Market basket analysis

The Apriori Algorithm is a basic algorithm proposed by Agrawal & Srikant in 1994 for the determination of the frequent itemset for Boolean association rules. The principles of Apriori state that “if an itemset is frequent, then all its subset items will be frequent”. If the support for the itemset is more than the support level, the itemset is “frequent”. The algorithm is based on the prediction of items, which move from the previous stage on a regular basis. The name derived from the term "prior". Apriori algorithm includes the type of association rules in data mining. The rule that states associations between multiple attributes is often called affinity analysis or market basket analysis.

The Apriori algorithm works in two steps:

Prune and Join:

1.Generate all frequent item sets – A frequent item set is an item set that has transaction support above minimum support.

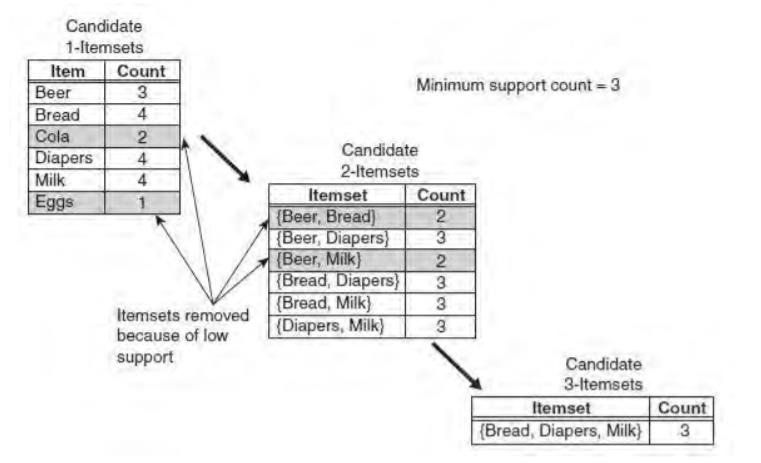
2.Generate all confident association rules from frequent item sets – A confident association rule is a rule with confidence above minimum confidence.

To apply Apriori algorithm on Instacart dataset, the Apriori class is applied that is imported from the Apyori library.

• k-itemset is the itemset which contains: k element number.

The Apriori function reduces the number of items to be searched to find frequent sets of items. This algorithm continues to identifying 2-itemsets using 1-itemsets, and 3-itemsets using 2-itemsets in an iterative way. This can be generalized as follows; frequent item sets

(k-1) elements are used to find frequent item candidates for k elements.



In the above instance, each item is initially considered as a 1-point candidate. Thereafter we count on their support and the itemset appearing less than minimum support count was discarded. As a consequence {Cola} and {Eggs} are removed. In the following iteration, candidate 2-itemset is generated with the help of frequent 1-itemset, because the Apriori principle ensures that all supersets of the rare 1-itemsets must be rare.

The number of candidate 2-itemsets generated by the algorithm is (4C2) = 6, because there are only four frequent 1-itemsets. The performance of the pruning strategy can be demonstrated by counting candidate generated item sets. A brute -force strategy to list al item sets (up to length 3) as candidates will give in 41 candidates.

**Frequent Pattern Growth Algorithm :**

The FP-Growth algorithm offers an alternate means of measuring a frequent item collection using an FP-Tree graphic data structure to compact transaction records. One can think of FP-Tree as turning the datasets into a graph format. Instead of the generation and check method used in the Apriori algorithm, FP-Growth generates the FP-Tree first, and uses this compact tree to produce the regular itemset. The FP-Growth algorithm's efficiency depends as to how much compression can be performed while generating the FP-Tree.The FP-Growth method transforms the problem of repeating the search of the minors and then combining the suffixes in the discovery of broad specific models. With the use of having slightly repetitive objects as a suffix it provides strong efficacy. This approach decreases search costs significantly.

**FP-Tree representation:**

A FP-tree is a compact data structure representing a collection of tree-shaped records. Every transaction is read out and sorted to an FP-tree path. This will come into force until all transactions are read out. The tree remains compact because the paths overlap by different transactions which have common subsets.

FP-Growth algorithm is the main execution mechanism [43];

1. Initially, scan the database and you can find items equal to and above the threshold value.

2. Support values for specific products are displayed in a size (large to small) order.

3. It then produces a tree with only roots. Shruthi Gurudath 2990078

4. The database is re-scanned for each sample;



A data collection of 5 transactions and five items can be seen in the diagram. The FP tree structures also appear in the figure after taking the first three transactions. Each node in the tree includes an item's label and also a tracker that represents the number of transactions that have taken the specified path. The way the FP-tree is generated is illustrated below:

1. The data is scanned at first to produce the support value for each item. Items which are not frequent are removed, but at the other side items which are frequent are organized in decreasing order. The figure above shows that a has been the most common item, then c, then d and ultimately e.

Shruthi Gurudath 2990078

2. The algorithm then crosses the data again for the FP-tree structure. The nodes a and b are generated after reading the first transaction {a, b}. The transaction in a tree is then generated from root- > a-> b. Now every node has its count value.

3. Then new nodes are created to represent b, c , and d when the second transaction is crossed {b , c, d}. Then a path is formed by the connection of the b , c and d nodes (root->b->c->d). Whereas the first two transactions involve b, these will not connect because they have a separate predecessor.

4. Then perhaps the third transaction {a, c, d , e} has an initially transacted common predecessor. As long as the item predecessor matches, the path overlaps.

5. The same process goes on until all data is put into the FP-tree.