Helsinki (1996), which emphasizes performance optimization by leveraging clustering on frequent itemsets.

By comparing both approaches, we aim to provide a more insightful understanding of association rule mining techniques and how they can be applied and interpreted in real-world scenarios.

2 Objective

The objective of this project is to apply **association rule mining** techniques, specifically the two algorithms **Standard Apriori Algorithm, Clustering-base Apriori**, to analyze transactional data and extract patterns that can reveal customer purchasing behavior. Our focus is not only on discovering product combinations that frequently occur together in shopping baskets but also on understanding the strength and relevance of these associations. To achieve this, the project is designed with the following key goals:

- To implement and apply the **Standard Apriori Algorithm** on a real transaction dataset to discover frequent itemsets and generate association rules.
- To explore a **Clustering-based Apriori** approach that groups frequent itemsets for better pattern analysis, aligning with ideas proposed in Toivonen's research.
- To compare the effectiveness, visualization, and interpretability of both methods.
- To enhance our understanding of association rule mining beyond basic algorithm use by integrating clustering and critical evaluation.

3 Methodology

3.1 Dataset Description

This project makes use of two datasets provided by **Prof. Wang Chao**, in text format to conduct market basket analysis:

- `productList.txt`: This file contains **1,560 unique product IDs**, each associated with one or more product names. Some IDs correspond to a single product name, while others may represent grouped or bundled products, listing multiple names. This product reference list is essential for translating numerical IDs into human-readable item names when interpreting the final association rules.
- `Sales1998.txt`: This file includes **34,070 transaction** records, where each line represents a transaction consisting of space-separated product IDs. Each transaction reflects the products purchased together by a customer, serving as the basis for identifying co-occurring items and frequent purchasing patterns.

3.2 Implementation Process

3.2.1 Standard Apriori Algorithm

To implement Basket Analysis applied with **Standard Appriori algorithm**, we followed a structured pipeline that transforms raw transaction data into useful association rules. This implementation was completed using Python and key libraries such as `pandas`, `mlxtend`, and `NumPy`. The following is a detailed breakdown of the steps taken:

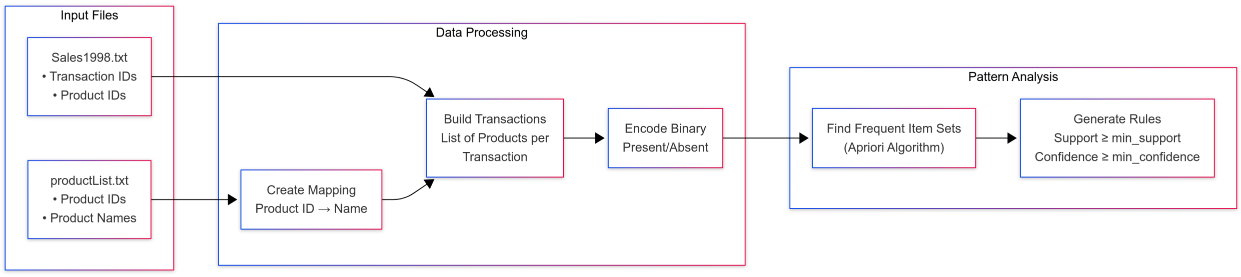


Figure 1: standard-apriori-diagram

Step 1: Data Loading

We first loaded two data files:

- `productList.txt`, which contains 1,560 product IDs. Each ID may map to one or multiple product names, reflecting product name variations or aliases.
- `Sales1998.txt`, consisting of 34,070 transaction records, where each line is a shopping basket with a list of product IDs.

Step 2: Mapping Product IDs to Names A dictionary was created from `productList.txt` to map product IDs to their human-readable product names. This step ensures that the final output rules are interpretable for business analysis.

Step 3: Transaction Formatting The list of product IDs from `Sales1998.txt` was cleaned and split into **individual transactions**. Each transaction was stored as a list of product names by converting the IDs using the mapping dictionary.

Step 4: Binary Encoding Using `TransactionEncoder` from `mlxtend.preprocessing`, we transformed the cleaned transaction list into a binary matrix format, where each row represents a transaction and each column corresponds to a product. A value of **1** indicates presence of the item, and **0** indicates absence. This format is required for applying the Apriori algorithm.

Step 5: Applying Apriori Algorithm We applied the standard Apriori algorithm using `mlxtend.frequent_patterns.apriori()` to find frequent itemsets with a specified minimum support threshold.

Step 6: Generating Association Rules Using `mlxtend.frequent_patterns.association_rules()`, we derived association rules from the frequent itemsets based on confidence and lift thresholds.

3.2.2 Clustering-Base Apriori

To complement the traditional Apriori method, we implemented a **clustering-based approach** to group the frequent itemsets into distinct clusters. This helps uncover grouped behavioral patterns among products that frequently co-occur. The steps below follow the flow illustrated in the accompanying diagram:

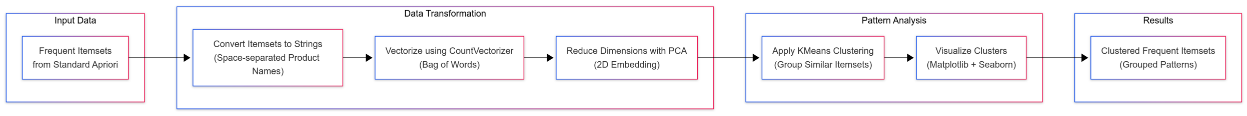

Input Data
Frequent Itemsets from Standard Apriori
Data Transformation
Convert Itemsets to Strings (Space-separated Product Names)
Vectorize using CountVectorizer (Bag of Words)
Reduce Dimensions with PCA (2D Embedding)
Pattern Analysis
Apply KMeans Clustering (Group Similar Itemsets)
Visualize Clusters (Matplotlib + Seaborn)
Results
Clustered Frequent Itemsets (Grouped Patterns)

Figure 2: clustering-diagram

Step 1: Reuse Frequent Itemsets We reused the frequent itemsets obtained from the standard Apriori algorithm (those with length $\geqslant 2$) as the input for clustering. These itemsets represent frequently co-purchased product combinations and serve as the foundation for identifying behavioral patterns.

Step 2: Convert Itemsets to Strings (Space-separated Product Names) Each frequent itemset, initially stored as a Python `frozenset`, was converted into a string of product names separated by spaces. This makes the data compatible with the text vectorization process.

Step 3: Vectorize Itemsets (Bag of Words Approach) Using `CountVectorizer` from `sklearn.feature_extraction.text`, we converted each stringified itemset into a numerical feature vector based on word frequency (i.e., product names). Each vector represents the presence of items in the itemset, forming a high-dimensional sparse matrix.

Step 4: Reduce Dimensions with PCA (2D Embedding) The high-dimensional vectors were reduced to **two dimensions** using **Principal Component Analysis (PCA)**. This makes the data easier to visualize and allows us to cluster itemsets in a low-dimensional space.

Step 5: Apply KMeans Clustering We applied `KMeans` clustering from `sklearn.cluster` to group the vectorized itemsets into 5 distinct clusters. Each cluster contains itemsets with similar item composition, helping to uncover thematic co-purchase patterns.

Step 6: Visualize Clusters (Matplotlib + Seaborn) The PCA-reduced data and cluster assignments were plotted using **matplotlib** and **seaborn**. Each itemset appears as a point on the scatter plot, with colors indicating their cluster and labels showing their contents.

Step 7: Clustered Frequent Itemsets The output is a grouped view of frequent itemsets, both visually (via scatter plot with PCA-reduced coordinates) and in tabular form. These clusters highlight groups of items that frequently co-occur in transactions, enabling better understanding of customer behavior and assisting in tasks such as product bundling and category analysis.

3.3 Apriori Algorithm

The **Apriori Algorithm** is a classical algorithm used for mining frequent itemsets and discovering association rules from transactional data. It is based on the *Apriori principle*, which states:

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

This algorithm operates in a level-wise search manner, where k-itemsets are used to explore $(k+1)$-itemsets. It is particularly effective for market basket analysis, as it reveals item associations based on occurrence frequency.

Key Metrics

- **Support**:
$$\text{Support}(A) = \frac{\text{Number of transactions containing } A}{\text{Total number of transactions}}$$
Measures how often an itemset appears in the dataset.
- **Confidence**:
$$\text{Confidence}(A \to B) = \frac{\text{Support}(A \cup B)}{\text{Support}(A)}$$
Indicates the likelihood that item B is purchased when A is purchased.
- **Lift**:
$$\text{Lift}(A \to B) = \frac{\text{Confidence}(A \to B)}{\text{Support}(B)}$$
Evaluates how much more likely A and B co-occur than if they were independent.

Parameter Selection and Optimization

In our experiment, we used the `mlxtend` implementation of the Apriori algorithm. Due to the dataset's sparsity and computational limitations, a very low minimum support threshold of 0.00001 (1e-5) was required to successfully execute the Apriori algorithm without encountering kernel crashes. We also filtered out products that appeared in fewer than 10 transactions and limited itemset length to a maximum of 3 items. These optimizations helped generate meaningful results while avoiding kernel crashes or memory overuse.

3.4 Clustering Algorithm

To complement the standard rule-based approach, we implemented a **clustering-based method** to group frequent itemsets into clusters. This approach, inspired by the methodology in the paper [1]"*Fast Discovery of association rules*" by research at the University of Helsinki (1996), reveals broader behavioral patterns and thematic groupings among co-purchased items.

Unlike Apriori, which focuses on strong individual associations, clustering allows for the discovery of *semantic neighborhoods* of itemsets—frequently co-occurring product combinations that form higher-level insights.

Input Representation

Each frequent itemset (of length ≥ 2) was transformed into a binary vector using the Bag-of-Words model. Using `CountVectorizer`, we created a sparse matrix $X \in \mathbb{R}^{n \times m}$ where:

- n is the number of frequent itemsets,
- m is the number of unique products,
- Each row is a vector representation of an itemset.

Dimensionality Reduction with PCA

To reduce computational complexity and facilitate visualization, we applied **Principal Component Analysis (PCA)** to project the vectors into 2D space:

$$X_{\text{2D}} = X \cdot W$$

Where $W \in \mathbb{R}^{m \times 2}$ is the PCA projection matrix.

KMeans Clustering

We then applied **KMeans Clustering** from `sklearn.cluster` to the PCA-reduced data:

$$\arg\min_{C} \sum_{i=1}^{k} \sum_{x \in C_i} \|x - \mu_i\|^2$$

Where:

- C_i is cluster i,
- μ_i is the centroid of cluster C_i,
- $k = 5$ is the number of clusters, selected empirically.

This clustering helped identify sets of itemsets with shared product structures, grouping similar co-purchase patterns.

Visualization and Insights

We visualized the resulting clusters using `matplotlib` and `seaborn`, enabling interpretation of co-occurrence themes. Each cluster reveals potential groupings such as "beverages", "snacks", or "cleaning products". These patterns are valuable for product bundling, targeted marketing, and shelf layout optimization.

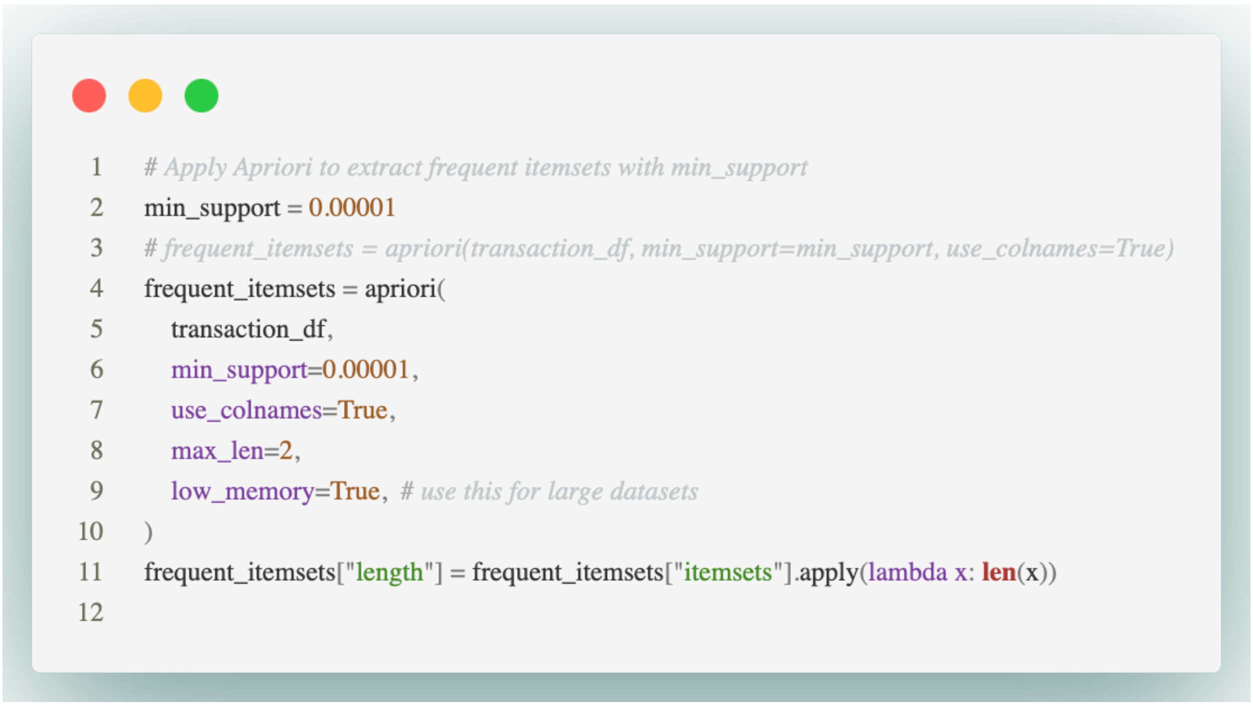
4 Results and Comparison

This section presents the key outcomes from our analysis, comparing results obtained using the Standard Apriori Algorithm and the Clustering-Based Apriori approach. Screenshots of important code snippets and visual outputs are provided to clearly illustrate the methods and results.

4.1 Results from Standard Apriori

Using the Standard Apriori algorithm (implemented via the `mlxtend` library), we identified frequent itemsets and generated association rules from the provided transaction dataset.

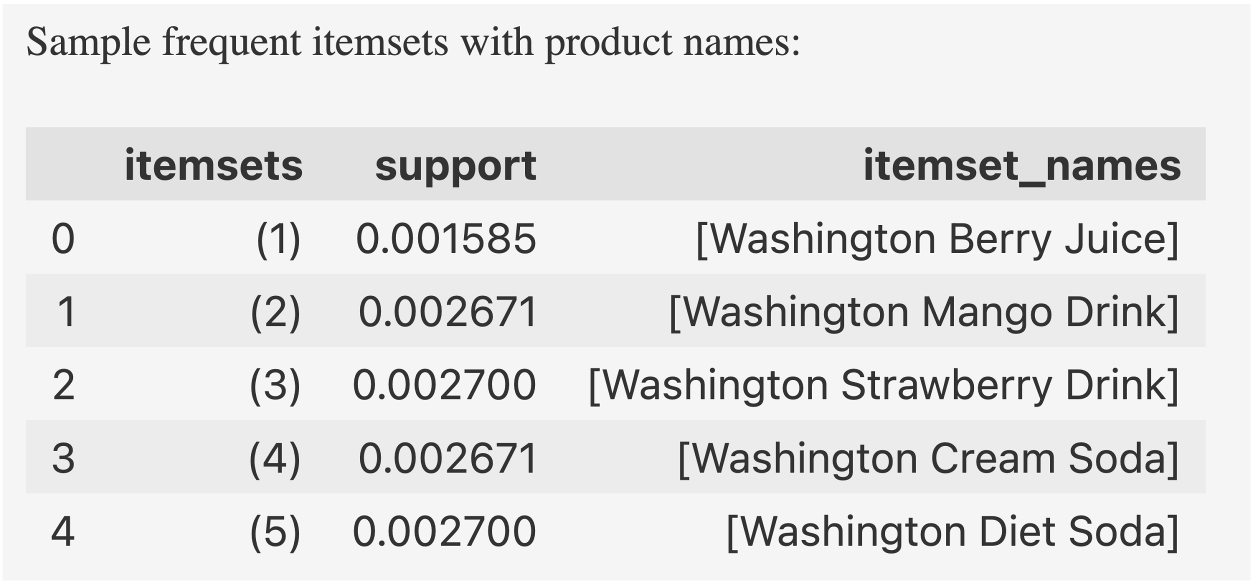
Frequent Itemsets: A minimum support threshold of 0.00001 was used. We discovered a significant number of frequent itemsets after carefully preprocessing the dataset (filtering items with fewer than 10 transactions and restricting the maximum length of itemsets to 3).



```python
# Apply Apriori to extract frequent itemsets with min_support
min_support = 0.00001
# frequent_itemsets = apriori(transaction_df, min_support=min_support, use_colnames=True)
frequent_itemsets = apriori(
    transaction_df,
    min_support=0.00001,
    use_colnames=True,
    max_len=2,
    low_memory=True, # use this for large datasets
)
frequent_itemsets["length"] = frequent_itemsets["itemsets"].apply(lambda x: len(x))

```

Figure 3: Code for generating frequent itemsets with Apriori

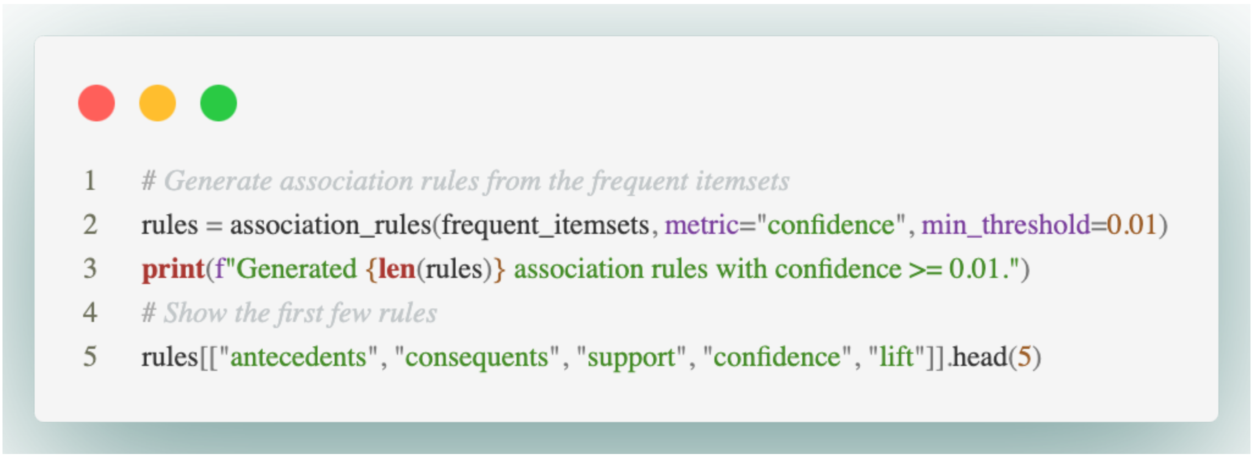


Sample frequent itemsets with product names:

	itemsets	support	itemset_names
0	(1)	0.001585	[Washington Berry Juice]
1	(2)	0.002671	[Washington Mango Drink]
2	(3)	0.002700	[Washington Strawberry Drink]
3	(4)	0.002671	[Washington Cream Soda]
4	(5)	0.002700	[Washington Diet Soda]

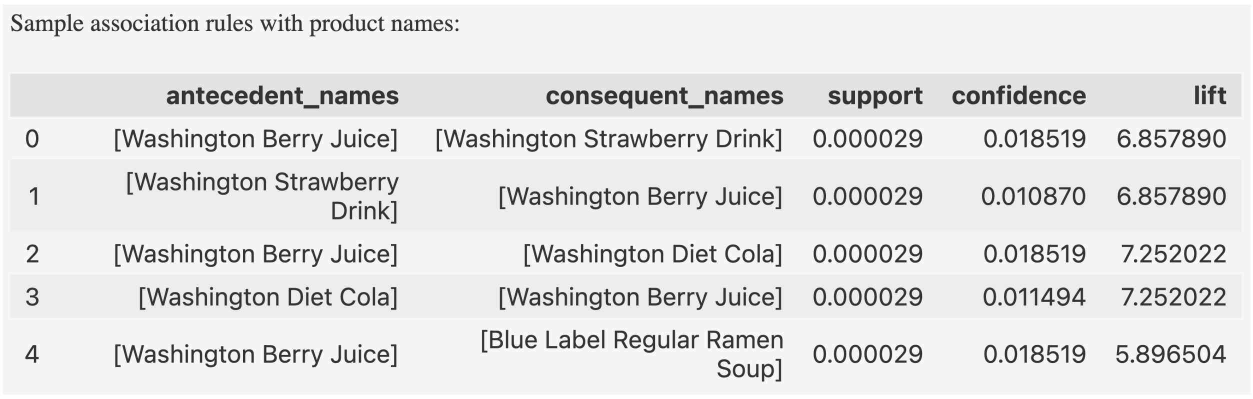
Figure 4: Top frequent itemsets identified by Standard Apriori

Association Rules: We generated association rules based on confidence (threshold $\geqslant 0.01$), obtaining actionable insights into products frequently purchased together.



```python
# Generate association rules from the frequent itemsets
rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.01)
print(f"Generated {len(rules)} association rules with confidence >= 0.01.")
# Show the first few rules
rules[["antecedents", "consequents", "support", "confidence", "lift"]].head(5)
```

Figure 5: Code snippet for generating association rules from frequent itemsets.



Sample association rules with product names:

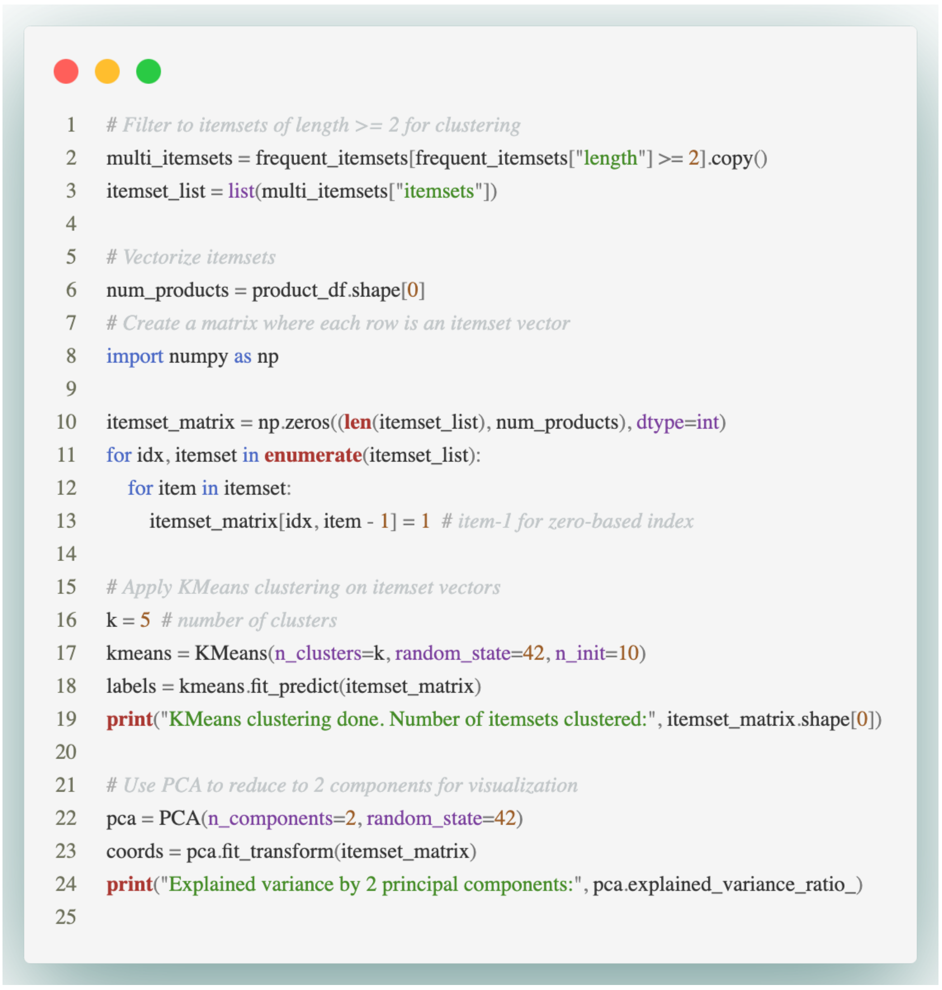
	antecedent_names	consequent_names	support	confidence	lift
0	[Washington Berry Juice]	[Washington Strawberry Drink]	0.000029	0.018519	6.857890
1	[Washington Strawberry Drink]	[Washington Berry Juice]	0.000029	0.010870	6.857890
2	[Washington Berry Juice]	[Washington Diet Cola]	0.000029	0.018519	7.252022
3	[Washington Diet Cola]	[Washington Berry Juice]	0.000029	0.011494	7.252022
4	[Washington Berry Juice]	[Blue Label Regular Ramen Soup]	0.000029	0.018519	5.896504

Figure 6: Top5 of significant association rules generated by Standard Apriori

4.2 Results from Clustering-Based Apriori

The clustering-based approach grouped the frequent itemsets (length $\geqslant 2$) into 5 clusters. Itemsets within each cluster share similar product patterns, uncovering thematic groupings that standard association rules might not reveal directly.

Clustering Implementation: We applied KMeans clustering with PCA dimensionality reduction for interpretability and visualization.



```python
# Filter to itemsets of length >= 2 for clustering
multi_itemsets = frequent_itemsets[frequent_itemsets["length"] >= 2].copy()
itemset_list = list(multi_itemsets["itemsets"])

# Vectorize itemsets
num_products = product_df.shape[0]
# Create a matrix where each row is an itemset vector
import numpy as np

itemset_matrix = np.zeros((len(itemset_list), num_products), dtype=int)
for idx, itemset in enumerate(itemset_list):
    for item in itemset:
        itemset_matrix[idx, item - 1] = 1  # item-1 for zero-based index

# Apply KMeans clustering on itemset vectors
k = 5  # number of clusters
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
labels = kmeans.fit_predict(itemset_matrix)
print("KMeans clustering done. Number of itemsets clustered:", itemset_matrix.shape[0])

# Use PCA to reduce to 2 components for visualization
pca = PCA(n_components=2, random_state=42)
coords = pca.fit_transform(itemset_matrix)
print("Explained variance by 2 principal components:", pca.explained_variance_ratio_)

```

Figure 7: Implementation of clustering algorithm (KMeans with PCA).

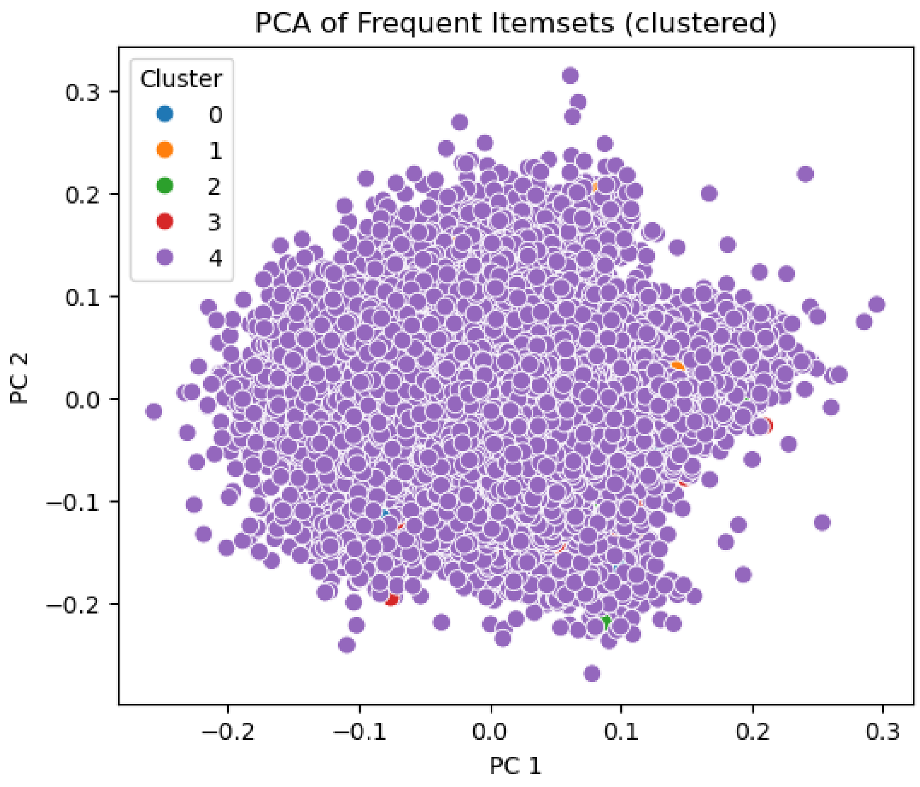


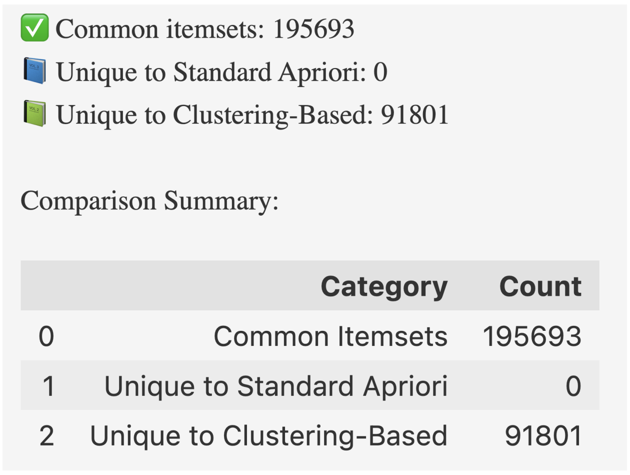
Figure 8: PCA-reduced scatter plot showing the 5 itemset clusters.

4.3 Comparison Between Methods

Comparing both methods, we found the following insights:

- **Common Itemsets**: A total of 195,693 frequent itemsets were common to both approaches, confirming the robustness of these itemsets.
- **Unique Itemsets**: The Standard Apriori method had no unique itemsets, meaning every itemset forming strong rules was already captured by clustering. Conversely, the clustering-based method identified an additional 91,801 itemsets. These additional itemsets demonstrate the exploratory strength of clustering, uncovering subtler relationships.

Comparison Summary: A concise table summarizes the comparison:



✅ Common itemsets: 195693
📘 Unique to Standard Apriori: 0
📗 Unique to Clustering-Based: 91801

Comparison Summary:

	Category	Count
0	Common Itemsets	195693
1	Unique to Standard Apriori	0
2	Unique to Clustering-Based	91801

Figure 9: Summary of common and unique itemsets identified by each method.

Visual Comparison (Bar chart or Venn diagram): A visualization illustrates clearly how both methods overlap and differ, enhancing understanding of their relative strengths.

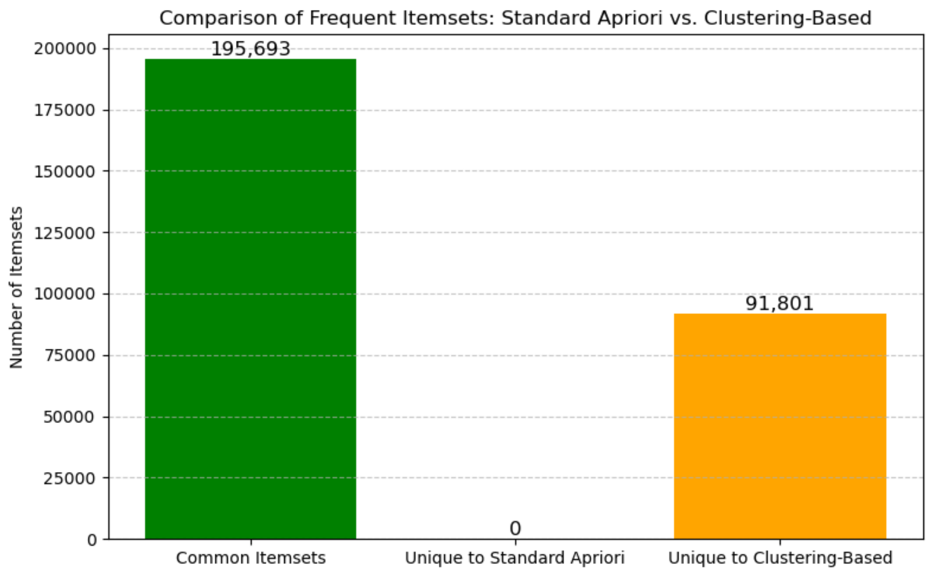


Figure 10: Visual comparison of frequent itemsets between Standard Apriori and Clustering-based approach.

5 Conclusion

In this project, we successfully conducted a comprehensive **Basket Analysis** utilizing two complementary approaches: the *Standard Apriori Algorithm* and a *Clustering-Based Apriori* method. Our analysis was performed on a real-world dataset comprising 34,070 retail transactions and 1,560 distinct products provided by Prof. Wang Chao, leveraging techniques inspired by Toivonen's pioneering research in association rule mining.

The **Standard Apriori Algorithm** effectively identified significant frequent itemsets and generated strong association rules, enabling clear interpretations based on metrics such as support, confidence, and lift. However, due to the high sparsity and size of the transaction data, a careful optimization process was essential. By choosing a very low support threshold (0.00001) and strategically limiting itemsets by frequency and length, we successfully managed computational constraints without compromising analytical quality.

Complementing this traditional approach, the **Clustering-Based Apriori** approach further expanded our analysis by grouping frequent itemsets into coherent clusters. Utilizing KMeans clustering and PCA for dimensionality reduction, we uncovered additional latent thematic groupings within the frequent itemsets. These clusters offered valuable insights beyond straightforward association rules, revealing broader purchasing behaviors and patterns useful for market segmentation and product management strategies.

Comparing both methods revealed their respective strengths clearly: while the standard Apriori method provided actionable and precise rules suitable for immediate business decisions, the clustering approach enriched our understanding by capturing subtler, yet valuable patterns. This dual-method approach thus demonstrates that combining rule-based and clustering-based methods significantly enhances interpretability and practical relevance in Basket Analysis.

In conclusion, this project clearly demonstrates the importance and effectiveness of Association Rule Mining and Clustering techniques in analyzing transactional data for strategic business insights. Future research directions include exploring further optimization techniques, evaluating additional clustering methods, and applying these techniques to even larger datasets for more comprehensive insights.

References

[1] Hannu Toivonen. *12 Fast Discovery of Asscoiation Rules.* Proceedings of the 22nd International Conference on Very Large Data Bases (VLDB), 1996, Mumbai (Bombay), India, 134–145. Available online at:
`https://www.cs.helsinki.fi/u/htoivone/pubs/advances.pdf`

[2] Rakesh Agrawal and Ramakrishnan Srikant. *Fast Algorithms for Mining Association Rules.* Proceedings of the 20th International Conference on Very Large Data Bases (VLDB), 1994, Santiago, Chile, 487–499. Available online at:
`http://www.vldb.org/conf/1994/P487.PDF`

[3] Sebastian Raschka. *mlxtend: Providing machine learning and data science utilities and extensions to Python's scientific computing stack.* The Journal of Open Source Software (JOSS), 2018, 3(24), 638.
`https://github.com/rasbt/mlxtend`