Data Analytics

Frequent Itemset Mining

Mehul Goyal - 2018101018

Tushar Joshi - 2018101013

DataSet Used: Retail, Sign, MSNBC

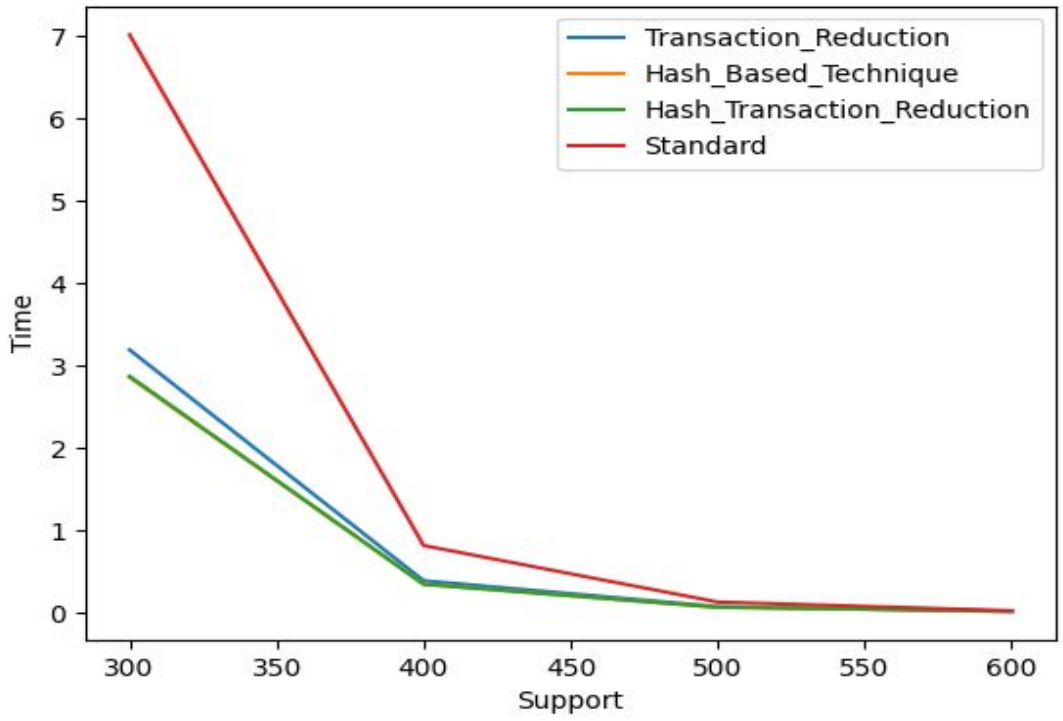
Apriori Algorithm

Strategies Implemented

- Transaction Reduction
  - It is based on the fact that a transaction that does not contain any frequent k-itemsets cannot contain any frequent (k + 1) - itemsets. Therefore, such a transaction can be removed from further consideration.
  - We sorted our database according to the length of the transactions, whenever in an iteration if we are going to count the k-frequent itemsets we first remove the transactions whose item-sets length is less than k.
  - Then from Lk we calculate Lk+1 frequent itemsets by generating candidates and counting them in the transaction list.
  - This way database size gets reduced for further iterations and thus a good optimisation to naive apriori algorithm.

- Hash Based Technique
  - When scanning each transaction in the database to generate the frequent 1-itemsets, L1 we can generate all the frequent 2-itemsets for each transaction, hash them into the different buckets of a hash table structure, and increase the corresponding bucket counts. A 2-itemset with a corresponding bucket count in the hash table that is below the support threshold cannot be frequent and thus should be removed from the candidate set.
  - We iterate through all the transitions in the database and remove the 1 - itemsets which are not frequent from each transaction . For each transaction we then form all the 2 - candidates and hash them into the table with the help of hash function and increment their count.
  - Hash Function : (sum of ascii values of both items)%p ,
    p - > Prime Number. Eg -> Hash(Milk,Jam) , p =7 => 5
  - The ones whose bucket count greater than or equal to minSupport are a part of frequent 2 - itemsets.

Sign Dataset



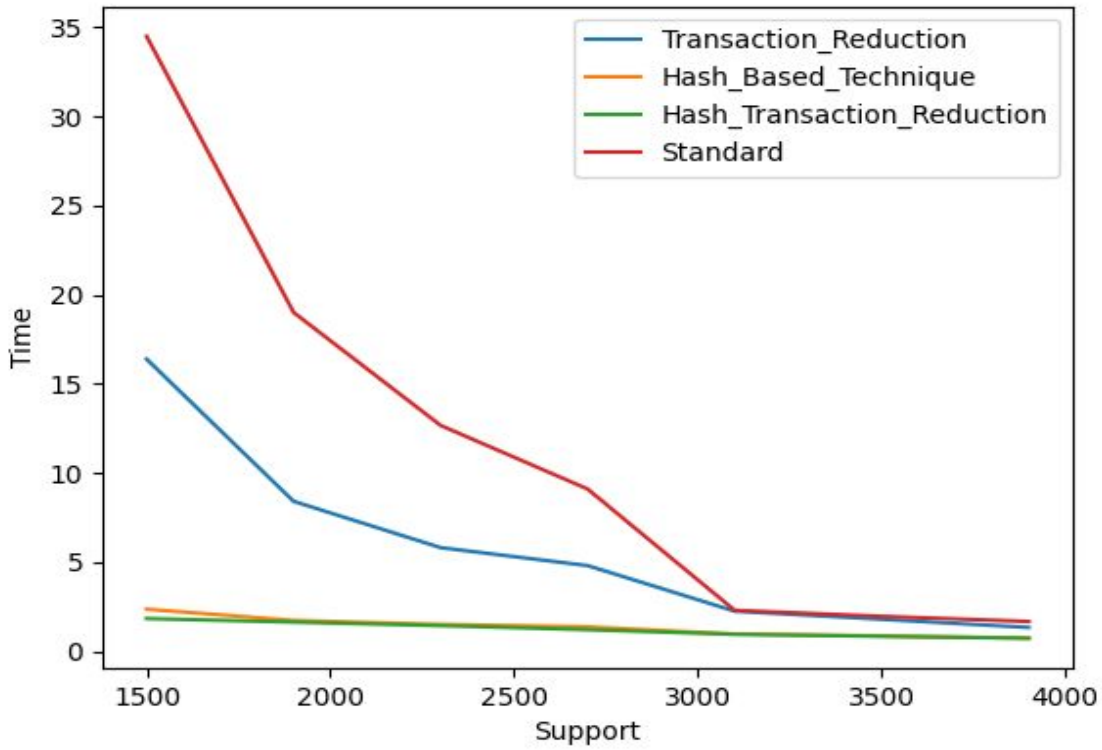
Observations

- Hashing and Transaction Reduction techniques produce significant improvement in the sign dataset for low support count. Optimised algorithms give results in almost half the time.

- Hash Based approach performs almost the same as combination of both(hidden behind green line).

- In this dataset the average length of the transactions is less because, on increasing support count transaction reduction technique behaves good and hence some optimization happens and total items are also less because almost same optimisation of both algorithms.

Retail Dataset



Observations

- Hashing and Transaction Reduction techniques combined produce significant improvement in the Retail dataset especially at low support count optimisations give much faster results. Naive gives result in about 35 sec and optimised techniques gives in less than 5 seconds.

- Transaction Reduction is slower than a hash based approach.

- In this dataset not many transactions get deleted until the later stages of the algorithm. More Items are there in the dataset. Frequent Items are distributed between various transactions. Number of unique elements is less as compared to transactions.

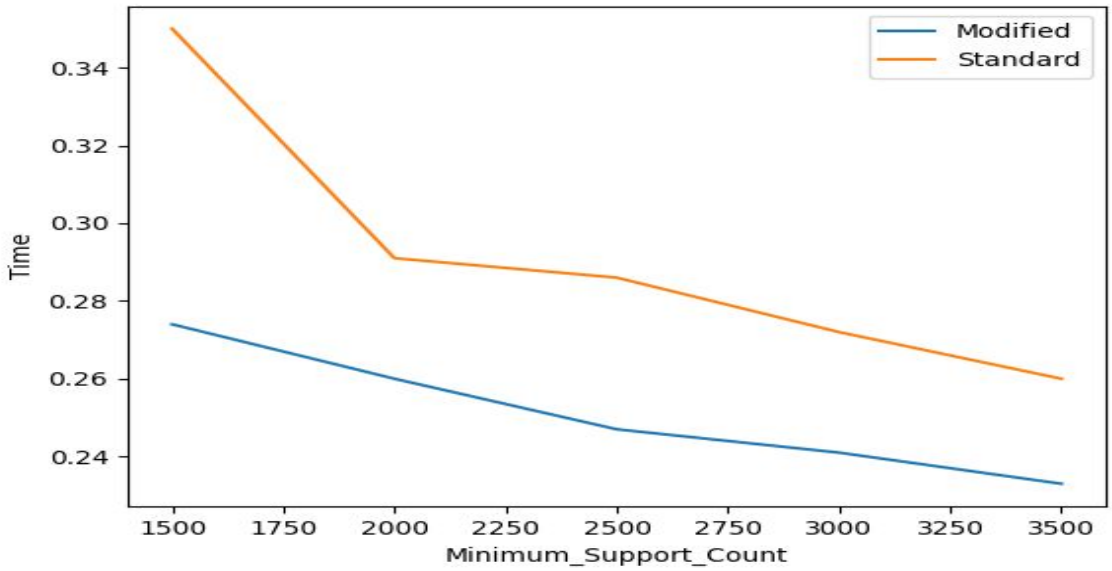
Frequent Pattern Growth

Strategy Implemented Merging

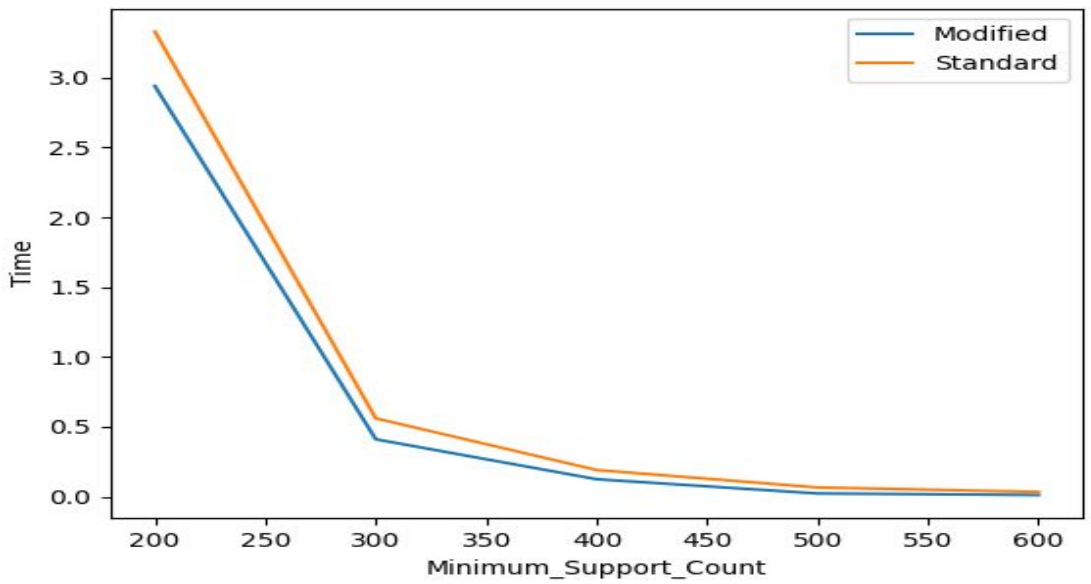
- We have started conditional tree generation from the top of the FP tree because if we use a bottom to up approach , we would need to store extra information about linking nodes and parent nodes and this would increase memory usage of the algorithm.

- We start depth first search from the root node , and as we move down we know that nodes that are in the path from root to current node would appear in the conditional pattern base of the current node. In conventional approach we move from leaf node to root and while finding the pattern base of node lying in the path we again traverse same nodes. But in our implementation we traverse a path from root to leaf only once , this is a merging strategy.

MSNBC dataset



Sign dataset



Observations

- Difference in performance of the modified algorithm and the standard algorithm will decrease , as the number of nodes in the fp tree decreases. We can observe this from the graph , with increase in minimum support count , the number of nodes in fp tree decreases so the gap between modified and standard algorithm decreases.

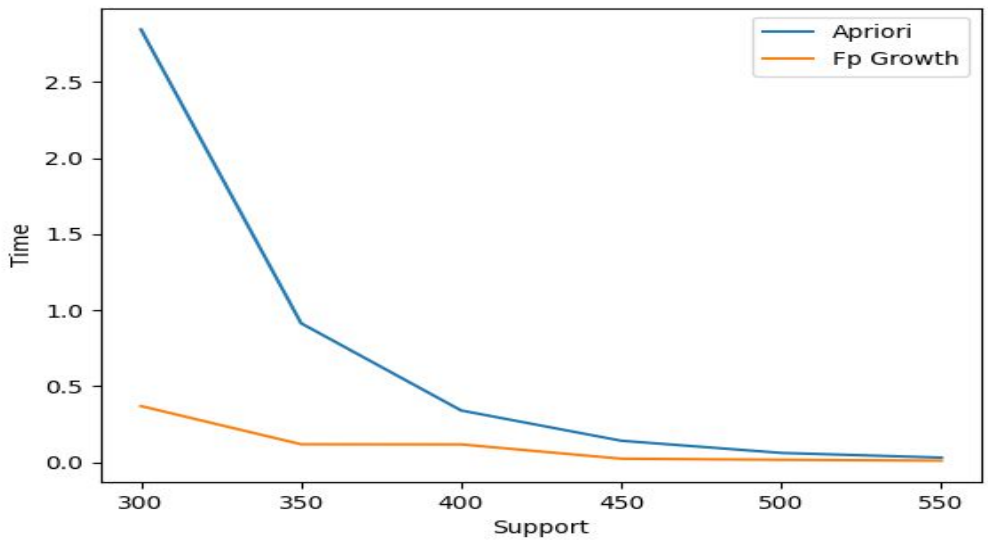
- The gap in performance of algorithms is more in case of sign data set as compared to that in MSNBC dataset . The number of different items is more in Sign dataset as compared to MSNBC dataset , so the number of nodes in fp tree will be more in Sign dataset. Since , the number of nodes in fp tree is more in Sign dataset , performance gap is more in case of Sign dataset than MSNBC.

Comparative Analysis

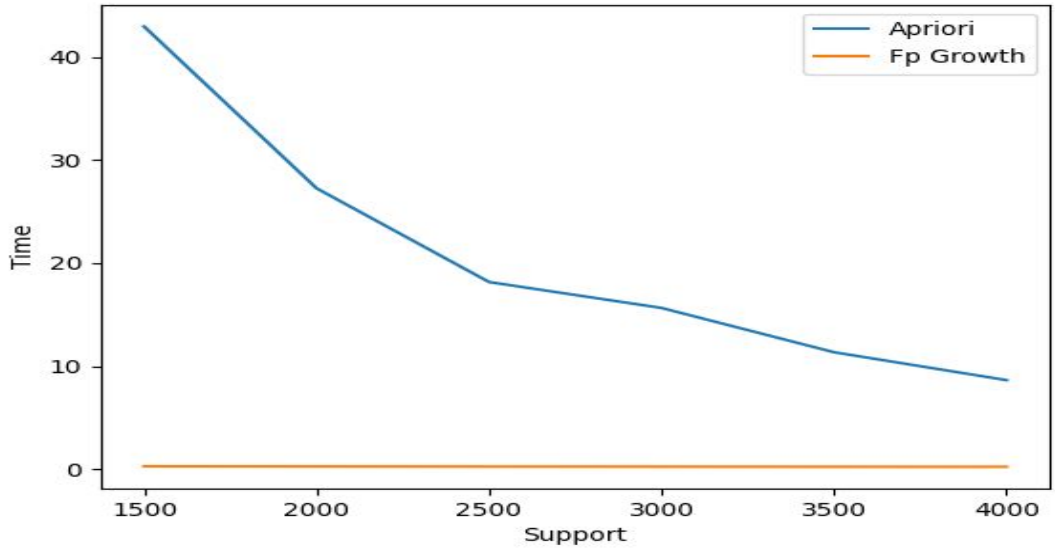
- Apriori is exponential w.r.t time and support. If a lower support leads to a huge number of frequent itemsets then apriori tends to become very slow. If the number of unique elements is large then the 2-itemset generation becomes very slow since it is O((Items)^2) w.r.t the number of unique elements**.**

- The FP algorithm is faster as compared to Apriori. The runtime of algorithm changes linearly with change in the number of itemsets. There is no candidate generation in FP, the database is fragmented using one frequent item. The itemsets of these fragmented patterns are analyzed. Thus with this method, the search for frequent itemsets is reduced comparatively.

Sign Dataset



MSNBC Dataset



Observations :

- In the MSNBC dataset Fp growth performs much better than Apriori. This tells that the database is too dense which means that there are more transactions than the number of items itself in MSNBC.

- In the Sign dataset there is not much difference between performance of Apriori and Fp growth this tells that the dataset is sparse , that is number of items is comparable to number of transactions.

- At higher values of support count both algorithms take almost equal time which is expected because performance of apriori improves exponentially ,when the number of frequent itemsets decreases.