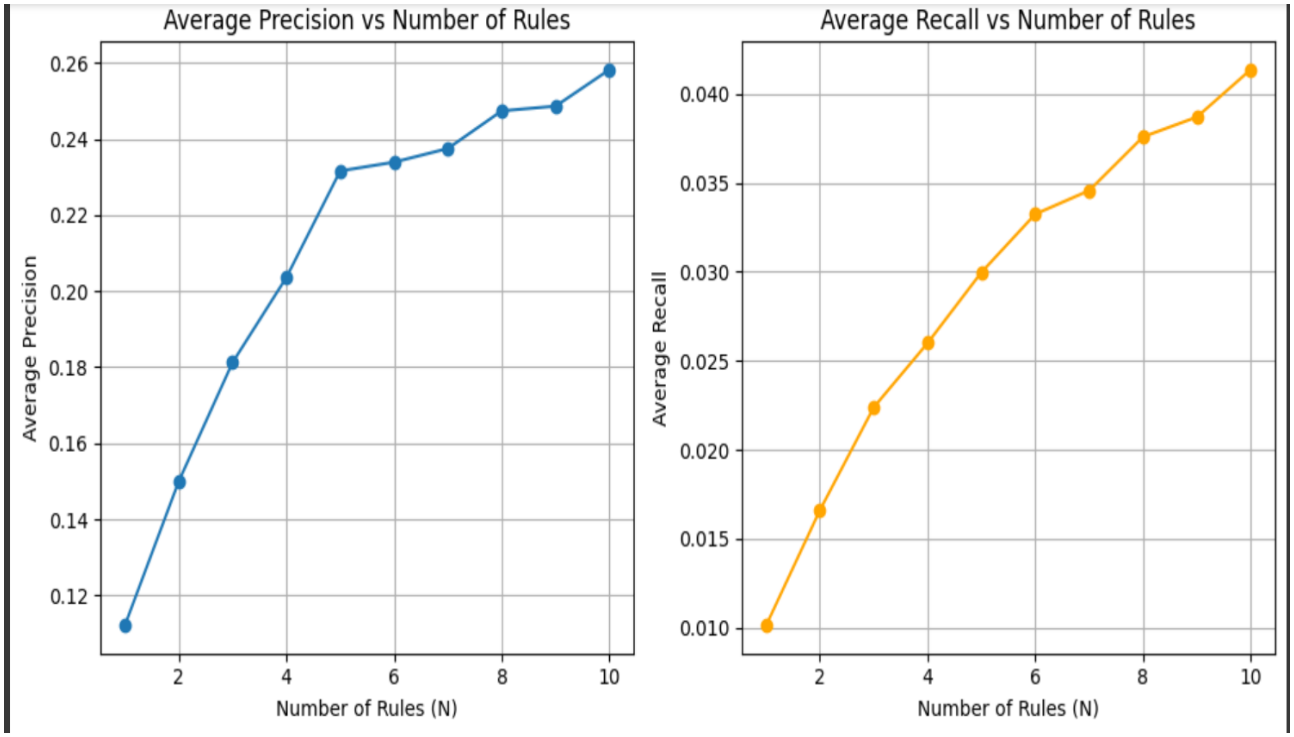
DATA ANALYTICS I

Assignment-3 Report



Average Precision vs. Number of Rules (left graph):

- Precision measures how many of the generated rules are actually correct or useful (true positives compared to all predicted positives).
- Initially, as the number of rules (frequent patterns) increases, precision improves because the initial rules are likely based on the most frequent and reliable patterns.
- However, after a certain point (around N = 7 in the graph), precision starts to level off or slightly decrease. This happens because adding more rules (i.e., rules generated from less frequent patterns) may introduce less reliable rules, increasing false positives. Therefore, while we are finding more patterns, the quality of the new rules may decline, causing precision to plateau.

Average Recall vs. Number of Rules (right graph):

- Recall measures how many of the actual frequent patterns are captured by the generated rules (true positives compared to all actual positives).
- As we increase the number of rules, recall improves steadily. This is because more rules mean that we are capturing a larger portion of the frequent patterns in the dataset. Even though

some rules might not be very precise, they help capture more true positive patterns, hence recall increases.

- Unlike precision, recall continues to rise without plateauing, as more rules allow the model to identify more frequent patterns.

Reason for the Behaviour in the FP-Tree Method:

- Trade-off between precision and recall: In FP-Growth, the initial frequent patterns mined are based on the highest support thresholds (the most common patterns), which lead to high precision but lower recall. As we lower the support threshold and generate more rules, recall improves because we capture more patterns, but precision can start to stagnate due to the inclusion of patterns with lower confidence.
- Quality of Rules vs. Quantity: Early rules are based on strong, highly frequent patterns, leading to high precision. As we add more rules (lowering the support threshold), we are also adding rules based on less frequent, noisier patterns, which can reduce precision but boost recall.
- FP-Tree Efficiency and Pattern Generation: The FP-Growth method efficiently mines patterns, and increasing the number of rules usually captures more true positive patterns, which leads to an increase in recall. However, the patterns become less reliable as we mine less frequent ones, which explains the levelling off of precision.

Justification of Algorithm Selection

We selected the Frequent Pattern Growth (FP-Growth) algorithm for mining frequent itemsets and generating association rules from user transaction data. The FP-Growth algorithm is an efficient and scalable method for discovering patterns in large datasets without generating candidate itemsets explicitly, unlike the Apriori algorithm.

The decision to use the FP-Growth algorithm was based on several factors:

- ***Efficiency with Large Datasets:*** Our dataset consisted of numerous users and movies, resulting in a vast number of transactions. FP-Growth is well-suited for large datasets because it compresses the data into a compact structure called the FP-tree, which reduces the number of database scans and speeds up the mining process.

- ***Avoidance of Candidate Generation:*** Unlike Apriori, FP-Growth does not generate candidate itemsets, which can be computationally expensive and memory-intensive with large datasets. This makes FP-Growth more efficient and scalable.

- ***Better Performance:*** FP-Growth generally outperforms other algorithms like Apriori in terms of execution time and memory usage, especially when dealing with dense and large datasets, as it reduces the computational overhead.

- ***Relevance to Association Rule Mining:*** Our objective was to generate association rules that can be used to make recommendations. FP-Growth effectively discovers frequent itemsets that serve as a foundation for generating high-confidence association rules.

Recommendation System Implementation

- ***Data Preprocessing:*** We began by preprocessing the dataset to ensure it was suitable for mining. This involved:
  - ***Data Cleaning:*** Removing users with 10 or fewer movie ratings to focus on active users with sufficient data for analysis.
  - ***Transaction Construction:*** Creating a dictionary of transactions where each user ID mapped to a list of movies they had rated.
  - ***Training and Test Split:*** Dividing the dataset into training and test sets by randomly selecting 20% of each user's movies to form the test set and using the remaining 80% for training.

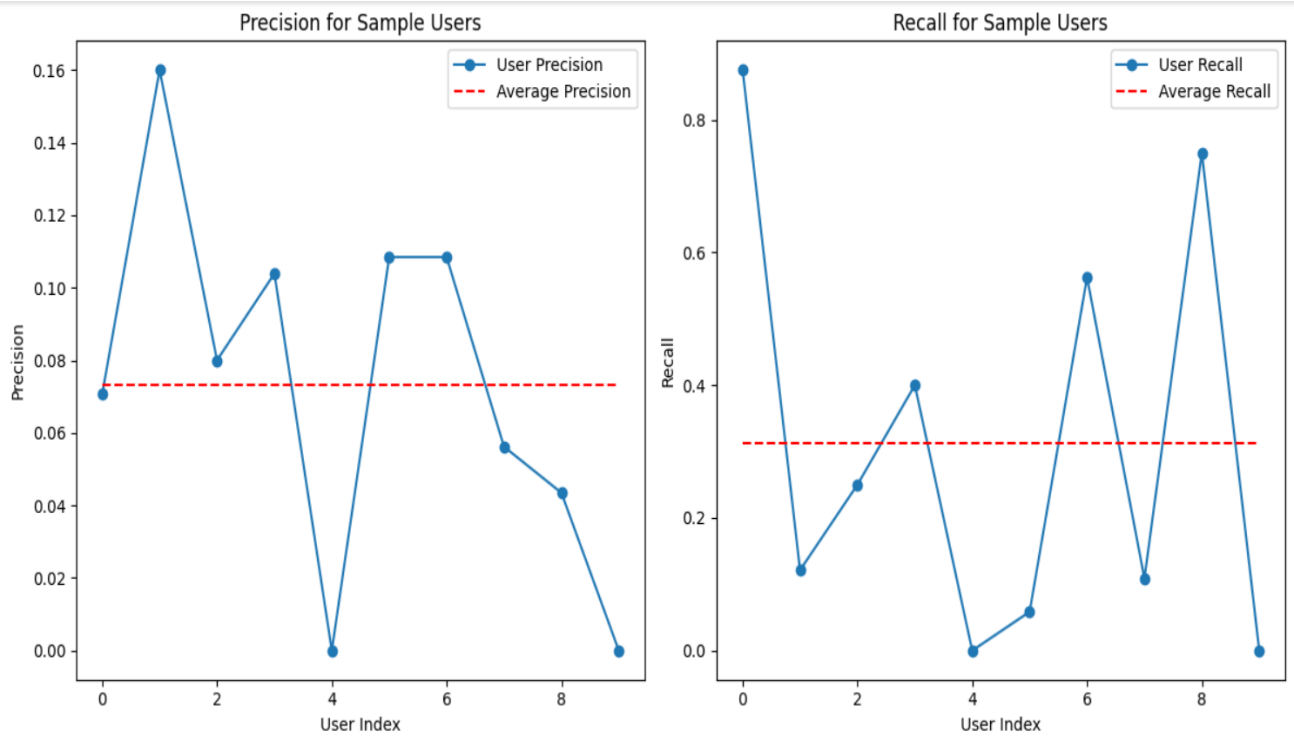
- ***Building the FP-Tree***: Using the training transactions, we built the FP-tree as follows:
  - ***Counting Item Frequencies:*** Calculated the support count of each movie (item) across all transactions.
  - ***Pruning Infrequent Items:*** Removed movies that did not meet the minimum support threshold, ensuring only frequent items were considered.
  - ***Tree Construction:*** Created the root of the FP-tree and inserted transactions in a sorted order based on item frequency, updating counts and maintaining node links for efficient traversal.

- ***Mining Frequent Itemsets and Association Rules:*** With the FP-tree constructed, we mined frequent itemsets by recursively exploring the tree:
  - ***Conditional Pattern Bases:*** For each item, we built its conditional pattern base and conditional FP-tree to find frequent patterns associated with it.
  - ***Frequent Itemset Generation:*** Collected all frequent itemsets that met the minimum support threshold.
    From the frequent itemsets, we generated association rules:
  - ***Rule Generation:*** For each frequent itemset of size two or more, we created rules where the antecedent was a single movie, and the consequent was the remaining movies in the itemset.
  - ***Calculating Confidence:*** Computed the confidence of each rule and retained those that met the minimum confidence threshold.

- ***Recommendation Generation:*** To generate recommendations:
  - ***Rule Matching:*** For each user, we matched the movies they had rated in the training set with the antecedents of the association rules.
  - ***Candidate Recommendations:*** Collected the consequents of matching rules as potential recommendations.
  - ***Filtering:*** Excluded movies the user had already rated to avoid redundant suggestions.
  - ***Ranking:*** Ranked the recommendations based on the confidence of the associated rules, providing the most relevant suggestions first

- ***Evaluation:*** We evaluated the recommendation system by computing precision and recall metrics on a sample of users:
  - ***Precision:*** Measured the proportion of recommended movies that the user had rated in the test set.
  - ***Recall:*** Assessed the proportion of movies in the test set that were successfully recommended.

  By varying the number of recommendations and analysing the precision-recall trade-off, we were able to gauge the effectiveness of our system.



- ***Conclusion:***

  In summary, the selection of the FP-Growth algorithm was instrumental in efficiently mining frequent patterns from a large dataset, enabling us to generate meaningful association rules for our recommendation system. The algorithm's ability to handle large-scale data and produce accurate patterns justified its selection. The system we built leverages these patterns to provide personalized movie recommendations, demonstrating satisfactory performance as indicated by our evaluation metrics.