Assignment 3 - Data Analytics

Team 31

Movie Recommendation System

This report documents the process of building a movie recommendation system using the MovieLens dataset, which contains over 100,000 ratings and tag applications across nearly 9,000 movies. The goal of this system is to recommend movies to users based on their prior ratings. We employed association rule mining techniques to identify patterns between user preferences and movies to generate recommendations. We then evaluated the model's performance using precision and recall metrics.

Algorithm Selection

We chose the FP-Growth Algorithm for mining association rules due to its efficiency and scalability especially in the case of large datasets like MovieLens. FP-Growth compresses the dataset into a FP-tree structure, which is a modified Trie structure, that allows for faster computation of frequent itemsets because it only scans the dataset twice - once to build the tree and once to mine the frequent itemsets, avoiding candidate generation. The Apriori algorithm requires iterative candidate generation and multiple passes. Because of this, we can say that FP-Growth is more efficient in terms of memory than algorithms like Apriori which is important when dealing with large datasets. This choice was driven by the need to handle high-volume data with minimal computational overhead.

Building the Recommendation System

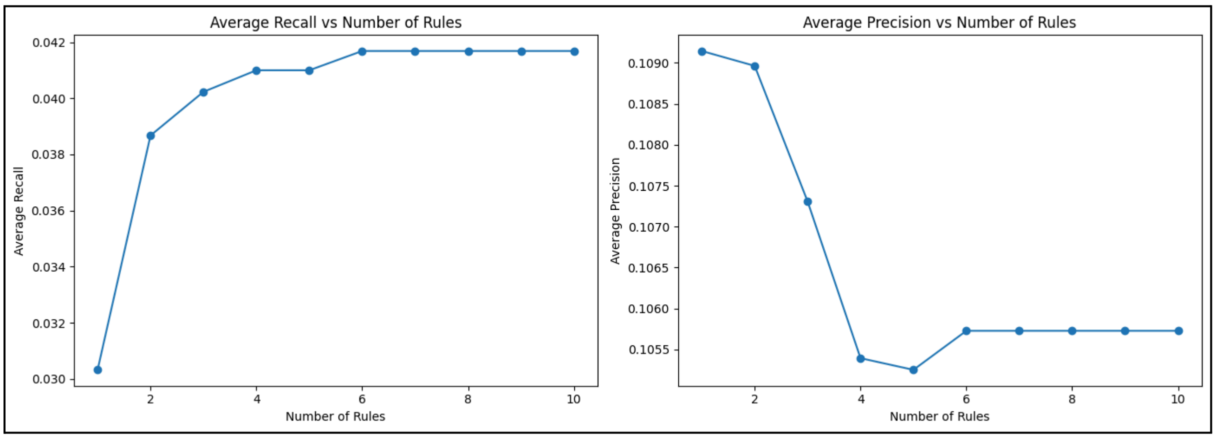
- **Dataset Preparation**:
    - The **MovieLens dataset** was used, consisting of users' ratings on various movies.
    - To ensure transactional data is meaningful, users who rated a sufficient number of movies (10) to ensure they are active and movies with ratings above 2, which are interesting were retained for analysis. This transactional data was split into 80% training and 20% test sets for each user.
- **FP-Growth Algorithm**:
    - The FP-Growth algorithm was implemented to discover frequent itemsets from users' movie ratings.
    - We define a Node class to represent each node in the FP-tree. Each node stores:
        - Item : The movie associated with the node.
        - Count : The number of times this movie appears in the path represented by the node.
        - Parent : The parent node in the tree.
        - Children : A dictionary mapping each child item to its corresponding node.

    - We begin by counting the frequency of each movie in the dataset. Movies that do not meet the minimum support threshold are filtered out. The tree is constructed by inserting transactions, where movies are ordered by their frequency in descending order.
    - Each transaction is inserted into the FP-tree, updating the count for existing nodes or creating new nodes when necessary. The header table is updated with links to nodes of the same item, allowing us to efficiently mine conditional pattern bases later.
    - We recursively mine frequent patterns using conditional pattern bases. For each item in the header table, we build its conditional pattern base by tracing paths from the leaf nodes to the root. A subtree is constructed using the conditional pattern base, and the process is repeated recursively to find frequent itemsets. The resulting patterns are added to the set of frequent patterns.
- **Association Rule Generation**:
    - After mining frequent patterns, **association rules** were generated from these patterns, representing combinations of movies that were often rated together by users. Confidence and support values are used to rank these rules:
        - **Support**: How often the movies occur together.
        - **Confidence**: How likely it is that a user who rated one movie will rate the other.
    - Based on these itemsets, association rules were generated. A minimum confidence and support threshold was set to filter out weak rules, ensuring that only the most relevant recommendations are considered.
- **Generating Recommendations**:
    - Recommendations were generated by looking at rules with single-item antecedents (i.e., recommendations for a single movie). For each user, if a movie they rated is found in the antecedents of a rule, the corresponding consequent (recommended movie) is retrieved.
    - For each user, based on their past ratings, the system would recommend movies that were associated with the movies they had previously enjoyed, as identified by the frequent itemsets.
    - The top 100 rules based on support and confidence were selected for making recommendations.
- **Evaluation**:
    - Precision and Recall were calculated to evaluate the quality of the recommendations.
        - **Precision**: The proportion of recommended movies that are relevant (i.e., movies that the user actually rated in the test set).
        - **Recall**: The proportion of relevant movies that were successfully recommended to the user.
    - **Graphs** showing the relationship between the number of rules and the precision/recall as well as graphs of precision/recall for a sample of users were plotted to visualise system performance. The evaluation showed that as more

rules are used, recall improves, but precision decreases, indicating a trade-off between recommending more movies (recall) and keeping the recommendations accurate (precision).

Performance Evaluation:

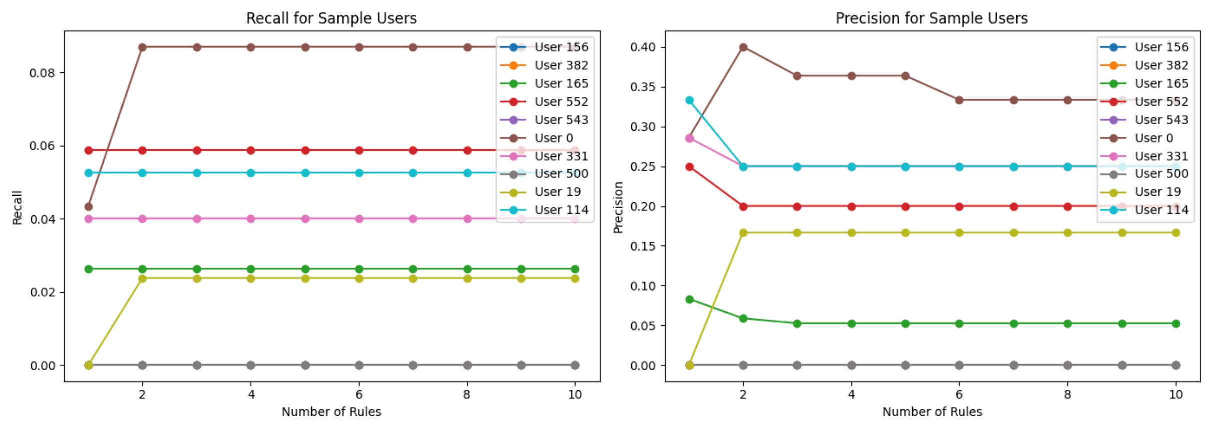
Recall and Precision vs Number of Rules:



The performance of the recommendation system was measured using **precision** and **recall**. These metrics help us understand how well the system is balancing the accuracy of recommendations with the number of relevant movies suggested.

- **Precision**: Precision measures the proportion of recommended movies that are relevant to the user. Initially, precision starts high but gradually declines as more association rules are applied. This decline happens because as we expand the number of recommendations, we start to include more irrelevant items, lowering the overall precision.
- **Recall**: Recall reflects how many of the relevant movies (i.e., movies the user might like) are included in the recommendations. As more rules are applied, recall improves since more relevant movies are captured, but this improvement eventually levels off when we reach a saturation point where adding additional rules doesn't yield new relevant recommendations.
- **Balancing Precision and Recall**: The trade-off between precision and recall is evident in the results. Initially, increasing the number of rules helps the system recommend more relevant movies (boosting recall), but at the expense of precision. After a certain point, recall levels off, and adding more rules introduces noise, which further decreases precision.

Recall and Precision for Sample Users:



Recall for Sample Users (Left Plot):

- Recall measures the ratio of relevant movies (those in the test set) that were successfully recommended.
- For some users, recall remains constant across the increasing number of rules. This suggests that adding more rules doesn't increase the number of relevant items retrieved for the users.
- A higher recall means the system is recommending more of the relevant items (movies in the user's test data).

Precision for Sample Users (Right Plot):

- Precision is the ratio of recommended movies that are relevant (from the test set).
- For most users, precision tends to drop and then stabilise as the number of rules increases. This indicates that as more rules are considered, more irrelevant items are retrieved and the recommendations become less precise.
- A higher precision means the recommendations are more accurate (a higher proportion of recommended movies are in the user's test set)..

This suggests the recommendation system performs better for some users than others, balancing recall and precision differently depending on the user.