AOI Based Characterization and Discrimination of Amazon Reviews

Data Mining Project Report

Subject Code: CS6001

Submitted by

Akshaya Srikrishna
2022103065

Krishnendu M R
2022103081

Submitted to

Prof. Dr. S. Chitrakala

Department of Computer Science and Engineering
College of Engineering Guindy

Contents

List of Figures

1 Abstract

This study applies Attribute-Oriented Induction (AOI) to extract characterization and discriminant rules from Amazon 2023 Fashion Reviews. Characterization rules summarize patterns in review length, sentiment, helpfulness, and verification status, highlighting common behaviors in different review types. Discriminant rules identify key differences, such as sentiment-based helpfulness trends and rating variations between verified and unverified purchases. These insights improve understanding of consumer sentiment, review credibility, and engagement patterns, aiding data-driven decision-making in e-commerce.

2 Introduction

2.1 Background

The rapid rise of e-commerce has resulted in vast amounts of customer reviews, making it difficult to extract meaningful insights. **Attribute-Oriented Induction (AOI)** simplifies this process by generalizing raw data into higher-level concepts, facilitating efficient pattern discovery. This study applies AOI to Amazon 2023 Fashion Reviews, focusing on **data characterization and discrimination** to reveal consumer behavior trends. Summarizing review patterns and distinguishing differences improves review-based decision-making.

2.2 Problem Statement

Unstructured review data is high-dimensional, redundant, and inconsistent, making pattern discovery challenging. Existing sentiment analysis methods struggle to differentiate between **Frequent vs Infrequent reviews** or capture key characteristics like **length and helpfulness**. This study applies **AOI** to generalize, characterize, and discriminate review attributes, generating **characterization rules** for summarizing trends and **discriminant rules** for identifying significant differences. The objective is to enhance review analysis by uncovering meaningful patterns in consumer feedback.

2.3 Objectives

This study aims to:

- **Generalize Review Data:** Simplify raw Amazon Fashion review attributes by categorizing continuous data into discrete bins, making it easier to analyze.
- **Characterize Data:** Summarize key trends in sentiment, word frequency, and verification patterns across the reviews.
- **Generate Characterization Rules:** Extract high-level rules that define common characteristics within the reviews.
- **Discriminate Data:** Identify statistical differences between contrasting classes (e.g., reviews from different seasons) to uncover key trends.
- **Generate Discriminant Rules:** Create rules with weights to highlight the most significant differentiating factors in the reviews.

3 Literature Review

Data mining techniques play a crucial role in extracting meaningful insights from large datasets, enabling businesses and researchers to uncover patterns, relationships, and trends. One such approach, Attribute-Oriented Induction (AOI), is widely utilized for data generalization, characterization, and discrimination.

Han et al.'s book on data mining [1] explores various techniques, including AOI, which simplifies data by transforming raw attributes into generalized forms while preserving essential trends. This methodology has been applied across multiple domains, demonstrating its effectiveness in summarizing data and identifying key patterns.

A significant amount of research has focused on sentiment analysis using machine learning. The study "Amazon Product Reviews Sentimental Analysis using Machine Learning" [2] investigates the effectiveness of Naïve Bayes and Support Vector Machine (SVM) classifiers in classifying sentiments in Amazon product reviews. By employing key preprocessing techniques such as tokenization, stop-word removal, stemming, and lemmatization, the study enhances the quality of textual data. Feature extraction methods like Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) further improve model accuracy, with Naïve Bayes achieving 80% accuracy and SVM 76%. While this research establishes machine learning as a viable tool for sentiment classification, it primarily focuses on classification rather than deeper data characterization and generalization.

Our research builds on these prior studies by applying AOI-based characterization and discrimination techniques to gain deeper insights into review patterns. AOI facilitates the transformation of raw review attributes into generalized forms, aiding in the extraction of high-level patterns and distinctions within datasets. By leveraging AOI, we aim to provide a more structured and interpretable analysis of consumer reviews, offering insights beyond simple sentiment classification.

4 Dataset Description

The dataset used was sourced from Hugging Face under the repository `McAuley-Lab/Amazon-Reviews-2023`, specifically the `raw_review_Amazon_Fashion` subset. It contains around 2.5 million rows and includes customer reviews for fashion-related products on Amazon.

Each review is represented with fields such as `rating` (numeric rating from 1 to 5), `title` (short summary of the review), `text` (full review text), `images` (list of associated images if any), `asin` (Amazon Standard Identification Number), `parent_asin` (parent product ID), `user_id` (unique user identifier), `timestamp` (time of review in epoch milliseconds), `helpful_vote` (number of helpful votes received), and `verified_purchase` (boolean indicating whether the purchase was verified).


```
{'rating': 5.0,
 'title': 'Pretty locket',
 'text': 'I think this locket is really pretty. The inside back is a solid silver
depression and the front is a dome that is not solid (knotted). You
could use it to store a small photo, lock of hair, etc but I use it when I
need to carry medication with me. Closes securely. High quality & very
pretty.',
 'images': [],
 'asin': 'B00LOPVX74',
 'parent_asin': 'B00LOPVX74',
 'user_id': 'AGBFYI2DDIKXC5Y4FARTYDTQBMFQ',
 'timestamp': 1578528394489,
 'helpful_vote': 3,
 'verified_purchase': True}
```


Figure 1: Sample Record

5 Proposed Architecture

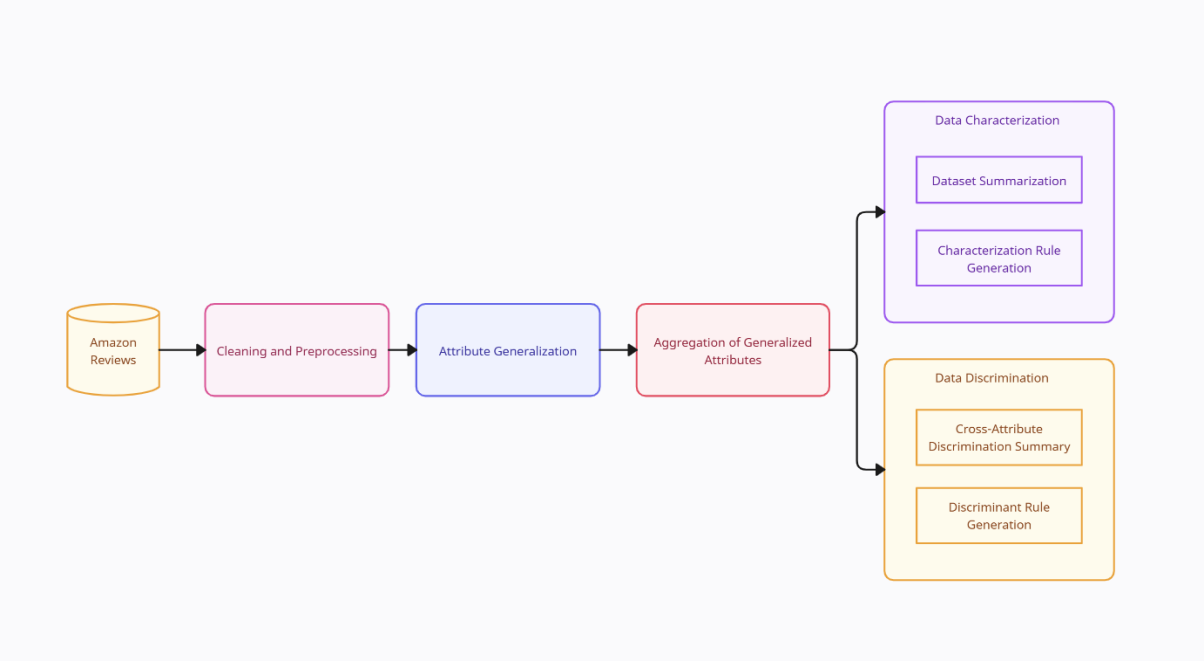

Figure 2: Architecture Diagram

6 Module-wise Description

6.1 Amazon Reviews Dataset

The dataset consists of 2023 Amazon Fashion reviews, which include textual reviews, ratings, verified purchase status, and helpful votes. This raw data serves as the input for further processing.

6.2 Cleaning and Preprocessing

The data preprocessing stage ensures consistency and improves analytical accuracy. It involves removing stop words, punctuation, and special characters, followed by tokenizing the text into individual words. Additionally, missing values are handled, and the data format is standardized to maintain uniformity across the dataset.

6.3 Attribute Generalization

To facilitate effective pattern analysis, key attributes are generalized into categorical labels. Review length is classified as Short, Medium, or Long, while helpfulness is categorized as Helpful or Not Helpful based on a threshold of user votes. Similarly, purchase verification status is distinguished as Verified or Unverified, enabling structured evaluation of review patterns.

6.4 Aggregation of Generalized Attributes

In this stage, the generalized attributes are grouped to form a structured dataset, facilitating both characterization and discrimination. By aggregating attributes like review length, helpfulness, and purchase verification, the dataset becomes more interpretable, allowing for the extraction of meaningful patterns and distinctions in customer reviews.

6.5 Data Characterization

The **Data Characterization** module aims to extract meaningful insights from the dataset by summarizing and analyzing general patterns. This process helps in understanding the distribution of various review attributes, including sentiment polarity, word count, helpfulness, and verification status. By aggregating data into predefined categories, characterization enables the identification of trends that define different groups of reviews.

One key aspect of data characterization is distribution analysis, where we study how reviews are spread across sentiment types (positive, negative, and neutral), length categories (short, medium, long), and helpfulness ratings (highly helpful vs. not helpful). This breakdown helps in identifying which types of reviews dominate the dataset and what patterns emerge across different product categories.

Another important function is **pattern rule extraction**, which involves identifying frequently occurring words or phrases within each category. For example, highly positive reviews may frequently contain words such as "comfortable" or "perfect," while negative reviews may focus on issues like "poor quality" or "wrong size." These insights can be structured into **characterization rules** that define common review behaviors, allowing businesses to better understand customer perceptions. By summarizing key patterns, data characterization provides a **structured representation** of user feedback that can guide decision-making.

6.6 Data Discrimination

The **Data Discrimination** module focuses on identifying significant differences between various review categories. Unlike characterization, which summarizes general patterns, discrimination highlights specific attributes that set different groups apart. This analysis is particularly useful for comparing classes, such as purchasing season is summer or winter.

The first stage, **Discriminant Rule Generation**, involves formulating logical rules that describe these differences. These rules provide actionable insights into customer review behavior, enabling businesses to refine marketing strategies, enhance product descriptions, or improve customer engagement.

The second stage, **Cross-Attribute Summary**, examines statistical variations between different classes. For example, it can determine whether frequent reviewers are more likely to be marked as helpful compared to infrequent reviewers. This analysis helps uncover meaningful distinctions within the dataset. Data discrimination ensures that relevant disparities are captured, enabling more informed decision-making and improving the interpretability of review patterns.

7 Methodology

7.1 Data Preprocessing

To ensure that the textual data was structured and suitable for analysis, we applied various preprocessing techniques using Python. The preprocessing steps included:

- **Tokenization**: Splitting the text into individual words or tokens for further analysis.
- **Stopword Removal**: Eliminating common words (such as "the," "is," "and") that do not contribute to sentiment analysis.
- **Text Normalization**: Converting all text to lowercase and performing stemming and lemmatization to reduce words to their base or root form.

7.2 Generalization in Attribute Oriented Induction

Generalization is a crucial preprocessing step in Attribute-Oriented Induction (AOI) that simplifies data by replacing specific values with higher-level concepts. Before performing characterization and discrimination, raw data often contains redundant or overly detailed information, making pattern extraction difficult. By generalizing attributes such as review length, sentiment, and helpfulness votes into broader categories (e.g., short/medium/long reviews or high/low helpfulness), we reduce data complexity while retaining essential patterns. In our study, we generalize Amazon 2023 Fashion Reviews by grouping numerical attributes into categorical ranges and filtering out highly specific words. See algorithm **??**.

7.3 Aggregration of generalized attributes

This involves grouping the dataset based on generalized attributes—rating, review length, helpful votes, and verification status—to identify patterns and relationships within reviews. By aggregating key metrics such as review count, average rating, and average helpful votes, the approach enables characterization of different review behaviors.

7.4 Data Characterization

The following functionalities were implemented to characterize the *Amazon 2024 Fashion Reviews* dataset:

7.4.1 Sentiment-Specific Word Distributions

Reviews were grouped into **Positive**, **Negative**, and **Neutral** sentiment categories. Within each group, Term Frequency-Inverse Document Frequency (TF-IDF) was applied to extract the top sentiment-specific words. Neutral and contradictory terms were filtered out to ensure clarity in sentiment representation. Bar plots were used to visualize the most characteristic words per sentiment.

Algorithm 1 Sentiment-Specific Word Extraction

Require: Preprocessed reviews with sentiment labels (Positive, Negative, Neutral)
Ensure: Sentiment-specific words for each category

```
for each sentiment category in {Positive, Negative, Neutral} do
    for each review with that sentiment do
        Apply TF-IDF to extract sentiment-specific words
        Filter out neutral and contradictory terms
    end for
    Visualize the most characteristic words using bar plots
end for
Return categorized sentiment-specific words
```

7.4.2 Overall Sentiment Distribution

The sentiment distribution across the dataset was computed using the generalized_rating field, which categorizes reviews into **Positive**, **Negative**, and **Neutral**. A pie chart was then created to visualize the proportions of each sentiment label, providing a clear representation of overall customer sentiment.

Algorithm 2 Sentiment Distribution Calculation

Require: Dataset with sentiment labels (generalized_rating)
Ensure: Sentiment distribution visualization
1: Initialize sentiment counters for **Positive**, **Negative**, **Neutral**
2: **for** each review in dataset **do**
3: Increment corresponding sentiment counter based on generalized_rating
4: **end for**
5: Compute the sentiment distribution percentages
6: Create a pie chart visualization for sentiment distribution

7.4.3 Verified and Unverified Review Counts

The dataset was characterized based on the **verified_purchase** attribute. The number of **verified** and **unverified** reviews was calculated, and this distribution was visualized using bar plots. This helps in understanding the proportion of reviews that come from verified versus unverified purchases.

Algorithm 3 Verification-Based Review Count

Require: Dataset with **verified_purchase** attribute
Ensure: Count of verified and unverified reviews
1: **for** each review in dataset **do**
2: **if** review is verified **then**
3: Increment the verified reviews counter
4: **else**
5: Increment the unverified reviews counter
6: **end if**
7: **end for**
8: Visualize the count of verified and unverified reviews using bar plots

7.4.4 Sentiment Breakdown within Metadata Groups

For both **verified** and **unverified** reviews, the frequency of each sentiment label (**Positive**, **Negative**, and **Neutral**) was computed. The sentiment distributions within each group were plotted separately to highlight any differences in sentiment expression between verified and unverified reviews.

Algorithm 4 Sentiment Breakdown within Verified and Unverified Groups

Require: Dataset with **verified_purchase** and sentiment labels
Ensure: Sentiment breakdown for verified and unverified reviews
1: **for** each review in dataset **do**
2: **if** review is verified **then**
3: Increment sentiment counter for verified reviews
4: **else**
5: Increment sentiment counter for unverified reviews
6: **end if**
7: **end for**
8: Plot sentiment distributions separately for verified and unverified reviews

7.4.5 Helpfulness Metrics per Sentiment

The average number of helpful votes was calculated for both **Positive** and **Negative** reviews. These values were used to describe the level of engagement and helpfulness across sentiment categories. The average helpfulness per sentiment was visualized using bar plots to highlight engagement differences.

Algorithm 5 Helpfulness Metrics Calculation

Require: Dataset with helpful votes and sentiment labels
Ensure: Average helpful votes per sentiment category
1: **for** each sentiment category in {**Positive**, **Negative**} **do**
2: **for** each review with that sentiment **do**
3: Compute helpful votes for that review
4: **end for**
5: Calculate the average helpful votes for the sentiment category
6: Visualize the average helpful votes using bar plots
7: **end for**

7.4.6 Characterization Rule Extraction

Finally, we extract high-level **characterization rules** that summarize patterns in reviews. These rules describe common characteristics of positive, negative, and neutral reviews under verified and unverifed reviews, such as typical review length and helpfulness. By formalizing these patterns, we enhance the interpretability of the dataset and provide valuable insights into consumer behavior.

Algorithm 6 Characterization Rule Extraction

Require: Generalized dataset with categorized attributes
Ensure: Characterization rules for different review types
1: **for** each review category in {Positive, Negative, Neutral under Verified, Unverified} **do**
2: Compute most frequent review length
3: Compute average helpfulness percentage
4: Identify common word usage patterns
5: Store characterization rules
6: **end for**
7: Return characterization rules

7.5 Discriminative Analysis

7.5.1 T-weight and D-weight Formulae

T-weight (Typicality Weight):

$$T\text{-weight}(q_a, C_j) = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{n} \text{count}(q_i \in C_j)} \tag{1}$$

- q_a: A generalized tuple (or rule) being considered.
- C_j: The target class (e.g., "Frequent Reviewer").
- $\text{count}(q_a \in C_j)$: Number of tuples in class C_j that match the rule q_a.
- $\sum_{i=1}^{n} \text{count}(q_i \in C_j)$: Total number of tuples in C_j across all generalized tuples.

Interpretation: Measures how typical or representative the rule q_a is within the class C_j. Higher T-weight indicates higher typicality.

D-weight (Discriminability Weight):

$$D\text{-weight}(q_a, C_j) = \frac{\text{count}(q_a \in C_j)}{\sum_{k=1}^{m} \text{count}(q_a \in C_k)} \tag{2}$$

- q_a: The same generalized tuple (or rule).
- C_j: The target class.
- count$(q_a \in C_j)$: Number of tuples in class C_j matching q_a.
- C_k: Any class (where $k = 1$ to m), including C_j and other classes.
- $\sum_{k=1}^{m}$ count$(q_a \in C_k)$: Total number of times q_a appears across all classes.

Interpretation: Measures how discriminative the rule q_a is toward class C_j. A higher D-weight means q_a is mostly associated with C_j and less frequent in others.

7.5.2 Frequent vs. Infrequent Reviewers

Objective: The goal is to study the differences in review behaviors between frequent reviewers and infrequent reviewers. Frequent reviewers are defined as users who have written more than 10 reviews, while infrequent reviewers are those who have written 10 or fewer reviews.

Categorization:

- **Frequent Reviewers:** Users with more than 10 reviews.
- **Infrequent Reviewers:** Users with 10 or fewer reviews.

We aim to identify patterns in sentiment, review length, helpfulness, and rating distributions between these two categories.

Algorithm: Discriminative Rules for Reviewer Frequency

Algorithm 7 Discriminative Rules for Reviewer Frequency

Input: Dataset D
Split D into two subsets:
  $D_{\text{frequent}} \leftarrow$ reviews from users with more than 10 reviews
  $D_{\text{infrequent}} \leftarrow$ reviews from users with 10 or fewer reviews
For each subset D_{frequent} and $D_{\text{infrequent}}$:
  Generalize data
  Aggregate attributes like average sentiment score or rating
  Mine frequent itemsets using Apriori algorithm
  Generate association rules from the frequent itemsets
  Compute weights:
    $T_{\text{Weight}} = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{n} \text{count}(q_i \in C_j)}$
    $D_{\text{Weight}} = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{m} \text{count}(q_a \in C_i)}$
  Filter rules based on high discriminability weights D_{Weight}
  Rank the rules by D_{Weight}
Output: Top-ranked discriminative rules

The algorithm begins by dividing the dataset based on reviewer activity into two distinct groups: frequent reviewers (users with more than 10 reviews) and infrequent reviewers (users with 10 or fewer reviews). Within each group, the data is first generalized by categorizing continuous variables such as sentiment scores and helpfulness into discrete bins (e.g., high, medium, low).
Next, the Apriori algorithm is applied to mine frequent itemsets that represent commonly co-occurring attribute-value pairs. From these itemsets, association rules are generated to reveal underlying relationships. Each rule is evaluated using two metrics: typicality weight and discriminative weight, which indicate rule relevance and uniqueness. Finally, rules with high discriminative weights are filtered and ranked, highlighting patterns that distinctly characterize each group of reviewers. The output is a ranked list of discriminative rules that provide insights into behavioral differences based on review frequency.

7.5.3 Fashion Category-Based Patterns

Objective: We aim to analyze the review patterns across different fashion categories, such as Accessories, Activewear, Bottoms, Dresses, etc. The goal is to identify if certain trends in sentiment, rating, and review types are unique to specific categories.

Categorization: Reviews are categorized by fashion category (e.g., Accessories, Activewear, Dresses) to compare how review patterns differ by product type.

Algorithm: Category-Specific Rule Extraction In this algorithm, the data frame is first divided by fashion categories such as Accessories, Activewear, Kidswear, etc. For each category-specific subset, the data is generalized to transform continuous attributes like rating or sentiment into categorical ranges, such as "positive," "neutral," or "negative." Frequent item-sets are then mined using the Apriori algorithm, identifying co-occurring features that are prominent within each category. These item-sets are used to derive association rules, which are scored using both support-based and lift-based weighting schemes to evaluate their frequency and discriminative power. Rules are then filtered and ranked based on their discriminative weight, emphasizing category-specific behavioral or opinion trends. The final output includes top-ranked rules for each fashion category, offering actionable insights for targeted product and marketing strategies.

Algorithm 8 Category-Specific Rule Extraction

Input: Dataset D
Split D by fashion category:
  $D_{\text{Accessories}}, D_{\text{Kidswear}}, \ldots$
For each category:
  Generalize data
  Mine frequent itemsets using Apriori algorithm
  Generate association rules from the frequent itemsets
  Compute weights:
    $T_{\text{Weight}} = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{n} \text{count}(q_i \in C_j)}$
    $D_{\text{Weight}} = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{m} \text{count}(q_a \in C_i)}$
  Evaluate and rank rules based on D_{Weight}
Output: Top-ranked rules for each category

7.5.4 Pre-COVID vs. Post-COVID Trends

Objective: We are studying how the pandemic has influenced user review behavior by comparing reviews written before and after the COVID-19 pandemic. We aim to explore how sentiments, ratings, and review content may have shifted during this time.

Categorization:

- **Pre-COVID:** Reviews written before the start of the COVID-19 pandemic.
- **Post-COVID:** Reviews written after the onset of the COVID-19 pandemic.

We aim to detect changes in review behavior, such as shifts in sentiment or product focus, due to the pandemic's impact on consumer habits.

Algorithm: Temporal Pattern Discriminative Mining (Pre-COVID vs. Post-COVID)

This algorithm aims to uncover temporal shifts in review patterns by segmenting the dataset into pre-COVID (before March 2020) and post-COVID (after July 2020) subsets. This temporal division captures how customer behavior or sentiment may have evolved in response to global events. Within each subset, the data undergoes generalization, where continuous features like sentiment, rating, and helpfulness scores are mapped to categorical levels. Frequent item-sets are then mined using the Apriori algorithm to discover attribute combinations that are commonly seen in each period. From these, association rules are generated to expose relationships unique to the pre- or post-COVID era. Two weights are calculated—typicality and discriminative to assess how frequent and exclusive the patterns are. Rules with high discriminative weights are particularly interesting as they highlight behavioral changes that are distinctive to either time frame. The final step involves comparing these rules across periods to extract insights on how consumer feedback has shifted, enabling data-driven adaptation of business strategies to evolving consumer sentiment.

Algorithm 9 Temporal Pattern Discriminative Mining

Input: Dataset D
Split D into two subsets:
  $D_{\text{pre}} \leftarrow$ reviews before March 2020 (Pre-COVID)
  $D_{\text{post}} \leftarrow$ reviews after July 2020 (Post-COVID)
For each subset D_{pre} and D_{post}:
  Generalize the data
  Mine frequent itemsets using Apriori algorithm
  Generate association rules from the frequent itemsets
  Compute weights:
    $T_{\text{Weight}} = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{n} \text{count}(q_i \in C_j)}$
    $D_{\text{Weight}} = \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{m} \text{count}(q_a \in C_i)}$
  Compare the rules between pre-COVID and post-COVID
  Filter rules based on high D_{Weight}
Output: Most discriminative rules between pre-COVID and post-COVID

7.5.5 Seasonal Patterns (Summer vs. Winter)

Algorithm: Seasonal Rule Mining (Summer vs. Winter)

Algorithm 10 Seasonal Rule Mining

Input: Dataset D
Split D into two subsets:
  $D_{\text{summer}} \leftarrow$ reviews written between May and August
  $D_{\text{winter}} \leftarrow$ reviews written between November and February
for $D_{\text{season}} \in \{D_{\text{summer}}, D_{\text{winter}}\}$ **do**
  Generalize data
  Mine frequent itemsets using the Apriori algorithm
  Generate association rules from the frequent itemsets
  Compute the following weights for each rule:
    Topic Weight (T-Weight):
    T-Weight $= \frac{\text{count}(q_a \in C_j)}{\sum_{i=1}^{n} \text{count}(q_i \in C_j)}$
    Distinctiveness Weight (D-Weight):
    D-Weight $= \frac{\text{count}(q_a \in C_j)}{\sum_{k=1}^{m} \text{count}(q_a \in C_k)}$
  Rank and select the most distinctive rules using D-Weight
end for
Output: Most distinctive rules for summer and winter seasons

Objective: This analysis focuses on studying how seasonal changes influence user review behavior. Specifically, we compare reviews during the summer and winter months to identify seasonal patterns in sentiments and product preferences.

Categorization:

- **Summer:** Reviews written during the summer months (June, July, August).
- **Winter:** Reviews written during the winter months (December, January, February).

We aim to explore how product preferences, sentiments, and reviews vary with the seasons.
This algorithm identifies seasonal variations in customer reviews by splitting the dataset into two temporal windows: summer (May to August) and winter (November to February). Such seasonal segmentation allows the algorithm to uncover context-specific patterns that may not be visible in the overall dataset. Within each subset, generalization is performed by converting numeric attributes such as sentiment scores or helpfulness metrics into discrete bins (e.g., high, medium, low), simplifying the data for effective rule mining. The Apriori algorithm is applied next to mine frequent item sets that represent characteristic combinations of features for each season. These item sets are transformed into association rules, and both support and lift are calculated to determine how representative and unique each rule is. Rules are then filtered based on their discriminative weight and ranked to highlight those that most distinctly characterize seasonal behavior.

8 Results and Discussion

8.1 Generalization

Generalization has been done to the attributes resulting in the following outputs:
The generalized dataset reveals key distributions across sentiment, helpfulness, and length attributes before aggregation. Positive reviews form the majority, with 1,778,595 instances, while negative (476,873) and neutral (245,471) reviews are significantly fewer. In terms of helpfulness, most reviews (2,006,187) are labeled as "Not Helpful," with only 205,924 classified as "Helpful," indicating a large portion of reviews may not contribute meaningful insights. Regarding review length, shorter reviews dominate,

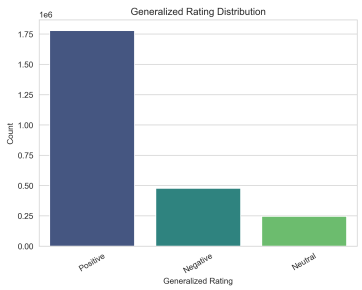

Figure 3: Rating

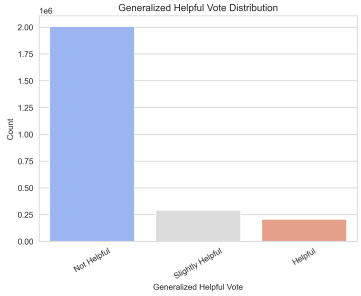

Figure 4: Helpful Votes

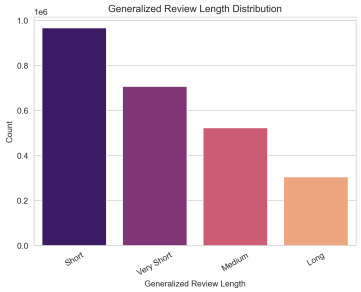

Figure 5: Review Length

with 966,686 classified as "Short" and 706,260 as "Very Short," whereas only 304,635 are considered "Long." These distributions set the foundation for further aggregation and deeper pattern analysis.

8.2 Characterization

The data characterization module of our project generated several insights and final rules based on the Amazon 2023 Fashion Reviews dataset. Our in-depth analysis uncovered patterns across sentiment, metadata, and user engagement metrics that shed light on how consumers interact with fashion products online.

8.2.1 Key Findings

To begin with, an examination of sentiment-specific word distributions revealed that positive reviews overwhelmingly contained favorable expressions such as `'love'`, `'great'`, and `'good'`. These terms were dominant across the dataset, reflecting a generally satisfied customer base. On the other hand, negative reviews highlighted issues related to sizing and material quality, frequently mentioning words like `'small'`, `'material'`, and `'cheap'`. Interestingly, neutral reviews displayed a linguistic overlap with both ends of the spectrum, using terms such as `'small'`, `'good'`, and `'nice'`, suggesting that some experiences, while not overtly polarized, still leaned toward either satisfaction or dissatisfaction.

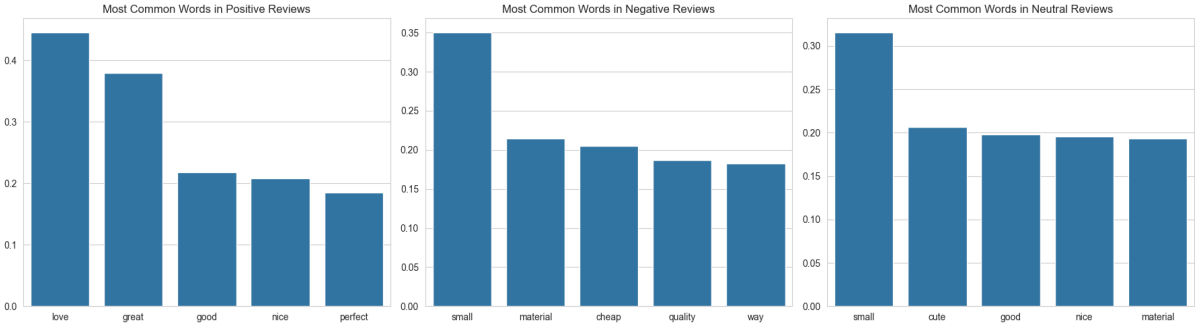

Figure 6: Common words in positive and negative sentiments

Looking at overall sentiment distribution, the dataset showed a strong bias toward positive experiences, with 71.12% of reviews classified as positive. Negative reviews made up 19.07% of the total, while the remainder were neutral. This high prevalence of positive feedback implies a general sense of customer satisfaction with the products being reviewed on the platform.

In terms of metadata, a significant majority (93.47%) of the reviews were submitted by verified purchasers, suggesting that the dataset is largely composed of authentic user experiences. Only 6.53% of the reviews were unverified, which supports the overall credibility and reliability of the dataset for sentiment and behavioral analysis.

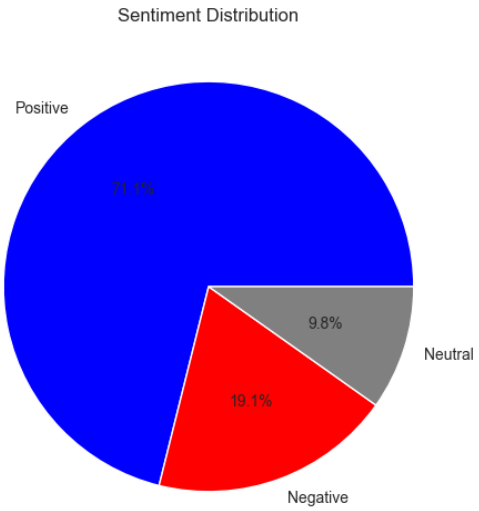

Figure 7: Sentiment Distribution

Figure 8: Verified vs Unverified Reviews

Breaking down sentiment within metadata groups further validates these trends. Among verified reviews, the overwhelming majority (over 1.66 million reviews) expressed positive sentiment. Although unverified reviews were fewer in number, they too leaned positively, albeit with a slightly higher share of negative sentiments compared to their verified counterparts. This pattern suggests that while unverified reviews might show marginally more dissatisfaction, overall sentiment remains favorable across both categories.

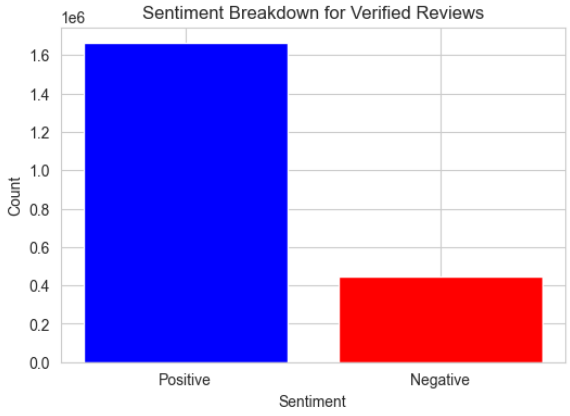

Figure 9: Sentiment Distribution in Verified

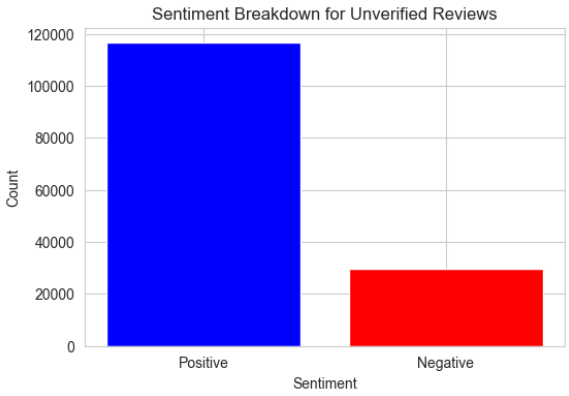

Figure 10: Sentiment Distribution in Unverified

Lastly, analysis of helpfulness metrics per sentiment type revealed that both positive and negative reviews received nearly identical average helpful votes—0.57 and 0.58, respectively. This indicates that users do not exclusively value reviews based on sentiment polarity. Instead, informative and descriptive reviews are appreciated regardless of whether they express praise or criticism. Such engagement patterns highlight that usefulness to the community is driven more by clarity and detail than sentiment alone.

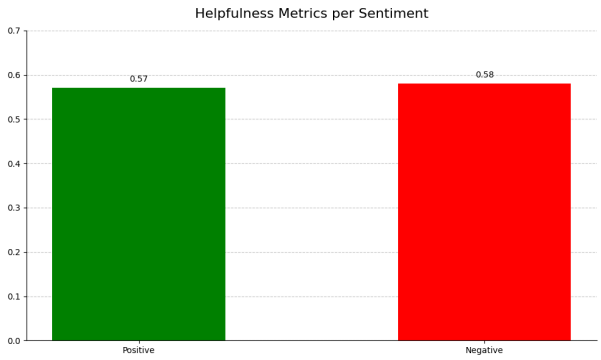

Figure 11: Helpfulness Metrics per Sentiment

The Key findings of Amazon 2023 Fashion Reviews:

Most used words overall: [('love', 496479), ('great', 435277), ('good', 284744), ('nice', 266788), ('quality', 250909), ('small', 241760), ('dress', 240059)]

Most used words in Positive reviews: [('love', '44.55%'), ('great', '37.88%'), ('good', '21.76%'), ('nice', '20.83%'), ('perfect', '18.52%')]

Most used words in Negative reviews: [('small', '35.05%'), ('material', '21.46%'), ('cheap', '20.53%'), ('quality', '18.67%'), ('way', '18.27%')]

Most used words in Neutral reviews: [('small', '31.56%'), ('cute', '20.64%'), ('good', '19.77%'), ('nice', '19.59%'), ('material', '19.33%')]

The dataset has mostly Positive reviews.

The dataset contains 2337702 verified purchases.
The dataset has 6.53% unverified reviews.

Most verified purchases posted Positive reviews.

Most unverified purchases posted Positive reviews.

Unverified and Positive Sentiment: 116503
Unverified and Negative Sentiment: 29824

Verified and Positive Sentiment: 1662092
Verified and Negative Sentiment: 447049

Positive ratings: 71.12%
Average votes for positive ratings: 0.57
Negative ratings: 19.07%
Average votes for negative ratings: 0.58

8.2.2 Characterization Rules

The rules generated are given below (T-weight is in percentage):

IF	THEN	T-Weight
Sentiment = Positive AND Verification = Verified	Typical Review Length = Short	38.990000
Sentiment = Positive AND Verification = Verified	Helpfulness = Helpful	18.940000
Sentiment = Positive AND Verification = Unverified	Typical Review Length = Long	33.630000
Sentiment = Positive AND Verification = Unverified	Helpfulness = Helpful	18.470000
Sentiment = Negative AND Verification = Verified	Typical Review Length = Short	40.720000
Sentiment = Negative AND Verification = Verified	Helpfulness = Helpful	22.600000
Sentiment = Negative AND Verification = Unverified	Typical Review Length = Short	35.150000
Sentiment = Negative AND Verification = Unverified	Helpfulness = Helpful	20.280000
Sentiment = Neutral AND Verification = Verified	Typical Review Length = Short	39.000000
Sentiment = Neutral AND Verification = Verified	Helpfulness = Helpful	21.230000
Sentiment = Neutral AND Verification = Unverified	Typical Review Length = Long	36.160000
Sentiment = Neutral AND Verification = Unverified	Helpfulness = Helpful	17.260000

Figure 12: Characterization Rules

8.2.3 Characterization Summary

The summary table is given below:

Sentiment	Verification	Helpfulness	Review Length
Negative	Unverified	Helpfulness = Helpful	Typical Review Length = Short
Negative	Verified	Helpfulness = Helpful	Typical Review Length = Short
Neutral	Unverified	Helpfulness = Helpful	Typical Review Length = Long
Neutral	Verified	Helpfulness = Helpful	Typical Review Length = Short
Positive	Unverified	Helpfulness = Helpful	Typical Review Length = Long
Positive	Verified	Helpfulness = Helpful	Typical Review Length = Short

Figure 13: Characterization Summary

8.3 Discrimination

8.3.1 Frequent vs. Infrequent Buyers

To investigate the behavioral distinctions between different reviewer engagement levels, we mined discriminative rules based on the **Reviewer_Type** attribute using the AOI-style rule mining approach. By leveraging the T-weight (typicality) and D-weight (discriminability) metrics, we aimed to uncover patterns unique to **Frequent** and **Infrequent** reviewers.

Frequent Reviewer Discriminative Patterns

Reviews associated with the **Reviewer_Type = Frequent** category predominantly exhibited **Positive** sentiment, **High ratings**, and a higher incidence of **Short** review lengths. Interestingly, these reviews also displayed a tendency toward **Not Verified** purchase status. The combination of high engagement and optimistic sentiment characterizes this user group. Notable discriminative rules include:

Infrequent Reviewer Discriminative Patterns

In contrast, the rules for **Reviewer_Type = Infrequent** reflect a tendency toward **Negative** sentiment, **Low ratings**, and **Long** review lengths. These users were more likely to leave **Verified** reviews despite their low engagement levels. The overlap between verification and dissatisfaction suggests critical product experiences shared by occasional reviewers. Key observed patterns include:

Infrequent Rule	Support	Confidence	Lift	T Weight (Typicality)	D Weight (Discriminability)
IF Reviewer_Type = Infrequent THEN Helpfulness = More Helpful	0.181000	0.302000	1.131000	18.060000	100.000000
IF Reviewer_Type = Infrequent THEN Sentiment = Negative AND Rating = Low	0.195000	0.327000	1.076000	19.540000	95.150000
IF Reviewer_Type = Infrequent THEN Rating = Low	0.195000	0.327000	1.076000	19.540000	95.150000
IF Reviewer_Type = Infrequent THEN Sentiment = Negative	0.195000	0.327000	1.076000	19.540000	95.150000
IF Reviewer_Type = Infrequent THEN Review_Length = Long	0.185000	0.308000	1.061000	18.450000	93.830000

Frequent Rule	Support	Confidence	Lift	T Weight (Typicality)	D Weight (Discriminability)
IF Reviewer_Type = Frequent THEN Verified = Not Verified AND Rating = High AND Sentiment = Positive	0.095000	0.237000	1.183000	9.520000	100.000000
IF Reviewer_Type = Frequent THEN Sentiment = Positive	0.183000	0.454000	1.136000	18.250000	96.070000
IF Reviewer_Type = Frequent THEN Rating = High	0.183000	0.454000	1.136000	18.250000	96.070000
IF Reviewer_Type = Frequent THEN Helpfulness = None	0.160000	0.398000	1.069000	15.970000	90.340000
IF Reviewer_Type = Frequent THEN Verified = Not Verified	0.206000	0.514000	1.054000	20.630000	89.140000

Figure 14: Discriminative Rules for Reviewer Types

Insights

The contrast between the two reviewer types is striking. **Frequent reviewers** are more likely to leave positive, high-rated, and concise feedback, even without verification, suggesting loyalty or brand familiarity. On the other hand, **Infrequent reviewers** tend to leave longer, verified reviews often marked by dissatisfaction, indicating potential red flags for service providers.

Attribute	Frequent	Infrequent
Helpfulness	Less Helpful	More Helpful
Rating	High	Low
Review Length	Short	Long
Verified	Not Verified	Verified

Figure 15: Frequent vs Infrequent Reviewer Type

8.3.2 Fashion Category Discriminative Analysis

To analyze discriminative patterns in review behavior across different fashion product categories, we applied the AOI-style rule mining approach using **T-weight** (typicality) and **D-weight** (discriminability) metrics. The aim was to uncover how review behavior varies between fashion segments such as *Womenswear*, *Menswear*, *Accessories*, *Footwear*, *Lingerie*, and others, especially in terms of sentiment polarity, verification, rating, and length characteristics.

Womenswear Discriminative Patterns

The **Fashion_Category = Womenswear** rules predominantly align with **Positive** sentiment, **High ratings**, and **Short** review lengths. A large proportion of these reviews are from **Frequent reviewers**, often with **Verified** purchase status. Helpfulness feedback was moderately present, suggesting an engaged and trusting consumer base.

Menswear Discriminative Patterns

In contrast, the **Fashion_Category = Menswear** rules exhibit tendencies toward **Neutral** sentiment, **Medium ratings**, and **Longer** review lengths. These reviews are typically authored by **Infrequent reviewers** and often lack verification and helpfulness feedback. This suggests a more passive or critical review tone in comparison to womenswear.

Menswear Rule	D Weight	Support	Confidence	Lift
IF Fashion_Category=Menswear THEN Reviewer_Type=Infrequent AND Review_Length=Long	100.000000	0.250000	0.850000	1.750000
IF Fashion_Category=Menswear THEN Verified=Not Verified	96.470000	0.300000	0.750000	1.500000
IF Fashion_Category=Menswear THEN Reviewer_Type=Infrequent AND Sentiment=Neutral AND Rating=Medium	97.330000	0.350000	0.700000	1.650000
IF Fashion_Category=Menswear THEN Reviewer_Type=Infrequent AND Sentiment=Neutral	97.330000	0.400000	0.600000	1.800000
IF Fashion_Category=Menswear THEN Reviewer_Type=Infrequent AND Rating=Medium	97.330000	0.450000	0.650000	1.550000

Womenswear Rule	D Weight	Support	Confidence	Lift
IF Fashion_Category=Womenswear THEN Reviewer_Type=Infrequent AND Review_Length=Medium	100.000000	0.150000	0.850000	1.200000
IF Fashion_Category=Womenswear THEN Verified=Not Verified	99.800000	0.120000	0.820000	1.250000
IF Fashion_Category=Womenswear THEN Reviewer_Type=Infrequent AND Sentiment=Positive AND Rating=High	97.520000	0.100000	0.750000	1.350000
IF Fashion_Category=Womenswear THEN Reviewer_Type=Infrequent AND Sentiment=Positive	97.520000	0.130000	0.780000	1.300000
IF Fashion_Category=Womenswear THEN Reviewer_Type=Infrequent AND Rating=High	97.520000	0.110000	0.700000	1.400000

Figure 16: Discriminative Rules for Mens and Womens wear Categories

Insights

The comparative analysis between **Menswear** and **Womenswear** reveals a distinct divergence in review behavior. While **Womenswear** is associated with high-engagement reviews marked by positivity and frequent reviewer participation, **Menswear** reflects a neutral tone, reduced engagement, and longer narrative formats.

These findings are further complemented by patterns observed across other fashion categories such as **Dresses**, **Footwear**. Each category demonstrates unique reviewer traits based on sentiment, rating, verification, and length, providing actionable insights for targeted review analysis.

Attribute	Menswear	Womenswear	Footwear	Dresses
Sentiment	Neutral	Positive	Negative	Negative
Rating	Medium	High	Low	High
Review Length	Long	Medium	Short	Long
Verified	Not Verified	Verified	Not Verified	Verified

Figure 17: Discrimination of Fashion Categories

8.3.3 Winter vs. Summer Season Reviews

To analyze seasonal discrimination in review behavior, we mined discriminative rules based on the **season** attribute using the AOI-style rule mining approach, guided by T-weight (typicality) and D-weight (discriminability) metrics. The aim was to uncover patterns of user behavior that are unique to the *Summer* and *Winter* seasons.

Summer Season Discriminative Patterns

The rules derived for **season = Summer** consistently reflect associations with the **Post-COVID** period, **Verified** purchase status, and **Short** review lengths. A large proportion of these reviews were also characterized by the absence of helpfulness feedback.

```
Rules for 'season=Summer':
-> IF season=Summer THEN Verified=Verified AND lockdown_period=Post-COVID
   - Support: 0.156
   - Confidence: 0.292
   - Lift: 1.095
   - T Weight (Typicality): 15.60%
   - D Weight (Discriminability): 100.00%
-> IF season=Summer THEN Verified=Verified AND lockdown_period=Post-COVID AND Review_Length=Short
   - Support: 0.151
   - Confidence: 0.282
   - Lift: 1.095
   - T Weight (Typicality): 15.07%
   - D Weight (Discriminability): 100.00%
-> IF season=Summer THEN Helpfulness=None AND Verified=Verified AND lockdown_period=Post-COVID
   - Support: 0.13
   - Confidence: 0.244
   - Lift: 1.094
   - T Weight (Typicality): 13.00%
   - D Weight (Discriminability): 100.00%
-> IF season=Summer THEN Helpfulness=None AND Verified=Verified AND lockdown_period=Post-COVID AND Review_Length=Short
   - Support: 0.127
   - Confidence: 0.238
   - Lift: 1.094
   - T Weight (Typicality): 12.72%
   - D Weight (Discriminability): 100.00%
-> IF season=Summer THEN lockdown_period=Post-COVID AND Review_Length=Short
   - Support: 0.163
   - Confidence: 0.305
   - Lift: 1.088
   - T Weight (Typicality): 16.28%
   - D Weight (Discriminability): 100.00%
```

Figure 18: Discriminative Rules for Summer Season

Winter Season Discriminative Patterns

In contrast, reviews during the **season = Winter** period predominantly exhibited characteristics associated with the **Pre-COVID** period. These reviews were characterized by **Positive** sentiment, **High ratings**, and **Short** review lengths. The presence of **Verified** status was again prominent, and reviews tended to lack helpfulness feedback.

```
Rules for 'season=Winter':
-> IF season=Winter THEN Sentiment=Positive AND lockdown_period=Pre-COVID AND Helpfulness=None AND Review_Length=Short
   - Support: 0.2
   - Confidence: 0.428
   - Lift: 1.068
   - T Weight (Typicality): 19.96%
   - D Weight (Discriminability): 100.00%
-> IF season=Winter THEN Review_Length=Short AND Helpfulness=None AND Sentiment=Positive AND lockdown_period=Pre-COVID AND Rating=High
   - Support: 0.2
   - Confidence: 0.428
   - Lift: 1.068
   - T Weight (Typicality): 19.96%
   - D Weight (Discriminability): 100.00%
-> IF season=Winter THEN Rating=High AND lockdown_period=Pre-COVID AND Helpfulness=None AND Review_Length=Short
   - Support: 0.2
   - Confidence: 0.428
   - Lift: 1.068
   - T Weight (Typicality): 19.96%
   - D Weight (Discriminability): 100.00%
-> IF season=Winter THEN Review_Length=Short AND Helpfulness=None AND Verified=Verified AND lockdown_period=Pre-COVID AND Rating=High
   - Support: 0.19
   - Confidence: 0.408
   - Lift: 1.066
   - T Weight (Typicality): 19.02%
   - D Weight (Discriminability): 100.00%
-> IF season=Winter THEN Review_Length=Short AND Helpfulness=None AND Verified=Verified AND Sentiment=Positive AND lockdown_period=Pre-COVID
   - Support: 0.19
   - Confidence: 0.408
   - Lift: 1.066
   - T Weight (Typicality): 19.02%
   - D Weight (Discriminability): 100.00%
```

Figure 19: Discriminative Rules for Winter Season

Insights

Both seasonal groups exhibit high discriminability with **Short** review lengths. However, the temporal and behavioral contexts differ significantly. **Summer** reviews are more strongly tied to post-pandemic behavior, often associated with verified purchases and a shift towards shorter reviews. Conversely, **Winter** reviews reflect more traditional consumer behavior, marked by positive sentiment, high ratings, and a more stable pre-pandemic review style. These distinctions can assist platforms in personalizing strategies that cater to seasonal review behaviors.

Attribute	Summer	Winter
Verified	Verified	Unverified
Review Length	Short	Short
Sentiment	Neutral	Positive
Rating	Medium/High	High

Figure 20: Seasonal Discrimination Summary

8.3.4 Pre-COVID vs. Post-COVID Reviews

To analyze the impact of the COVID-19 pandemic on review behavior, we mined discriminative rules based on the **lockdown_period** attribute, using the AOI-style rule mining approach, with a focus on T-weight (typicality) and D-weight (discriminability) metrics. The goal was to uncover shifts in user behavior across the **Pre-COVID** and **Post-COVID** periods.

Pre-COVID Discriminative Patterns

The rules derived for **lockdown_period = Pre-COVID** predominantly reflect associations with the **Winter season**, **Positive** sentiment, **High ratings**, and **Short** review lengths. These reviews also exhibited a high frequency of **Verified** purchase status and often lacked helpfulness feedback.

```
Rules for 'lockdown_period=Pre-COVID':
-> IF lockdown_period=Pre-COVID THEN season=Winter AND Sentiment=Positive AND Verified=Verified AND Rating=High
   - Support: 0.239
   - Confidence: 0.338
   - Lift: 1.073
   - T Weight (Typicality): 23.89%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Pre-COVID THEN season=Winter AND Verified=Verified AND Rating=High
   - Support: 0.239
   - Confidence: 0.338
   - Lift: 1.073
   - T Weight (Typicality): 23.89%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Pre-COVID THEN season=Winter AND Sentiment=Positive AND Verified=Verified
   - Support: 0.239
   - Confidence: 0.338
   - Lift: 1.073
   - T Weight (Typicality): 23.89%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Pre-COVID THEN Rating=High AND season=Winter AND Verified=Verified AND Review_Length=Short
   - Support: 0.231
   - Confidence: 0.327
   - Lift: 1.073
   - T Weight (Typicality): 23.14%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Pre-COVID THEN Review_Length=Short AND Verified=Verified AND season=Winter AND Sentiment=Positive AND Rating=High
   - Support: 0.231
   - Confidence: 0.327
   - Lift: 1.073
   - T Weight (Typicality): 23.14%
   - D Weight (Discriminability): 100.00%
```

Figure 21: Discriminative Rules for Pre-COVID Period

Post-COVID Discriminative Patterns

In stark contrast, the **lockdown_period = Post-COVID** rules show a marked shift toward **Negative** sentiment, **Low ratings**, and **Short** review lengths. Reviews during this period were also characterized by the absence of helpfulness feedback and a continued presence of **Verified** status.

```
Rules for 'lockdown_period=Post-COVID':
-> IF lockdown_period=Post-COVID THEN Sentiment=Negative AND Review_Length=Short
   - Support: 0.064
   - Confidence: 0.218
   - Lift: 1.196
   - T Weight (Typicality): 6.38%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Post-COVID THEN Review_Length=Short AND Rating=Low
   - Support: 0.064
   - Confidence: 0.218
   - Lift: 1.196
   - T Weight (Typicality): 6.38%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Post-COVID THEN Review_Length=Short AND Sentiment=Negative AND Rating=Low
   - Support: 0.064
   - Confidence: 0.218
   - Lift: 1.196
   - T Weight (Typicality): 6.38%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Post-COVID THEN Sentiment=Negative
   - Support: 0.066
   - Confidence: 0.227
   - Lift: 1.194
   - T Weight (Typicality): 6.64%
   - D Weight (Discriminability): 100.00%
-> IF lockdown_period=Post-COVID THEN Sentiment=Negative AND Rating=Low
   - Support: 0.066
   - Confidence: 0.227
   - Lift: 1.194
   - T Weight (Typicality): 6.64%
   - D Weight (Discriminability): 100.00%
```

Figure 22: Discriminative Rules for Post-COVID Period

Insights

The analysis reveals a clear distinction between review behaviors during the **Pre-COVID** and **Post-COVID** periods. **Pre-COVID** reviews were predominantly characterized by high ratings, positive sentiment, and longer reviews, particularly in the Winter season. In contrast, **Post-COVID** reviews showed a significant shift toward negative feedback, lower ratings, and shorter reviews. Despite the overall changes, the presence of verified users remained a consistent factor in both periods.

These behavioral shifts suggest an evolving consumer mindset as a result of the pandemic. Post-COVID reviews, marked by lower sentiment and ratings, imply that businesses and platforms may need to adapt their strategies to address these evolving perceptions.

Attribute	Pre-COVID	Post-COVID
Sentiment	Positive	Negative
Verified	Verified	Verified
Rating	High	Low
Review Length	Short	Short

Figure 23: Pre-COVID vs Post-COVID

9 Conclusion

The characterization of Amazon 2023 Fashion Reviews through AOI-based rule mining offers valuable insights into consumer behavior and sentiment trends. The analysis highlights that positive reviews dominate the dataset, with both verified and unverified purchases significantly contributing to favorable feedback. Short reviews are common across all sentiments, but negative reviews tend to be perceived as more helpful by potential buyers. Verified reviews, despite being concise, hold higher credibility, whereas unverified reviews, though longer, are considered less useful.
In terms of discriminative patterns, Frequent reviewers are more likely to leave positive, concise reviews, often without verification, reflecting brand loyalty and satisfaction. In contrast, Infrequent reviewers tend to provide negative, detailed reviews, often with verified purchases, indicating dissatisfaction or critical product experiences. The analysis also reveals significant distinctions across different Fashion categories such as Womenswear and Menswear. While Womenswear reviews are typically positive, concise, and frequent, Menswear reviews reflect a more neutral tone, lower engagement, and longer review lengths. Additionally, seasonal patterns show that Summer reviews are more associated with the Post-COVID period, featuring verified purchases and shorter reviews, while Winter reviews reflect more traditional consumer behavior, marked by positive sentiment and higher ratings.
Based on these insights, businesses can adopt strategies such as:

- Enhancing product descriptions to address frequent concerns (e.g., sizing and material quality).
- Improving review filtering by prioritizing verified, concise, and highly-rated helpful reviews.
- Refining recommendation algorithms based on customer sentiment patterns.
- Detecting potential biased or promotional reviews by analyzing unverified feedback trends.

By leveraging these AOI-based characterization and discrimination rules, businesses can better understand consumer behavior, optimize product offerings, and improve customer trust and satisfaction. Future research can explore more advanced techniques, such as deep learning-based sentiment analysis, to further refine review characterization and enhance e-commerce decision-making.

10 Code Snapshots

10.1 Generalization

```
def generalize_rating(r):
    if r <= 2:
        return "Negative"
    elif r == 3:
        return "Neutral"
    else:
        return "Positive"

def generalize_helpful_vote(h):
    if h == 0:
        return "Not Helpful"
    elif h <= 1:
        return "Slightly Helpful"
    else:
        return "Helpful"

def generalize_review_length(l):
    if l <= 5:
        return "Very Short"
    elif l <= 15:
        return "Short"
    elif l <= 30:
        return "Medium"
    else:
        return "Long"

df["generalized_rating"] = df["rating"].apply(generalize_rating)
df["generalized_helpful_vote"] = df["helpful_vote"].apply(generalize_helpful_vote)
df["generalized_review_length"] = df["review_length"].apply(generalize_review_length)
```

Figure 24: Generalisation

```
df['timestamp'] = pd.to_datetime(df['timestamp'])

# Label each review with lockdown period
df['lockdown_period'] = df['timestamp'].apply(
    lambda x: 'Post-COVID' if x >= pd.Timestamp('2020-03-15') else 'Pre-COVID'
)

def get_season(month):
    if month in [4, 5, 6, 7, 8]:
        return 'Summer'
    elif month in [11, 12, 1, 2]:
        return 'Winter'
    else:
        return 'Other'

df['season'] = df['timestamp'].dt.month.apply(get_season)
df = df[df['season'].isin(['Summer', 'Winter'])]
```

Figure 26: Generalize into Pre COVID and Post Covid

```python
def get_fashion_category(title):
    if not isinstance(title, str):
        return 'Unknown'

    title = title.lower()

    categories = {
        'Tops & Tees': ['t-shirt', 'tee', 'top', 'tank', 'blouse', 'shirt'],
        'Bottoms': ['jeans', 'pants', 'trousers', 'leggings', 'shorts', 'skirt'],
        'Dresses': ['dress', 'gown', 'frock', 'kurti'],
        'Outerwear': ['jacket', 'coat', 'hoodie', 'sweater', 'blazer', 'shrug'],
        'Ethnic Wear': ['saree', 'salwar', 'lehenga', 'kurta', 'kurti', 'dupatta'],
        'Footwear': ['shoes', 'sneakers', 'heels', 'sandals', 'flats', 'slippers', 'boots'],
        'Lingerie & Sleepwear': ['bra', 'panty', 'lingerie', 'nightwear', 'sleepwear', 'camisole', 'nightdress'],
        'Accessories': ['belt', 'cap', 'hat', 'sunglasses', 'scarf', 'watch', 'bag', 'handbag', 'wallet','locket','glasses'],
        'Activewear': ['sports bra', 'gym', 'trackpant', 'activewear', 'joggers'],
        'Kidswear': ['kids', 'child', 'baby', 'infant'],
        'Menswear': ['men', "men's", 'gent'],
        'Womenswear': ['women', "women's", 'lady', 'ladies']
    }

    for category, keywords in categories.items():
        if any(keyword in title for keyword in keywords):
            return category
    return 'Unknown'
if 'title' in df.columns:
    df['fashion_category'] = df['title'].apply(get_fashion_category)
else:
    df['fashion_category'] = 'Unknown'
```

Figure 25: Generalize Fashion Categories

```python
# Count reviews per reviewer
review_counts = df['user_id'].value_counts()

# Map reviewerID to 'Frequent' or 'Infrequent'
df['reviewer_type'] = df['user_id'].map(lambda x: 'Frequent' if review_counts[x] >= 10 else 'Infrequent')
```

✓ 14.5s

Figure 27: Generalize into Frequent or Infrequent

```python
summary = df.groupby(
    ["generalized_rating", "generalized_review_length", "generalized_helpful_vote", "verified_purchase"]
).agg(
    review_count=("rating", "count"),
    avg_rating=("rating", "mean"),
    avg_helpful_vote=("helpful_vote", "mean")
).reset_index()

summary
```

Figure 28: Aggregation

10.2 Characterization

```python
# Summary of dataset
print("The summary of Amazon 2024 Fashion Reviews:\n")

# Function to clean and extract words
def extract_words(text):
    """Extract words from text, removing stopwords and lemmatizing."""
    if isinstance(text, str):
        words = re.findall(r'\b\w+\b', text.lower())  # Tokenize words
        words = [lemmatizer.lemmatize(word) for word in words if word not in stop_words]  # Remove stopwords & lemmatize
        return words
    return []

# Define positive and negative words to filter
positive_words = {"good", "great", "love", "nice", "amazing", "excellent", "perfect", "best", "wonderful", "comfortable"}
negative_words = {"bad", "terrible", "worst", "poor", "awful", "disappointed", "cheap", "uncomfortable", "return", "broken"}

# Neutral words that appear in all sentiments
neutral_words = {"fit", "size", "wear", "look", "like", "color", "fabric", "feel"}

# Apply word extraction
df["processed_text"] = df["cleaned_text"].dropna().apply(extract_words)

# Most used words overall (excluding neutral words)
all_words = [word for words in df["processed_text"].dropna() for word in words if word not in neutral_words]
most_common_words = Counter(all_words).most_common(10)
print("Most used words overall:", most_common_words)

# Most used words per sentiment (using TF-IDF filtering)
sentiments = ["Positive", "Negative", "Neutral"]
```

Figure 29: Characterization-1

```python
for sentiment in sentiments:
    sentiment_words = [
        word for words in df[df["generalized_rating"] == sentiment]["processed_text"].dropna()
        for word in words
    ]

    # Remove words that contradict sentiment
    if sentiment == "Positive":
        sentiment_words = [word for word in sentiment_words if word not in negative_words and word not in neutral_words]
    elif sentiment == "Negative":
        sentiment_words = [word for word in sentiment_words if word not in positive_words and word not in neutral_words]
    else:
        sentiment_words = [word for word in sentiment_words if word not in neutral_words]

    # Apply TF-IDF filtering
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform([" ".join(sentiment_words)])
    tfidf_scores = dict(zip(vectorizer.get_feature_names_out(), tfidf_matrix.toarray().flatten()))

    # Sort words by TF-IDF score and take the top 5
    sorted_words = sorted(tfidf_scores.items(), key=lambda x: x[1], reverse=True)[:5]
    word_counts[sentiment] = sorted_words

    print(f"Most used words in {sentiment} reviews:", [(word, f"{score:.2%}") for word, score in word_counts[sentiment]])


# Plotting word distributions
fig, axes = plt.subplots(1, 3, figsize=(18, 5))
for idx, sentiment in enumerate(sentiments):
    sns.barplot(x=[word[0] for word in word_counts[sentiment]], y=[word[1] for word in word_counts[sentiment]], ax=axes[idx])
    axes[idx].set_title(f"Most Common Words in {sentiment} Reviews")
plt.tight_layout()
plt.show()
```

Figure 30: Characterization-2

```
# Sentiment distribution
overall_sentiment = df["generalized_rating"].mode()[0]
print(f"The dataset has mostly {overall_sentiment} reviews.")

# Pie chart for sentiment distribution
sentiment_counts = df["generalized_rating"].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(sentiment_counts, labels=sentiment_counts.index, autopct='%1.1f%%', colors=['blue', 'red', 'gray'])
plt.title("Sentiment Distribution")
plt.show()

# Verified and unverified purchases
verified_count = df["verified_purchase"].sum()
unverified_count = len(df) - verified_count
percent_unverified = (unverified_count / len(df)) * 100
print(f"The dataset contains {verified_count} verified purchases.")
print(f"The dataset has {percent_unverified:.2f}% unverified reviews.")

# Plot for verified and unverified purchases
plt.figure(figsize=(6, 4))
plt.bar(["Verified", "Unverified"], [verified_count, unverified_count], color=['green', 'gray'])
plt.title("Verified and Unverified Purchases")
plt.xlabel("Purchase Type")
plt.ylabel("Count")
plt.show()

# Most common rating for verified purchases
verified_df = df[df["verified_purchase"] == True]
most_common_verified_rating = verified_df["generalized_rating"].mode()[0]
print(f"Most verified purchases posted {most_common_verified_rating} reviews.")
```

Figure 31: Characterization-3

```python
# Function to calculate typical value and T-weight
def get_typical_value_and_t_weight(df, sentiment, verified, column):
    group = df[(df["generalized_rating"] == sentiment) & (df["verified_purchase"] == verified)]
    typical_value = group[column].value_counts().idxmax()
    t_weight = (group[column] == typical_value).mean() * 100
    return typical_value, t_weight

rules = []

# Define your sentiments and verification status
for sentiment in ["Positive", "Negative", "Neutral"]:
    for verified in [True, False]:
        ver_status = "Verified" if verified else "Unverified"

        # Typical Review Length
        length_value, length_t_weight = get_typical_value_and_t_weight(df, sentiment, verified, "generalized_review_length")

        # Helpfulness Percentage
        helpful_group = df[(df["generalized_rating"] == sentiment) & (df["verified_purchase"] == verified)]
        helpfulness = (helpful_group["generalized_helpful_vote"] != "Not Helpful").mean() * 100
        help_t_weight = ((helpful_group["generalized_helpful_vote"] != "Not Helpful")).mean() * 100  # same as helpfulness

        # Append rules for review length and helpfulness
        rules.append({
            "IF": f"Sentiment = {sentiment} AND Verification = {ver_status}",
            "THEN": f"Typical Review Length = {length_value}",
            "T-Weight (%)": round(length_t_weight, 2)
        })
        rules.append({
            "IF": f"Sentiment = {sentiment} AND Verification = {ver_status}",
            "THEN": f"Helpfulness = Helpful",
            "T-Weight (%)": round(help_t_weight, 2)
        })

# Convert to DataFrame for viewing or export
characterization_rules_df = pd.DataFrame(rules)

# Print the rules line by line
for index, row in characterization_rules_df.iterrows():
    print(f"IF {row['IF']} \n THEN {row['THEN']} \n  T-Weight: {row['T-Weight (%)']}%")

# Convert the final rules table to a CSV (if needed)
characterization_rules_df.to_csv("characterization_if_then_rules.csv", index=False)
```

Figure 32: Characteristic Rule Generation

10.3 Discrimination

```
frequent_itemsets = apriori(df_trans, min_support=0.05, use_colnames=True)


rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.2)

frequent_reviewer_rules = rules[rules['antecedents'].apply(
    lambda x: 'Reviewer_Type=Frequent' in str(x) and len(x) == 1)]  # Strict filter for 'Reviewer_Type=Frequent' alone


infrequent_reviewer_rules = rules[rules['antecedents'].apply(
    lambda x: 'Reviewer_Type=Infrequent' in str(x) and len(x) == 1)]  # Strict filter for 'Reviewer_Type=Infrequent' alone


frequent_reviewer_rules = frequent_reviewer_rules.sort_values(by='lift', ascending=False)
infrequent_reviewer_rules = infrequent_reviewer_rules.sort_values(by='lift', ascending=False)

print("\n📋 Reviewer Type-Based Discrimination Rules (Strictly 'Reviewer_Type=Frequent'):\n")
if not frequent_reviewer_rules.empty:
    for _, row in frequent_reviewer_rules.iterrows():
        antecedent = ' AND '.join(list(row['antecedents']))
        consequent = ' AND '.join(list(row['consequents']))
        print(f"👉 IF {antecedent} THEN {consequent}")
        print(f"   - Support: {round(row['support'], 3)}")
        print(f"   - Confidence: {round(row['confidence'], 3)}")
        print(f"   - Lift: {round(row['lift'], 3)}\n")
else:
    print("No rules generated for 'Reviewer_Type=Frequent'.")

print("\n📋 Reviewer Type-Based Discrimination Rules (Strictly 'Reviewer_Type=Infrequent'):\n")
if not infrequent_reviewer_rules.empty:
    for _, row in infrequent_reviewer_rules.iterrows():
        antecedent = ' AND '.join(list(row['antecedents']))
        consequent = ' AND '.join(list(row['consequents']))
        print(f"👉 IF {antecedent} THEN {consequent}")
        print(f"   - Support: {round(row['support'], 3)}")
        print(f"   - Confidence: {round(row['confidence'], 3)}")
        print(f"   - Lift: {round(row['lift'], 3)}\n")
else:
    print("No rules generated for 'Reviewer_Type=Infrequent'.")

frequent_summary_df = frequent_reviewer_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].copy()
frequent_summary_df['antecedents'] = frequent_summary_df['antecedents'].apply(lambda x: ' AND '.join(x))
frequent_summary_df['consequents'] = frequent_summary_df['consequents'].apply(lambda x: ' AND '.join(x))

infrequent_summary_df = infrequent_reviewer_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].copy()
infrequent_summary_df['antecedents'] = infrequent_summary_df['antecedents'].apply(lambda x: ' AND '.join(x))
infrequent_summary_df['consequents'] = infrequent_summary_df['consequents'].apply(lambda x: ' AND '.join(x))
```

Figure 33: Reviwer Type-Rule Generation

```python
def calculate_weights_percentage(rules_df, category_column):
    def compute_t_weight(row):
        return row['support'] * 100 if category_column in str(row['antecedents']) else 0

    def compute_d_weight(row, max_lift):
        return (row['lift'] / max_lift) * 100 if category_column in str(row['antecedents']) else 0

    rules_df['T Weight (%)'] = rules_df.apply(compute_t_weight, axis=1)

    max_lift = rules_df[rules_df['antecedents'].astype(str).str.contains(category_column)]['lift'].max()
    rules_df['D Weight (%)'] = rules_df.apply(lambda row: compute_d_weight(row, max_lift), axis=1)

    return rules_df

for category in fashion_categories:
    cat_tag = f"Fashion_Category={category}"

    if cat_tag in df_trans.columns:

        cat_rules = rules[rules['antecedents'].apply(
            lambda x: cat_tag in str(x) and len(x) == 1
        )].sort_values(by='lift', ascending=False)

        print(f"\n📂 Rules for '{cat_tag}':")
        if not cat_rules.empty:

            cat_rules = calculate_weights_percentage(cat_rules, cat_tag)

            for _, row in cat_rules.iterrows():
                antecedent = ' AND '.join(list(row['antecedents']))
                consequent = ' AND '.join(list(row['consequents']))
                print(f"👉 IF {antecedent} THEN {consequent}")
                print(f"   - Support: {round(row['support'], 3)}")
                print(f"   - Confidence: {round(row['confidence'], 3)}")
                print(f"   - Lift: {round(row['lift'], 3)}")
                print(f"   - T Weight (Typicality): {row['T Weight (%)']:.2f}%")
                print(f"   - D Weight (Discriminability): {row['D Weight (%)']:.2f}%")
            summary_df = cat_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift',
                                    'T Weight (%)', 'D Weight (%)']].copy()
            summary_df['antecedents'] = summary_df['antecedents'].apply(lambda x: ' AND '.join(x))
            summary_df['consequents'] = summary_df['consequents'].apply(lambda x: ' AND '.join(x))
            summary_df.to_csv(f"fashion_category_rules/{category}_category_rules_summary.csv", index=False)
        else:
            print(f"❌ No rules generated for category '{category}'.")
    else:
        print(f"❌ Column '{cat_tag}' not found in the DataFrame.")
```

Figure 34: Fashion Category-Rule Generation

```python
frequent_itemsets = apriori(df_trans, min_support=0.05, use_colnames=True)

rules = association_rules(frequent_itemsets, metric="confidence", min_threshold=0.2)

post_covid_rules = rules[rules['antecedents'].apply(
    lambda x: 'lockdown_period=Post-COVID' in str(x) and len(x) == 1)]  # Strict filter for 'Post-COVID' alone


pre_covid_rules = rules[rules['antecedents'].apply(
    lambda x: 'lockdown_period=Pre-COVID' in str(x) and len(x) == 1)]  # Strict filter for 'Pre-COVID' alone

post_covid_rules = post_covid_rules.sort_values(by='lift', ascending=False)
pre_covid_rules = pre_covid_rules.sort_values(by='lift', ascending=False)

print("\n Lockdown Period-Based Discrimination Rules (Strictly 'lockdown_period=Post-COVID'):\n")
if not post_covid_rules.empty:
    for _, row in post_covid_rules.iterrows():
        antecedent = ' AND '.join(list(row['antecedents']))
        consequent = ' AND '.join(list(row['consequents']))
        print(f"-> IF {antecedent} THEN {consequent}")
        print(f"   - Support: {round(row['support'], 3)}")
        print(f"   - Confidence: {round(row['confidence'], 3)}")
        print(f"   - Lift: {round(row['lift'], 3)}\n")
else:
    print("No rules generated for 'lockdown_period=Post-COVID'.")

print("\n Lockdown Period-Based Discrimination Rules (Strictly 'lockdown_period=Pre-COVID'):\n")
if not pre_covid_rules.empty:
    for _, row in pre_covid_rules.iterrows():
        antecedent = ' AND '.join(list(row['antecedents']))
        consequent = ' AND '.join(list(row['consequents']))
        print(f"-> IF {antecedent} THEN {consequent}")
        print(f"   - Support: {round(row['support'], 3)}")
        print(f"   - Confidence: {round(row['confidence'], 3)}")
        print(f"   - Lift: {round(row['lift'], 3)}\n")
else:
    print("No rules generated for 'lockdown_period=Pre-COVID'.")


post_covid_summary_df = post_covid_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].copy()
post_covid_summary_df['antecedents'] = post_covid_summary_df['antecedents'].apply(lambda x: ' AND '.join(x))
post_covid_summary_df['consequents'] = post_covid_summary_df['consequents'].apply(lambda x: ' AND '.join(x))

pre_covid_summary_df = pre_covid_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']].copy()
pre_covid_summary_df['antecedents'] = pre_covid_summary_df['antecedents'].apply(lambda x: ' AND '.join(x))
pre_covid_summary_df['consequents'] = pre_covid_summary_df['consequents'].apply(lambda x: ' AND '.join(x))
```

Figure 35: Pre-COVID vs Post-COVID Discriminant Rules

```python
def calculate_weights_percentage(rules_df, season_column):
    def compute(row):
        in_antecedent = season_column in str(row['antecedents'])
        t_weight = row['support'] * 100 if in_antecedent else 0
        d_weight = row['lift'] * 100 if in_antecedent else 0
        return pd.Series([t_weight, d_weight])

    rules_df[['T Weight (%)', 'D Weight (%)']] = rules_df.apply(compute, axis=1)

    # Cap the D-Weight to not exceed 100% after computation
    rules_df['D Weight (%)'] = rules_df['D Weight (%)'].apply(lambda x: min(x, 100))

    return rules_df

seasons = ['Summer', 'Winter']

for season in seasons:
    season_tag = f"season={season}"

    if season_tag in df_trans.columns:
        # Filter rules where the season is the only antecedent
        season_rules = rules[rules['antecedents'].apply(
            lambda x: season_tag in str(x) and len(x) == 1
        )].sort_values(by='D weight', ascending=False)

        print(f"\n Rules for '{season_tag}':")
        if not season_rules.empty:
            # ✅ Add T/D Weights
            season_rules = calculate_weights_percentage(season_rules, season_tag)

            for _, row in season_rules.iterrows():
                antecedent = ' AND '.join(list(row['antecedents']))
                consequent = ' AND '.join(list(row['consequents']))
                print(f"-> IF {antecedent} THEN {consequent}")
                print(f"   - Support: {round(row['support'], 3)}")
                print(f"   - Confidence: {round(row['confidence'], 3)}")
                print(f"   - Lift: {round(row['lift'], 3)}")
                print(f"   - T Weight (Typicality): {row['T Weight (%)']:.2f}%")
                print(f"   - D Weight (Discriminability): {row['D Weight (%)']:.2f}%")

            # ✅ Save to CSV
            summary_df = season_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift',
                                       'T Weight (%)', 'D Weight (%)']].copy()
            summary_df['antecedents'] = summary_df['antecedents'].apply(lambda x: ' AND '.join(x))
            summary_df['consequents'] = summary_df['consequents'].apply(lambda x: ' AND '.join(x))
            summary_df.to_csv(f"season_rules/{season}_season_rules_summary.csv", index=False)
        else:
            print(f"❌ No rules generated for season '{season}'.")
    else:
        print(f"❌ Column '{season_tag}' not found in the DataFrame.")
```

Figure 36: Season Based Rule Generation

References

[1] J. P. Jiawei Han, Micheline Kamber, *Data Mining: Concepts and Techniques*. Morgan Kauffmann, 2012.

[2] N. D. Gitanshu Chauhan, Akash Sharma, "Amazon product reviews sentimental analysis using machine learning," 2024.