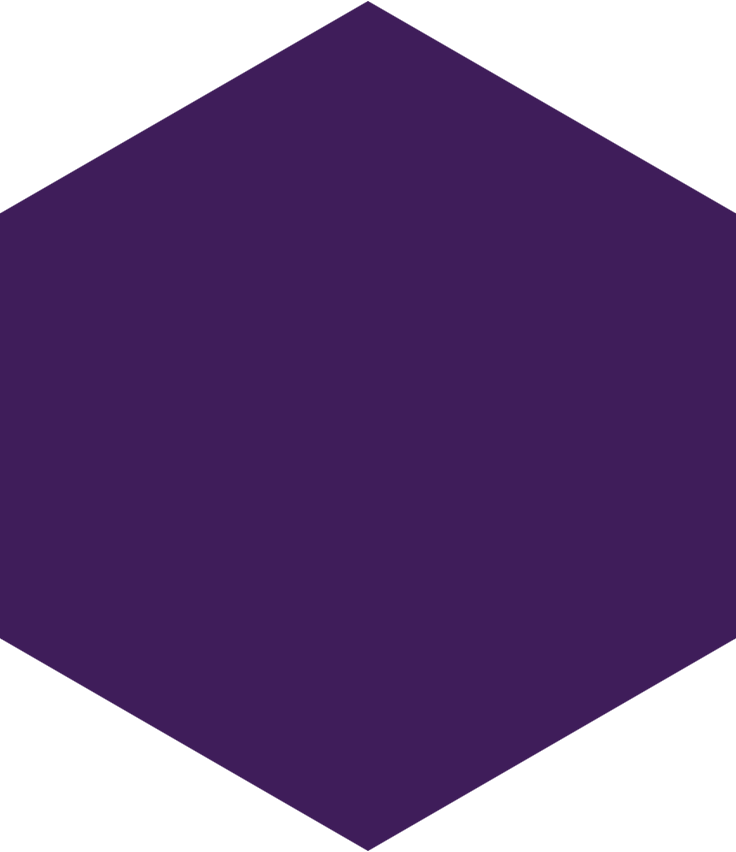


|  |
| --- |
| **A Comparative Analysis of**  **Apriori Algorithm and FP-Growth Algorithm**  **for Association Rule Mining** |
| Submitted to:Dr Hasan Mahmud |
| Computer Science and Engineering  Anindya Kumar Kundu | 2022-02-96-014  CSE522: Data Mining  Date of Submission: April 25, 2023 |
|  |





**Introduction:**

Association rule mining is a data mining technique used to discover interesting patterns and relationships in large datasets. It involves identifying frequent itemsets, subsets of items that occur together frequently, and generating association rules that express the relationships between them. Apriori and FP-Growth are two popular algorithms used for association rule mining. Apriori is a classical algorithm based on the concept of candidate generation and pruning, while FP-Growth is a more recent algorithm that uses a tree-based data structure for efficient mining. Both algorithms have their strengths and weaknesses, and their performance characteristics can vary depending on the characteristics of the dataset and the mining requirements. This report provides an in-depth analysis of the Apriori and FP-Growth algorithms, including their key concepts, steps, and performance characteristics, to help understand their similarities, differences, and applicability in different scenarios.

**Apriori Algorithm**

The Apriori algorithm is a classical association rule mining algorithm proposed by Agrawal and Srikant in 1994. It follows a two-step approach of candidate generation and pruning to mine frequent itemsets from a dataset. The "apriori" property states that if an itemet is infrequent, all its supersets are also infrequent.

The Apriori algorithm involves the following steps:

Frequent Itemset Generation: The algorithm starts by scanning the dataset to identify the frequent 1-itemsets, which are the individual items that occur with sufficient frequency. Then, it iteratively generates candidate k-itemsets by combining frequent (k-1)-itemsets, and prunes them based on the apriori property.

Support Counting: After generating the candidate itemsets, the algorithm scans the dataset again to count the occurrences of these candidate itemsets, and identifies the frequent itemsets based on a user-defined minimum support threshold. The support of an itemset is defined as the proportion of transactions in the dataset that contain the itemset.

Rule Generation: Once the frequent itemsets are identified, the Apriori algorithm generates association rules by considering all possible non-empty subsets of each frequent itemset, and calculates their confidence based on the support of the itemsets involved. A confidence threshold is used to filter the rules, and only those with confidence above the threshold are considered interesting.

The Apriori algorithm has several strengths, including its simplicity and ease of implementation, as well as its ability to handle datasets with large numbers of items and transactions. However, it also has some limitations. One major limitation is its candidate generation step, which can result in a large number of candidates, leading to high computational overhead. Another limitation is its need for multiple scans of the dataset, which can be time-consuming for large datasets.

**FP-Growth Algorithm**

The FP-Growth algorithm, proposed by Han et al. in 2000, is a more recent and efficient algorithm for association rule mining. It is based on a tree-based data structure called the FP-tree, which eliminates the need for candidate generation and pruning, making it faster than the Apriori algorithm.

The FP-Growth algorithm involves the following steps:

FP-Tree Construction: The algorithm scans the dataset once to construct the FP-tree, which is a compact and efficient data structure that represents the frequent itemsets in the dataset. The FP-tree is constructed by inserting each transaction into the tree in a way that preserves the itemset order and avoids the need for candidate generation. The tree is then pruned to remove infrequent items and branches with insufficient support, resulting in a compressed representation of the frequent itemsets.

FP-Tree Traversal: After constructing the FP-tree, the algorithm mines the frequent itemsets by recursively traversing the tree and constructing conditional FP-trees for each frequent item. The conditional FP-trees are constructed by pruning the tree based on the conditional support of the items, which is calculated from the original FP-tree. The mining process continues until no more frequent itemsets can be found.

Rule Generation: Once the frequent itemsets are identified, the FP-Growth algorithm generates association rules in a similar manner as the Apriori algorithm, by considering all possible non-empty subsets of each frequent itemset and calculating their confidence. However, unlike the Apriori algorithm, the FP-Growth algorithm does not require additional scans of the dataset for rule generation, as the frequent itemsets are already identified during the tree construction and traversal steps.

The FP-Growth algorithm has several strengths, including its efficiency and ability to handle large datasets with high dimensionality. It does not require candidate generation and pruning, which significantly reduces the computational overhead compared to the Apriori algorithm. Additionally, the FP-tree data structure allows for efficient tree-based mining, making it suitable for datasets with varying itemset sizes and support thresholds.

There are various metrics in place to help us understand the strength of association between antecedent and consequent:

* Support
* Confidence
* Lift or Correlation or interest
* Leverage
* Conviction

**Support**

It gives an idea of how frequent an itemset is in all the transactions. To say in formal terms, it's the fraction of the total no. of transactions in which the itemset occurs. We refer to an itemset as a "frequent itemset" if your support is larger than a specified minimum support threshold.

Range:[0,1] Value of support helps us identify the rules worth for future analysis.

**Confidence**

It defines the likelihood of occurrence of consequent on the cart given that cart already has antecedent. It signifies the likelihood of item Y being purchased when item X is purchased.

Range:[0,1]

If confidence is 0.75 then that implies that 75%of transactions containing X also contain Y .It can also be interpreted as the conditional probability P(Y|X), i.e, the probability of finding the itemset Y in transactions given the transaction already contains X.

It has a major drawback i.e. It only takes into account the popularity of the itemset X and not the popularity of Y. If Y is equally popular as X then there will be a higher probability that a transaction containing X will also contain Y thus increasing the confidence. To overcome this drawback there is another measure called lift.

**Lift**

Lift gives rise in the probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without knowledge about presence of {X}.

Range:[0,Infinity]

It can simply be considered as correlation between the antecedent and consequent. If the value of lift is greater than 1, it means that the itemset Y is likely to be bought with itemset X, while a value less than 1 implies that itemset Y is unlikely to be bought if the itemset X is bought.

**Levarage or Piatetsky-Snapiro**

It computes the difference between the observed frequency of X & Y appearing together and the frequency that we would expect if A and C are independent.

Range:[-1,1]

If X,Y are positively correlated then we get leverage>0 ,we need such type of rules.  
If X,Y are negatively correlated then we get leverage<0.  
If X,y are independent , then we get leverage = 0.

**Conviction**

It can be interpreted as the ratio of the expected frequency that X occurs without Y (that is to say, the frequency that the rule makes an incorrect prediction) if X and Y were independent divided by the observed frequency of incorrect predictions.

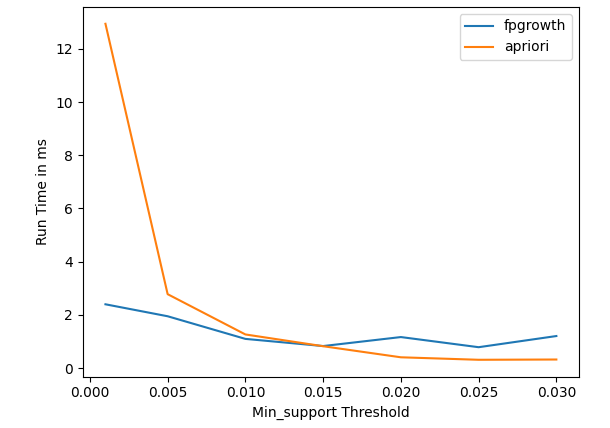
Conviction(X−>Y)=support(Y)confidence(X−>Y)

**Comparative analysis for different Dataset**

Dataset 02: Groceries dataset

The dataset has 38765 rows of the purchase orders of people from the grocery stores. These orders can be analysed and association rules can be generated using Market Basket Analysis by algorithms like Apriori Algorithm. The "Groceries" dataset is sparse nature, as customers typically purchase only a small fraction of the total available items in a grocery store. This sparsity poses challenges in data analysis and modeling, as it requires careful handling of missing values and zero values. The "Groceries" dataset also exhibits temporal properties, as it includes information on the transaction date. This temporal dimension allows for the exploration of seasonal patterns, trends, and changes in customer purchasing behavior over time, which can be useful for understanding customer preferences and informing marketing strategies.

The runtime results of Apriori and FP-Growth algorithms on the "Groceries" dataset with varying minimum support values (.001, .005, .01, .015, .02, .025, .03) reveal interesting insights.



*Fig01: Apriori vs FPGrowth runtime on different Minimum support threshold for Groceries dataset*

First, it is evident that as the minimum support value increases, the runtime of both algorithms decreases. This is expected, as higher support thresholds result in fewer frequent itemsets to be generated, reducing the computational burden. Apriori consistently takes more time compared to FP-Growth for all support values, which can be attributed to its exhaustive candidate generation and itemset counting steps.

Moreover, it is observed that the runtime of Apriori decreases significantly with higher support values, while FP-Growth shows relatively stable performance across support values. This is because Apriori's performance heavily depends on the number of candidates itemsets, which decreases with higher support thresholds, resulting in faster execution. On the other hand, FP-Growth's performance is not affected by the number of frequent itemsets, as it uses a compact tree-based structure for efficient itemset mining.

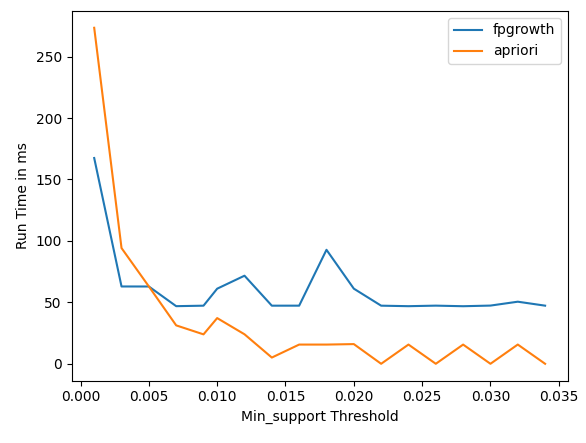
Another noteworthy observation is that the runtime of both algorithms varies across different support values. For instance, Apriori shows a significant decrease in runtime from .001 to .01, but the reduction slows down after that. FP-Growth, on the other hand, has slightly higher runtime for .015 and .03 compared to its neighboring support values.

Overall, the runtime analysis of Apriori and FP-Growth on the "Groceries" dataset suggests that higher minimum support values result in faster execution for both algorithms. However, Apriori's performance is more sensitive to the support threshold, while FP-Growth demonstrates relatively stable performance. Data analysts can use these insights to select an appropriate minimum support value based on their specific requirements for efficient and effective market basket analysis on the "Groceries" dataset.

Dataset 01: The Bread Basket

The "Bread Basket" dataset, available on Kaggle, is a fictional transaction dataset from a bakery that contains information about the items purchased, transaction IDs, and timestamps with 20507 rows. This dataset is commonly used for market basket analysis, which involves finding patterns and associations between items frequently purchased together.

In order to perform market basket analysis, two popular algorithms, Apriori and FP-Growth, were applied to this dataset. The experiments were conducted with different support values, which determine the minimum frequency threshold for an itemset to be considered "frequent". Lower support values represent less frequent itemsets, while higher support values represent more frequent itemsets. The results revealed that Apriori took more time to run with lower support values, as it requires multiple scans of the dataset, while FP-Growth was faster as it constructs a compact tree structure.



*Fig02: Apriori vs FPGrowth runtime on different Minimum support threshold for bread basket*

Interestingly, it was observed that Apriori performed better with higher support values. This is because with lower support values, the algorithm generates a larger number of candidates itemsets, resulting in a larger search space and longer execution time. However, with higher support values, the number of frequent itemsets is reduced, resulting in a smaller search space and faster execution time for Apriori.

**Conclusion:**

In conclusion, both the Apriori and FP-Growth algorithms are popular and widely used for association rule mining tasks. While the Apriori algorithm is a classic approach that is relatively simple to implement, it can suffer from computational overhead and scalability issues for large datasets. On the other hand, the FP-Growth algorithm is a more recent and efficient algorithm that uses a tree-based mining approach and eliminates the need for candidate generation and pruning, making it faster and more memory-efficient. However, it may be more complex to implement and less flexible in handling dynamic support thresholds. Both algorithms have their strengths and weaknesses, and the choice between them depends on the specific requirements of the task at hand, such as dataset size, dimensionality, and support threshold flexibility. Additionally, a hybrid approach that combines the strengths of both algorithms can also be used in certain scenarios.