# Final Report

1) Data Structures Used:

**Reading**: We used **List** to store the transactions read from the input file.

**Preparing Dataset**: We used **Data Frame** to store the transactions in the in the form of rows and items in the form of columns (representing in the form of asymmetric binary attributes) which we can later use to find the support in the following way. Sum of a single column will give the support of that item and sum of multiple columns will give the support of itemset.

Ex: If we want to find out the support of an itemset {‘1’,’2’,’3’} then we can select the rows from ‘1’,’2’,’3’ columns such that all the columns have 1 value. And finally count the rows so we get the support of {‘1’,’2’,’3’}.

**Generating Frequent 1-itemsets**: We used a **set** to store each item and then used **data frame** to find its support by summing up the column. We have the item and its support eliminate the itemset not satisfying minimum support. And now we used a **dictionary** to store them as key and value pair.

**Candidate Generation**: In order to generate kth candidates, we used Fk-1 x Fk-1. In a loop We used **set** operation union and checked its length is equal to k if satisfied then we added it to the **List** and then performed pruning on it. But to store them in dictionary the items should be immutable, so we use **Frozenset()** function.

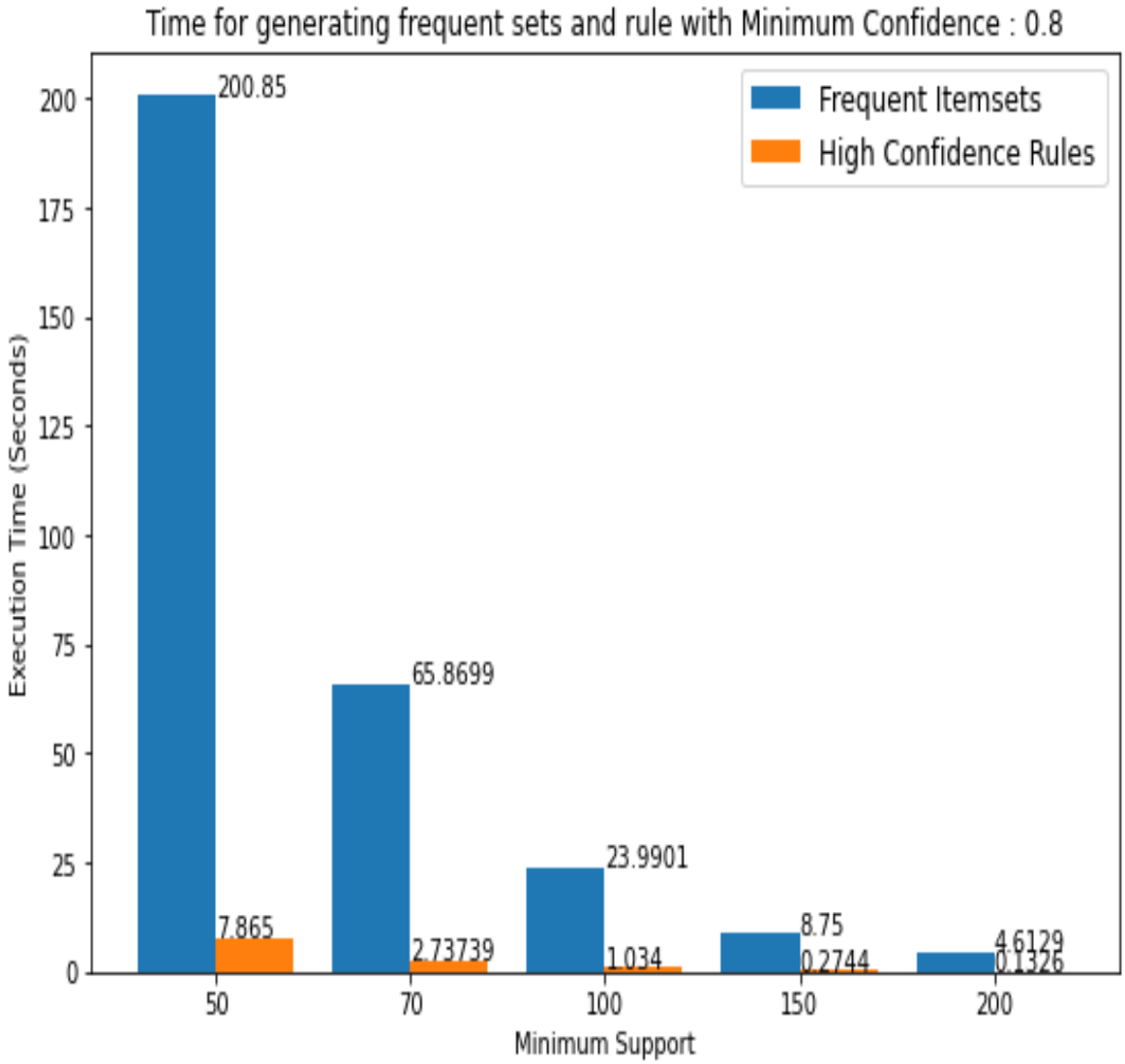
**Pruning**: After candidate generation we checked immediate subsets of each itemset in candidate generation from **List**. We generated k-1 all possible **subsets** of itemset using **itertools combination** and stored in a **List**. Now we checked every combination is frequent or not. If atleast one of them, is not frequent then we eliminate itemset.

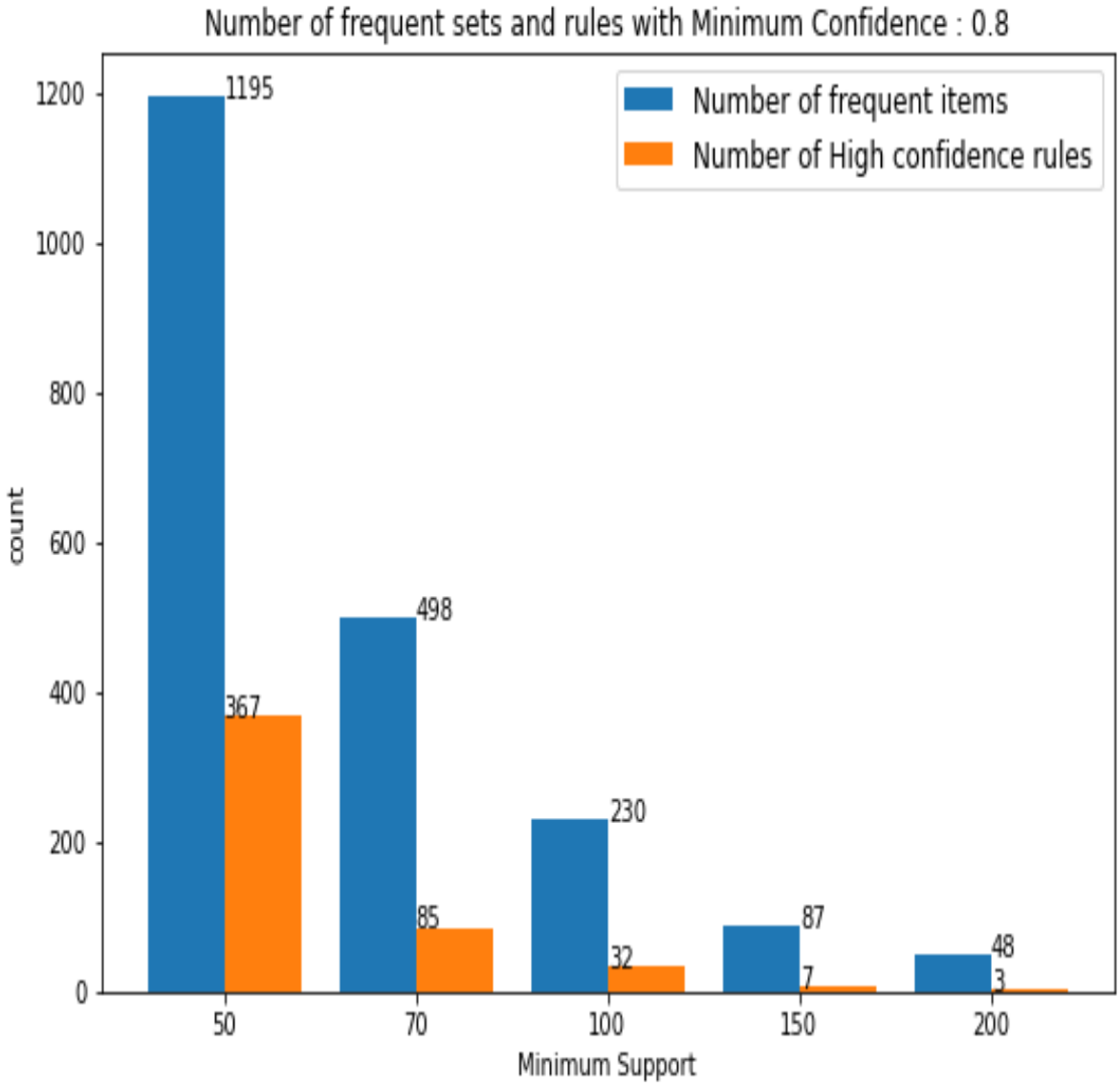
**Support Counting**: After Pruning we get a **List** of **sets** so we now calculated the support of each itemset from the **Data Frame** and store them as key-value pair in a **dictionary**.

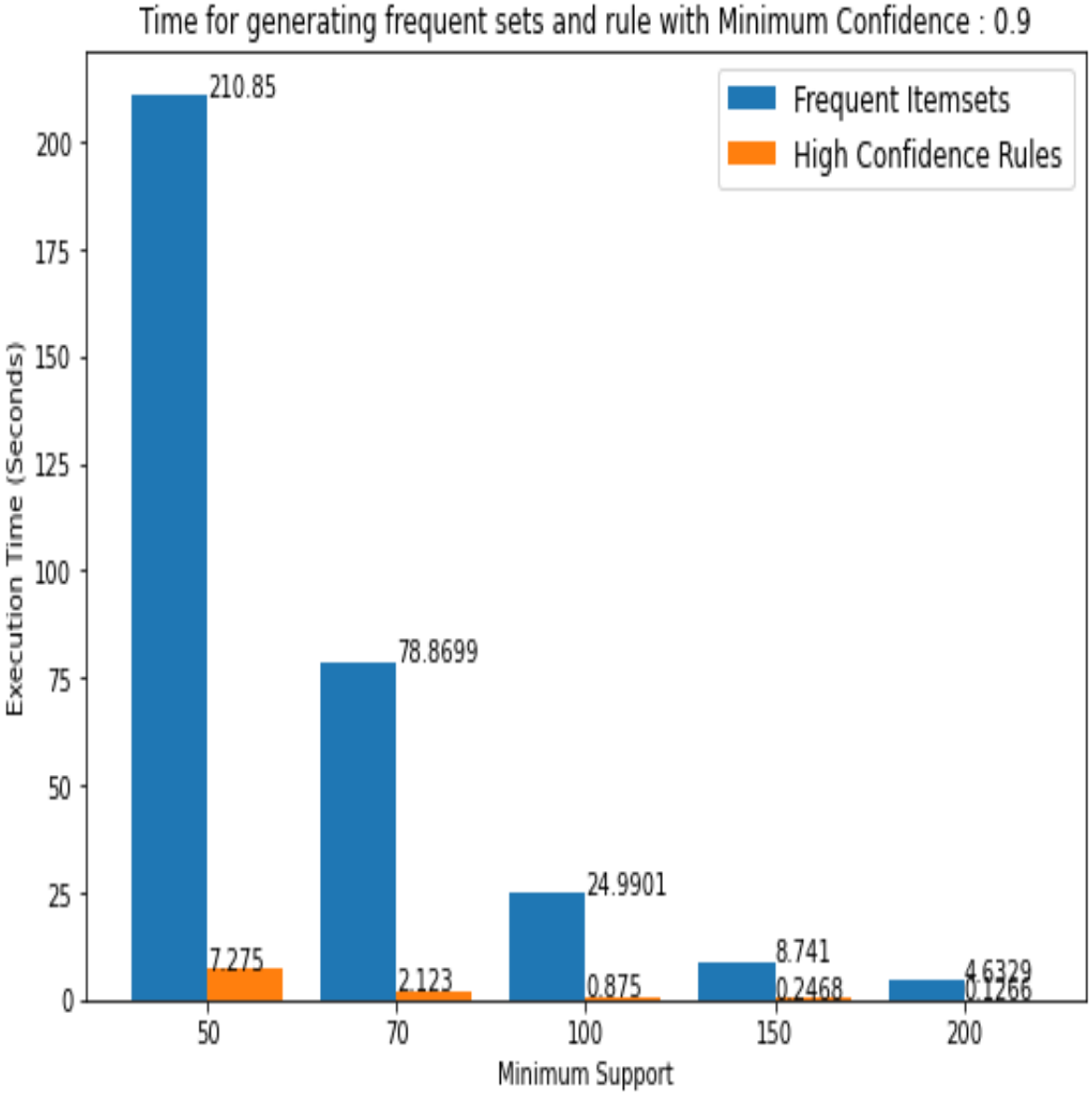
**Candidate Elimination**: Now we have keys(itemset) and values(support) in a **dictionary**. We check each key's support with minimum support if not satisfied then we eliminate them. Those which satisfy can be stored in a **List**. All such lists are stored in a **final List**.

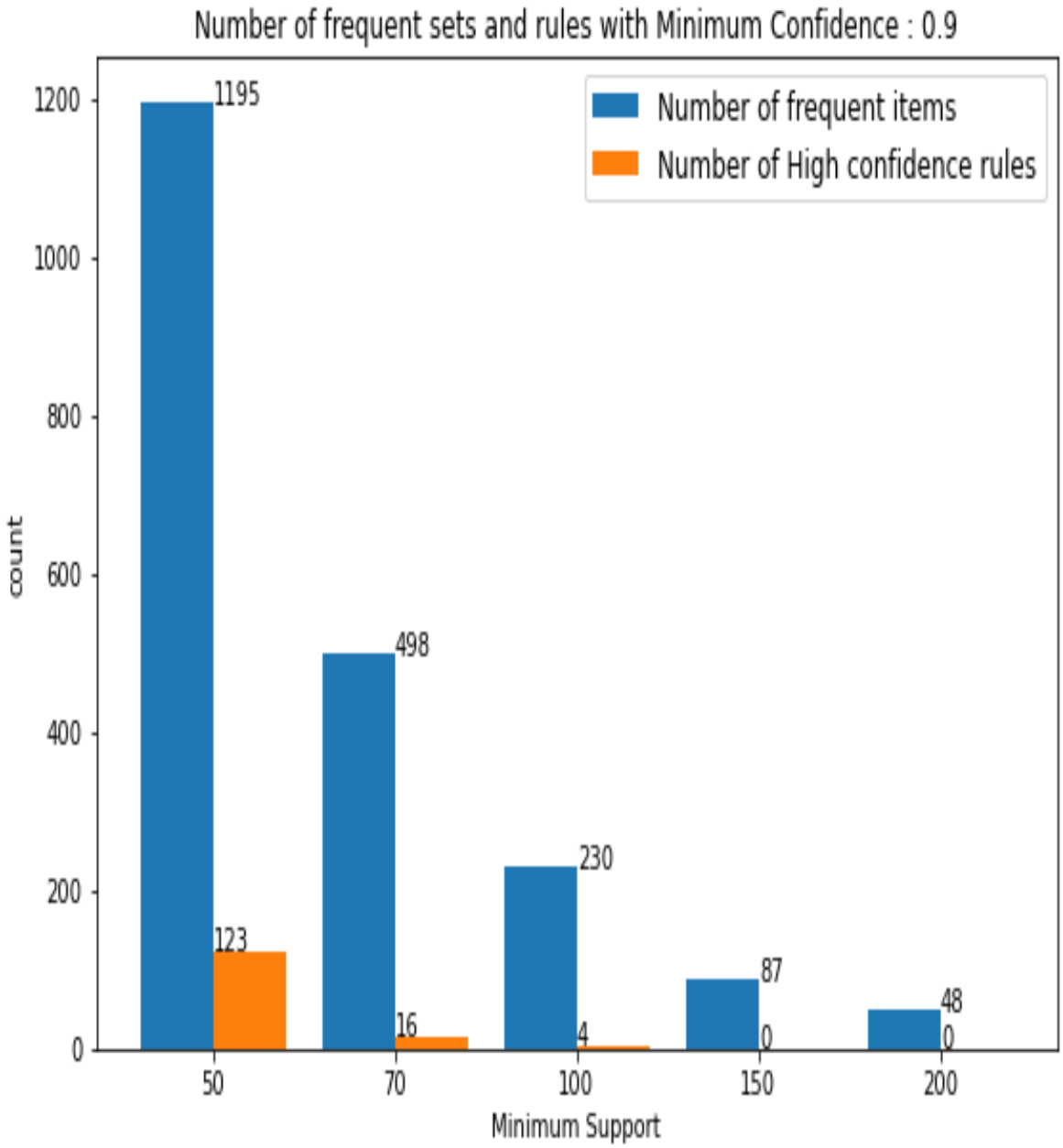
**Rule Generation**: Now we have the **final List** consisting of all the frequent itemset. Now we calculate each itemset **combinations List** from 2 to n-1. Calculate each combinations confidence using support from **Data Frame.** And check the minimum confidence if it satisfies then add to output.

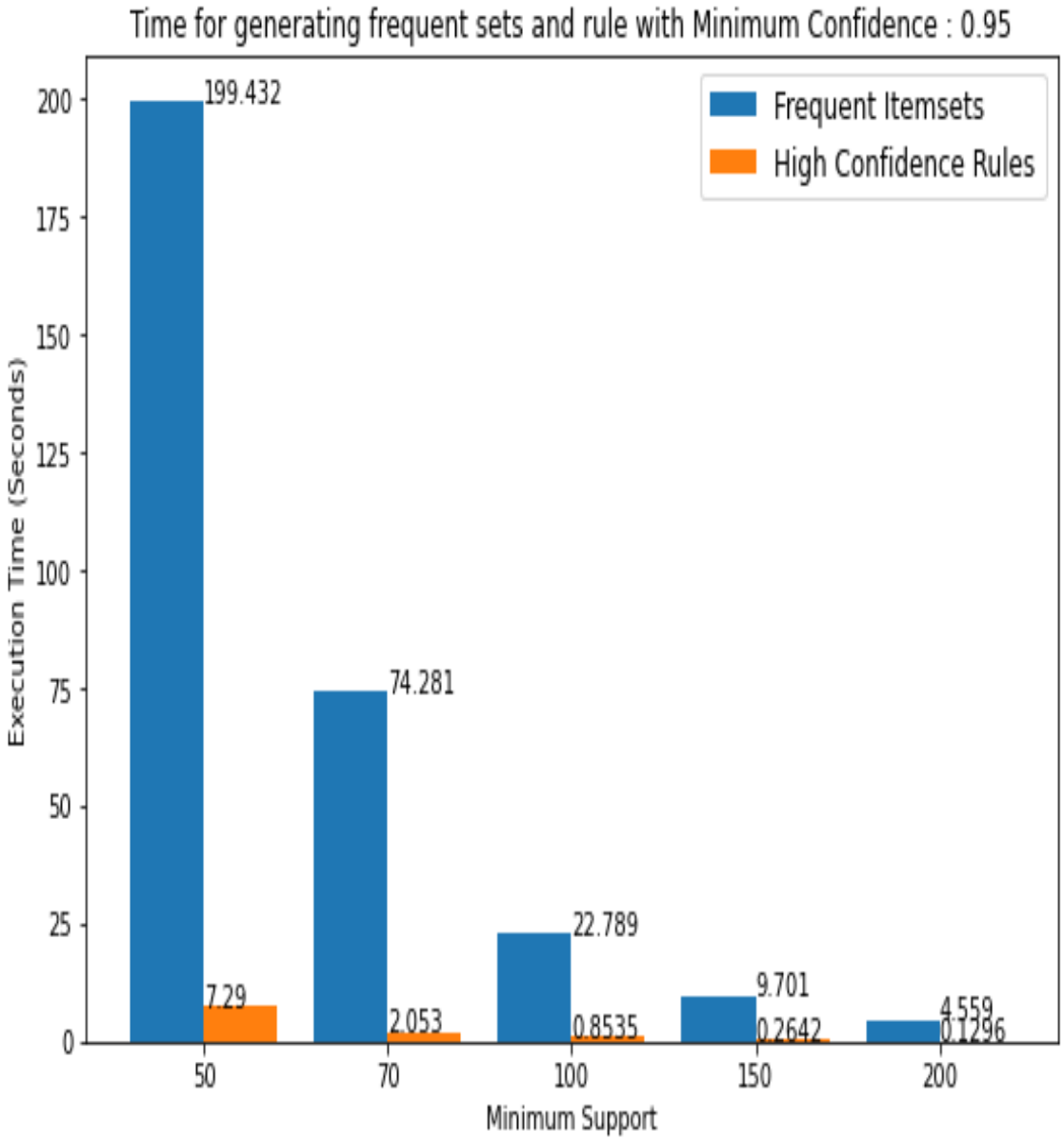
2)

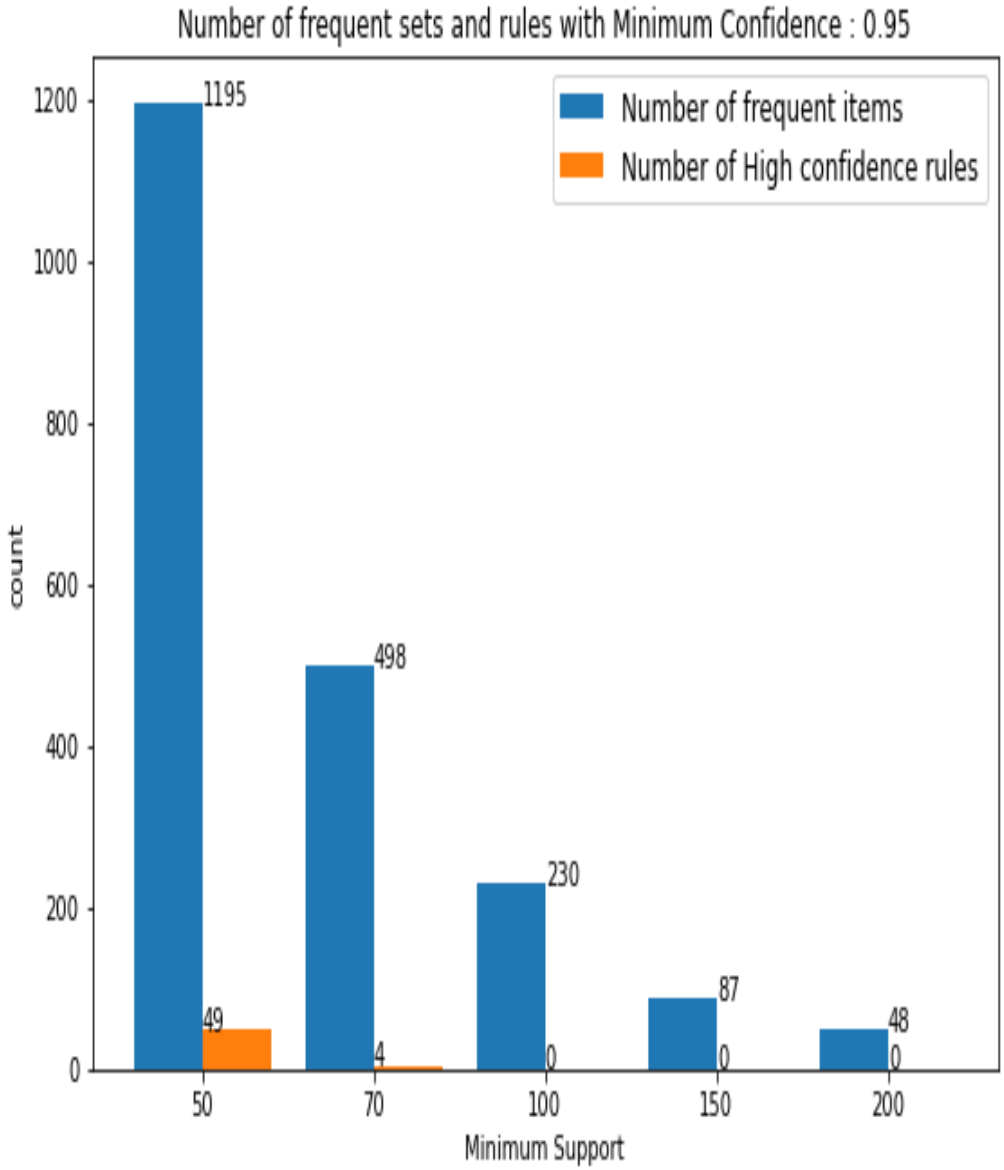

Time for generating frequent sets and rule with Minimum Confidence : 0.8
Frequent Itemsets
High Confidence Rules
Execution Time (Seconds)
Minimum Support
200.85
7.865
65.8699
2.73739
23.9901
1.034
8.75
0.2744
4.6129
0.1326
0
25
50
75
100
125
150
175
200
50
70
100
150
200


Number of frequent sets and rules with Minimum Confidence : 0.8
Number of frequent items
Number of High confidence rules
count
Minimum Support
1200
1000
800
600
400
200
0
50
70
100
150
200
1195
367
498
85
230
32
87
7
48
3


Time for generating frequent sets and rule with Minimum Confidence : 0.9
Frequent Itemsets
High Confidence Rules
Execution Time (Seconds)
Minimum Support
210.85
7.275
78.8699
2.123
24.9901
0.875
8.741
0.2468
4.6329
0.1266
0
25
50
75
100
125
150
175
200
50
70
100
150
200


Number of frequent sets and rules with Minimum Confidence : 0.9
Number of frequent items
Number of High confidence rules
count
Minimum Support
1200
1000
800
600
400
200
0
50
70
100
150
200
1195
123
498
16
230
4
87
0
48
0


Time for generating frequent sets and rule with Minimum Confidence : 0.95
Frequent Itemsets
High Confidence Rules
Execution Time (Seconds)
Minimum Support
199.432
7.29
74.281
2.053
22.789
0.8535
9.701
0.2642
4.559
0.1296
0
25
50
75
100
125
150
175
200
50
70
100
150
200


Number of frequent sets and rules with Minimum Confidence : 0.95
Number of frequent items
Number of High confidence rules
count
Minimum Support
1200
1000
800
600
400
200
0
50
70
100
150
200
1195
49
498
4
230
0
87
0
48
0