Programming Assignment - 1
Data Mining (Association Rule Mining)

Problem - 1

In this assignment, I implemented all the 3 algorithms (Apriori, FP-Tree, and Eclat algorithm). In the Apriori algorithm,

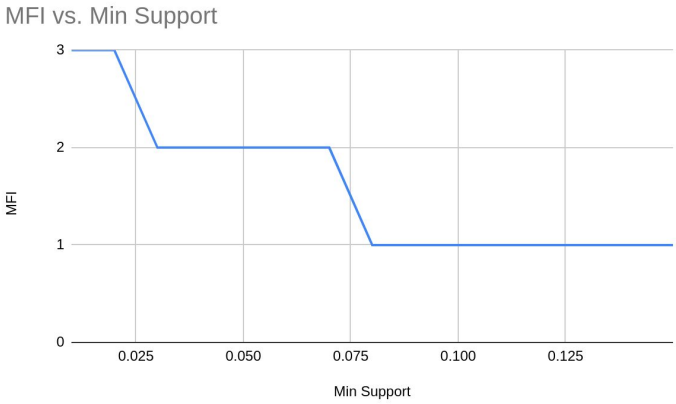
a) Apriori Algorithm - I improved the algorithm by pruning all the items whose subsets were not available in the previous layer. I also used the dictionary data structure in python language (which can find any item in O(1) (average)) for the candidate itemset. Initially, I also used the Hash-Tree algorithm to count the support for each candidate item in one go but due to the small size of maximal frequent itemsets, it was only increasing the complexity of the algorithm due to which I used normal iteration through all transactions for counting the support of each itemset.
b) FP-Tree Algorithm - In this algorithm, I created a class for the FP tree node, which contains a dictionary for pointing at each of the children due to which we can find any child of any node in O(1) time. I also removed all the itemsets whose support was less than the minimum support before building the FP-tree to avoid memory crashes.
c) Eclat Algorithm - I improved this algorithm by using diffset instead of Tidsets because this way the algorithm works much faster by shrinking the size of the intermediate sets. I also use sets implementation in python to find the difference in two diffsets. For finding the items transactions, I removed all the items having support less than the minimum support to reduce the number of items in consideration

In this assignment, we worked on a total of four datasets including T10I4D100K, T40I10D100K, retail, and groceries dataset. All these datasets were given some transactions with each transaction containing some number of items that were brought in that transaction.

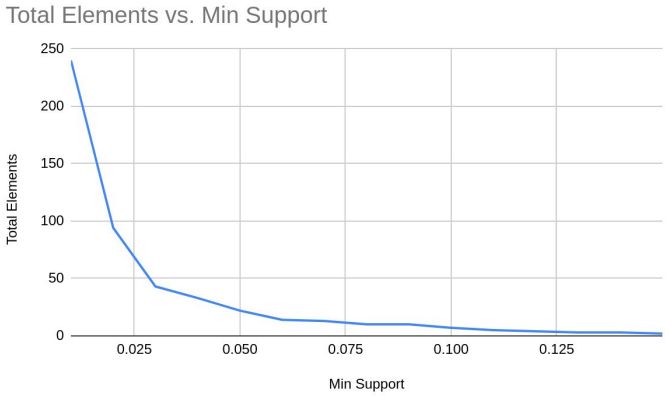
a) T10I4D100K - Number of transactions: 100000
   Size of maximal frequent itemset: 3 (for minimum support=0.008)
   Total number of items: 1000
   The average width of transitions: 10.1028
b) T40I10D100K - Number of transactions: 100000
   Size of maximal frequent itemset: 18 (for minimum support=0.008)
   Total number of items: 1000
   The average width of transitions: 39.6057
c) Retail - Number of transactions: 88162
   Size of maximal frequent itemset: 4 (for minimum support=0.008)
   Total number of items: 16469
   The average width of transitions: 10.3057
d) Groceries - Number of transactions: 9835
   Size of maximal frequent itemset: 4 (for minimum support=0.008)
   Total number of items: 167
   The average width of transitions: 3.4094

Next, We compare the tree algorithm on the basis of the time taken and we also see the different properties of these datasets.

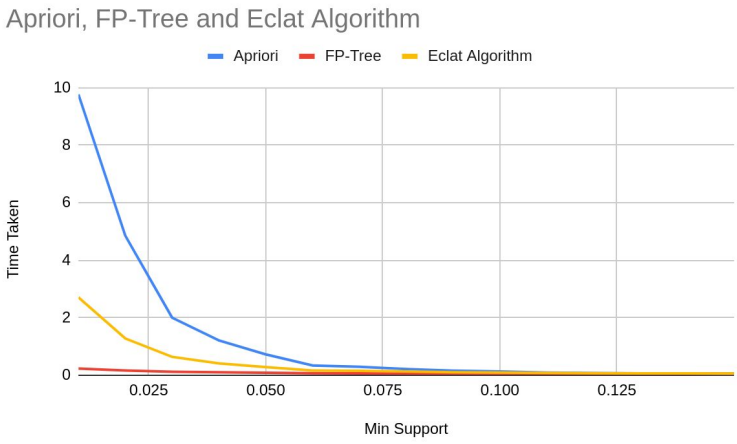
1) Groceries.csv



This is the graph between the minimum support and the size of the maximum frequent itemset.



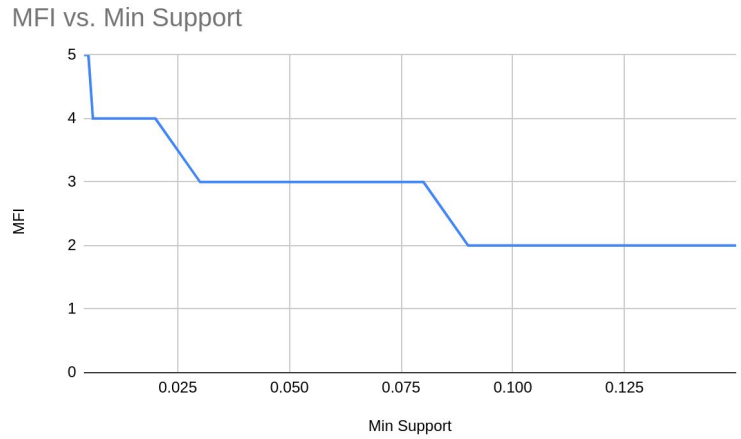
Number of frequent itemsets vs the minimum support



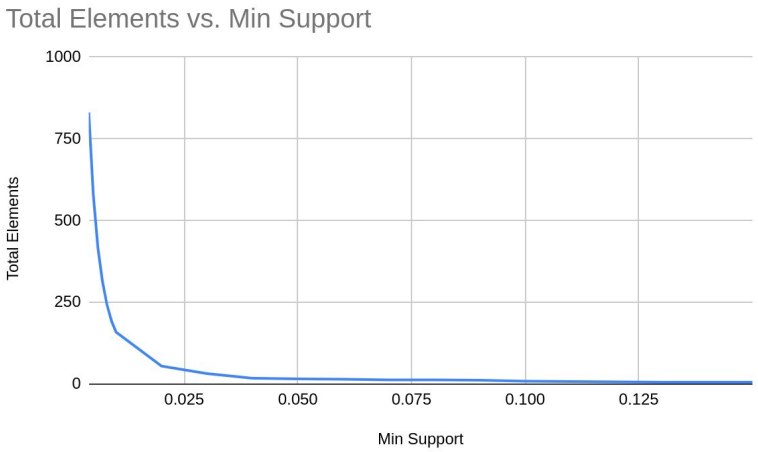
Time Taken vs the minimum support for all three algorithms. Red - FP-Tree Algorithm, Yellow - Eclat Algorithm, and Blue - Apriori Algorithm

From the above graphs, we can see that the size of the maximal frequent itemset of the Groceries.csv dataset is 3 at a minimum support of 0.01. Initially, the size of the maximal frequent itemset for the dataset is 1 and then it increases stepwise. We can clearly see that the number of frequent itemsets increases exponentially as we decrease the minimum support for the algorithms.
Next, we observe that the time taken by all the algorithms is almost similar for large minimum support, and it increases exponentially for all of them as we decrease the value of the minimum support. We observe that the time taken by the apriori algorithm increases much faster as compared to the other two algorithms. Major jumps in the time taken are observed when the size of the maximal frequent itemsets increases. FP-Tree is the fastest algorithm and it performs the best for this dataset followed by the Eclat algorithm followed by Apriori.

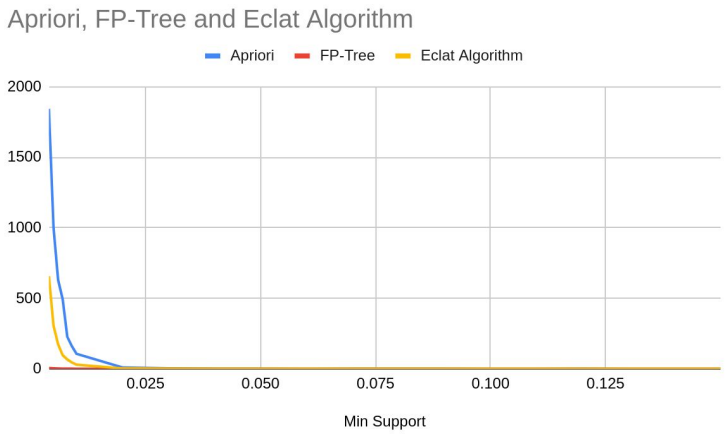
2) Retail.txt



This is the graph between the minimum support and the size of the maximum frequent itemset.



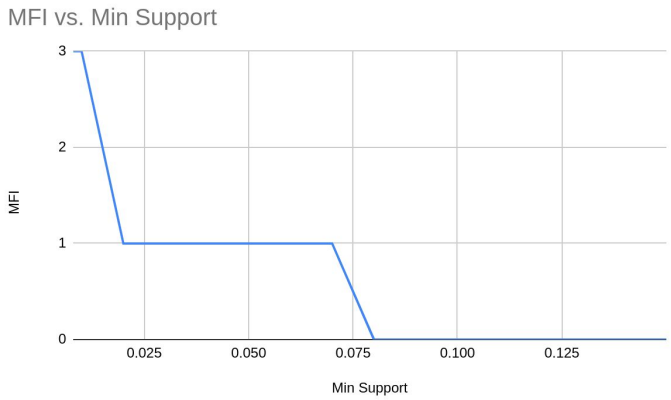
Number of frequent itemsets vs the minimum support



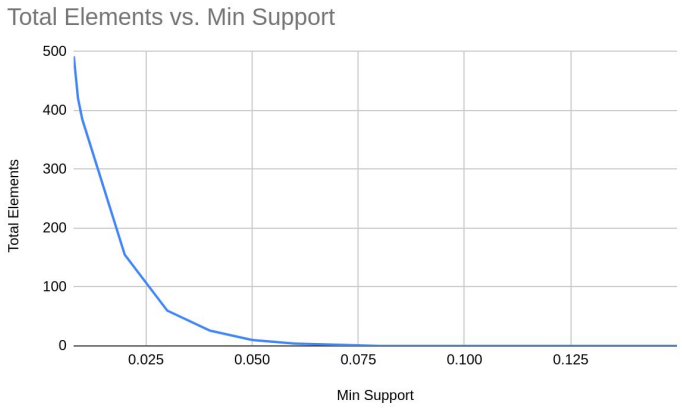
Time Taken vs the minimum support for all three algorithms. Red - FP-Tree Algorithm, Yellow - Eclat Algorithm, and Blue - Apriori Algorithm

From the above graphs, we can see that the size of the maximal frequent itemset of the Groceries.csv dataset is 5 at a minimum support of 0.004. As we decrease the minimum support we can clearly see the size of frequent itemset increasing exponentially. It is also the expected result because as we decrease the minimum support the chances of the itemset being a frequent itemset increase.
Next, we observe that the time taken by all the algorithms is almost similar for large minimum support, and it increases exponentially for all of them as we decrease the value of the minimum support. We observe that the time taken by the apriori algorithm increases much faster as compared to the other two algorithms. Apriori algorithm takes a lot of time compared to other algorithms because of the fact it has to traverse across all the transactions. Eclat algorithm just iterates across all the items which are much less than the total number of transactions. Therefore it is faster than the Apriori algorithm,. FP-Tree is the fastest algorithm and it performs the best for this dataset followed by the Eclat algorithm followed by Apriori.

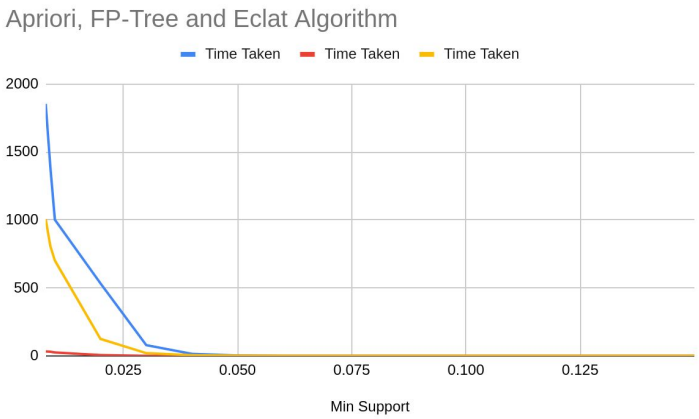
3) T10I4D100K.txt



This is the graph between the minimum support and the size of the maximum frequent itemset.



Number of frequent itemsets vs the minimum support

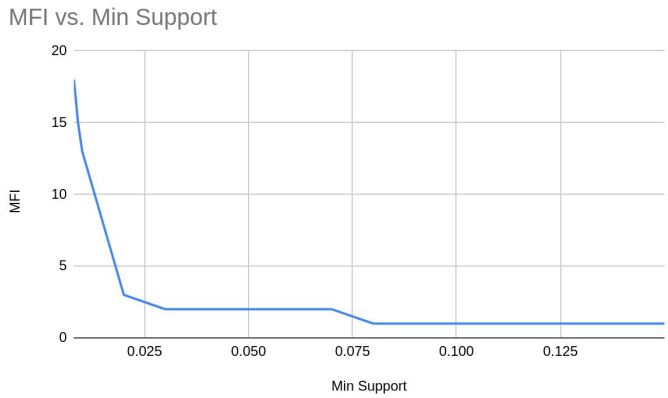


Time Taken vs the minimum support for all three algorithms. Red - FP-Tree Algorithm, Yellow - Eclat Algorithm, and Blue - Apriori Algorithm

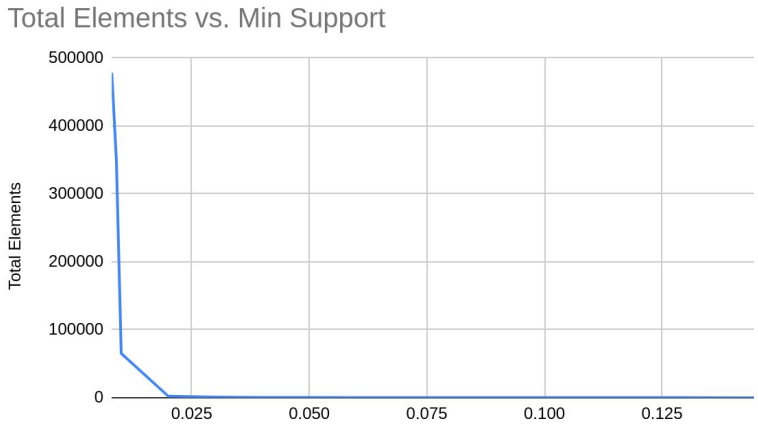
From the above graphs, we can see that the size of the maximal frequent itemset of the Groceries.csv dataset is 3 at a minimum support of 0.008. The size of the maximum frequent itemset is so low for 100000 transaction itemset because of the fact each transaction contains only 10 items due to which it is much more difficult for multiple items to occur concurrently. As we decrease the minimum support we can clearly see the size of frequent itemset increasing exponentially.

Next, we observe that the time taken by all the algorithms is almost similar for large minimum support, and it increases exponentially for all of them as we decrease the value of the minimum support. We observe that the time taken by the apriori algorithm increases much faster as compared to the other two algorithms. Apriori algorithm takes a lot of time compared to other algorithms because of the fact it has to traverse across all the transactions. Eclat algorithm just iterates across all the items which are much less than the total number of transactions. Therefore it is faster than the Apriori algorithm,. FP-Tree is the fastest algorithm and it performs the best for this dataset followed by the Eclat algorithm followed by Apriori. FP-Tree algorithm has to store all the transactions in its tree due to which it sometimes on some computers end with memory exhausted error. I ran it on 12GB ran.

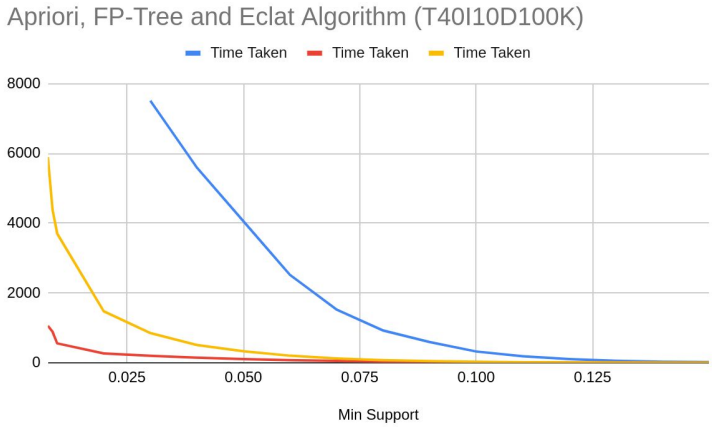
4) T40I10D100K.txt



This is the graph between the minimum support and the size of the maximum frequent itemset.



Number of frequent itemsets vs the minimum support



Time Taken vs the minimum support for all three algorithms. Red - FP-Tree Algorithm, Yellow - Eclat Algorithm, and Blue - Apriori Algorithm

From the above graphs, we can see that the size of the maximal frequent itemset of the Groceries.csv dataset is 18 at a minimum support of 0.008. As we decrease the minimum support we can clearly see the size of frequent itemset increasing exponentially. In this case, the size of the maximal frequent itemset also increases exponentially because each transaction contains on average 40 items which give rise to a high number of common items. The total number of frequent items explodes as the minimum support is decreased below 0.011 and finally at 0.008 we observe 478374 frequent itemsets.
Next, we observe that the time taken by all the algorithms is almost similar for large minimum support, and it increases exponentially for all of them as we decrease the value of the minimum support. We observe that the time taken by the apriori algorithm increases much faster as compared to the other two algorithms. Apriori algorithm takes a lot of time compared to other algorithms because of the fact it has to traverse across all the transactions. As the size of itemsets increases the number of iterations over the transactions increases due to which the time taken increases exponentially Eclat algorithm just iterates across all the items which are much less than the total number of transactions. Therefore it is faster than the Apriori algorithm,. FP-Tree is the fastest algorithm and it performs the best for this dataset followed by the Eclat algorithm followed by Apriori.
Please note that because of the large amount of time taken by the apriori algorithm we are not able to find the time taken by the algorithm for very low minimum support.

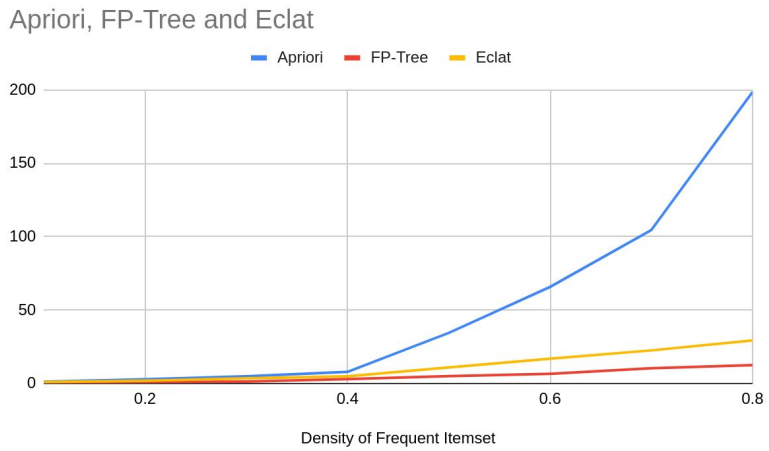
Next, we discuss the advantages and disadvantages of the algorithms. Apriori algorithm is the simplest to implement with very little complexity and it works well for fewer transactions but as the number of transactions increases the time taken by the algorithm also increases exponentially. Eclat Algorithm iterates on the number of items in the dataset because of which it was observed to be faster than the Apriori algorithm in all the above cases. In case, where there are many items and fewer transactions Apriori Algorithm will be faster than the Eclat algorithm. Finally, the FP-Tree algorithm is the fastest algorithm of the three as it uses the tree structure ti evaluate the frequent itemset. The major drawback of this algorithm is that it includes all the transactions in the tree structure due to which the memory consumption of this algorithm is very high.
As discussed in the previous section all algorithms have their advantages and disadvantages it is the dataset on which it depends which algorithm to apply.

Problem - 2

In this problem, we implemented an algorithm that took the number of transactions, size of maximal frequent itemset, the average width of each transaction, and the total number of items as input and produced a fake dataset.
We fix the density of frequent itemset to 0.2 for matching the input given by the user. (You can change it within the code.)

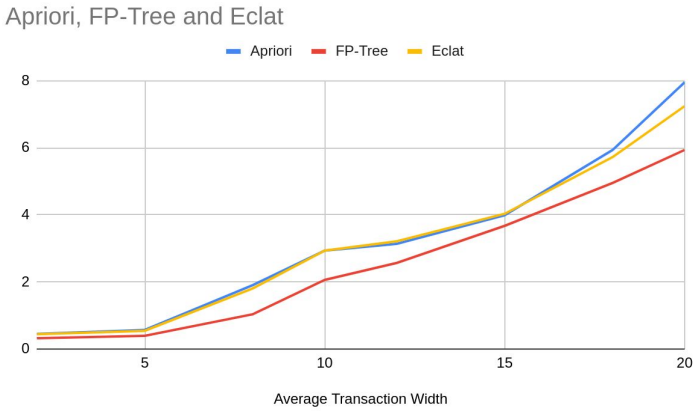
The following are some graphs on different measures for developing the datasets vs the time taken by the three algorithms.

1) Time Taken by different algorithms vs The density of maximal frequent itemsets



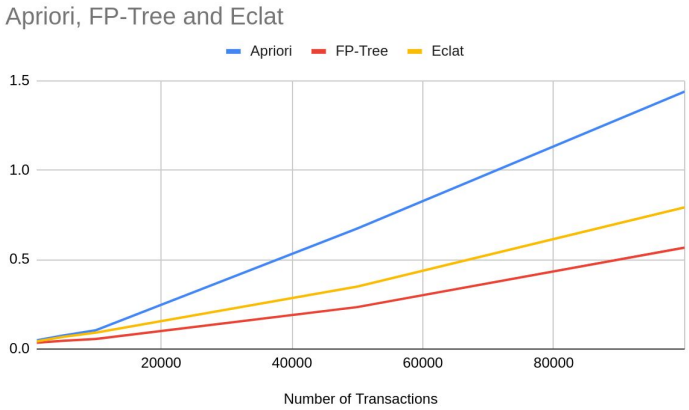
As we have also talked about in the previous problem the time taken by the apriori algorithm drastically increases as we increase the size of the itemset. In the above figure, we observe that initially till 0.4 density of frequent itemset all algorithms perform similarly but as we increase the frequency of frequent itemsets time taken by the apriori algorithm increases drastically. Fp-Tree algorithm performs best out of 3 algorithms and the time taken by the Eclat algorithm is almost equal to the time taken by the FP-Tree Algorithm.

2) Time Taken by different algorithms vs The Average width of the Frequent Itemsets



In the above figure, we observe that increasing the average width of each transaction increases the time taken for all the algorithms. Time taken by all the algorithms is almost similar to the minimum time taken by the Fp-Tree algorithm followed by the Eclat algorithm followed by the Apriori algorithm.

3) Number of Transactions vs The time Taken by the algorithms



As we increase the number of transactions time taken by each of the algorithms increases linearly. We test this dataset only for 5 values including 1000, 5000, 10000, 50000, 100000. FP-Tree algorithm performs better than all the other algorithms in terms of the time complexity.