**Project Description:**

The Apriori algorithm is used in association rule mining, concentrating on the generation of the frequently used patterns, in the form of association rules.

The association rule mining is the process of identifying a pattern of the frequently occurring values in the given data set.

For example,

Vacation (X, “New Year”) ∧ Price (X, “50K..220K”) ⇒ Searches(X, “cruise-holiday”) [support=22%, conﬁdence=55%],

where X is a variable representing the internet search engine user. The rule indicates that of the internet search engine users under study, 22% of the users search for new year vacation between price 50k to 220k, also searches for cruise-holiday vacation online. There is a 55% probability that a user in this group searches for a cruise-holiday online.

In the above example, using association, the internet search engine company can use these mined strong rules to determine what the users search for and/or buy online along with other sites or products, viewed or bought respectively. With this information, the internet search engine company can then post relevant advertisements to that kind of customers who exhibit a high likelihood of visiting related sites or purchasing additional products.

**Description of the program:**

In the given project, the dataset given is based on a retail market basket data obtained from an anonymous Belgian retail store. The dataset contains information of 88162 transactions in total. Each transaction consists of a number of data items. Now, by using Apriori algorithm, one can determine a set of association rules by identifying the frequent patterns in these transactions. The user inputs a minimum support and minimum confidence. The minimum support is the number of occurrences of an item in consideration below which no transactions are considered for the association ruling. The minimum confidence is the indication of how often the pattern has found to be true, below which the transactions are not considered for the association rule mining.

The language used here to implement the Apriori algorithm is Python. The program initially accepts the command line input of the input file name, minimum support and minimum confidence given in the percentage form (eg. python sample.py .2 .2), by using the sys library functions. In order to handle the .dat file, the program reads the file into a list called all\_trans[] in the form of a frozenset by using the split() function, thus, loading the file into an appropriate data structure. The program, then, creates a list, itemsets[], with each item in the all\_trans[] along with its frequency or number of occurrence of each item in all\_trans[]. This list would use defaultdict(int) which is a dictionary data structure that is used to handle collections and can be retrieved by a key along with which the count of each of the items in this list is determined. This list hence has the initial set of Candidate List C.

The support of each item in the list is then calculated by dividing the frequency by the number of transactions in all\_trans[], by calling the support\_evaluation() function. The transactions that have a frequency count greater than or equal to the minimum support are filtered into another list. This list gives the frequent itemset list L. The program then creates the next candidate list by creating a union of the items in the frequent itemset by calling the list\_creation(), which in turn calls the citemset\_creation() function every time the list has a length > = 2. This function is called every time after the initial Candidate List creation. Each union set is considered an item in the new list. The list\_creation() calls the item\_occurence() in order to calculate the frequency of the item, which determines the occurrence of each of these items in all\_trans[] is calculated and save in a list. This gives the next candidate list. The list\_creation() function then uses an if-condition to re-loop the same process of creating the next candidate list and its respective frequent item set list again.

The program then calls the association\_rule() function which is used to calculate the confidence of an item in consideration by calling the confidence\_evaluation(). The initial items that are a result of union, are divided into subsets, so that the confidence of each subset can be calculated. The confidence of the item is given by dividing the support of that item by the support of the subset of that item.

Finally, the association rules for the subsets are then generated and displayed on screen.

**Sample Output:**

C:\Users\Randhina\AppData\Local\Programs\Python\Python35> python sample.py retail.dat 0.1 .2

…

#The candidate list from which the final frequent itemset is generated

\*\*\*\*\*\*\*Candidate List c:

[defaultdict(<class 'int'>, {frozenset({'39', '41'}): 11414, frozenset({'38', '32'}): 2833, frozenset({'48', '41'}): 9018, frozenset({'38', '39'}): 10345, frozenset({'38', '48'}): 7944, frozenset({'38', '41'}): 3897, frozenset({'32', '48'}): 8034, frozenset({'39', '32'}): 8455, frozenset({'32', '41'}): 3196, frozenset({'39', '48'}): 29142})]

#The final Frequent Itemset L genererated is:

\*\*\*\*\*\*\*\*Frequent itemset list L:

[defaultdict(<class 'int'>, {}), [frozenset({'39', '41'}), 11414], [frozenset({'48', '41'}), 9018], [frozenset({'38', '39'}), 10345], [frozenset({'39', '48'}), 29142]]

#The subsets generated and their respective frequencies:

[defaultdict(<class 'int'>, {}), defaultdict(<class 'int'>, {}), [frozenset({'32'}), 15167], [frozenset({'41'}), 14945], [frozenset({'48'}), 42135], [frozenset({'38'}), 15596], [frozenset({'39'}), 50675], defaultdict(<class 'int'>, {}), [frozenset({'41', '48'}), 9018], [frozenset({'39', '41'}), 11414], [frozenset({'39', '48'}), 29142], [frozenset({'38', '39'}), 10345]]

#Association rules for the above set of items:

\*\*\*\*\*\*\*Association Rules Determined:

Confidence: 0.6034125125460019, rule: frozenset({'41'}) ->frozenset({'48'})

Confidence: 0.2140263438946244, rule: frozenset({'48'}) ->frozenset({'41'})

Confidence: 0.22523926985693143, rule: frozenset({'39'}) ->frozenset({'41'})

Confidence: 0.7637336901973905, rule: frozenset({'41'}) ->frozenset({'39'})

Confidence: 0.5750764676862358, rule: frozenset({'39'}) ->frozenset({'48'})

Confidence: 0.6916340334638661, rule: frozenset({'48'}) ->frozenset({'39'})

Confidence: 0.663311105411644, rule: frozenset({'38'}) ->frozenset({'39'})

Confidence: 0.20414405525407006, rule: frozenset({'39'}) ->frozenset({'38'})