APriori algorithm

The Apriori algorithm, proposed by Aggarwal and Srikant in 1994 in Fast Algorithms for Mining Association Rules is used for finding frequent item sets over a given dataset. The name apriori comes from the fact that the algorithm uses prior knowledge of frequent item set. The Apriori Algorithm initially proposed involved an iterative approach in which k itemsets are use to explore k+1 itemsets.

The apriori algorithm has an interesting pruning property. In order for an item set to be frequent, all non empty subsets of frequent item sets must be frequent. The dataset D is first scanned and the items which have minimum Support S are added to the frequent item set L1. The L1 frequent item set is then used to generate C2 candidate pairs by $L1 \bowtie L1$. The C2 candidate pairs consists of 2 items set pairs. All of the pairs in C2 candidate set are frequent because all the subsets are frequent. The database D is scanned and the transactions where those 2 items occur are counted. The set of L2 is then determined if the minimum support is met. Similarly join between $L2 \bowtie L2$ is performed to generate C3 and so on.

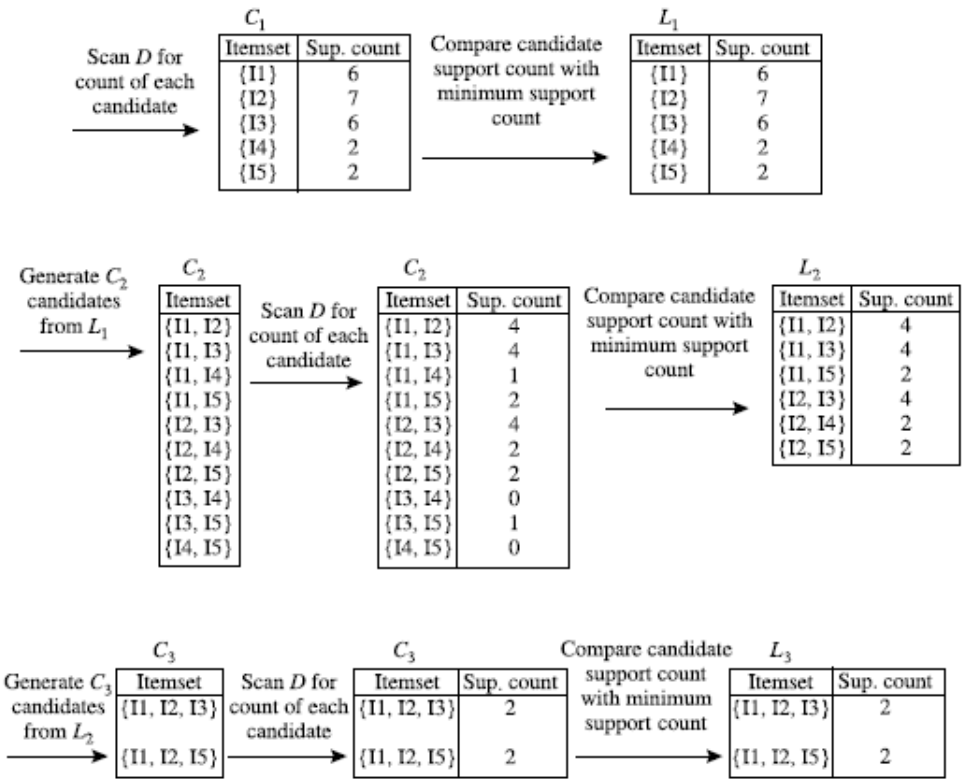
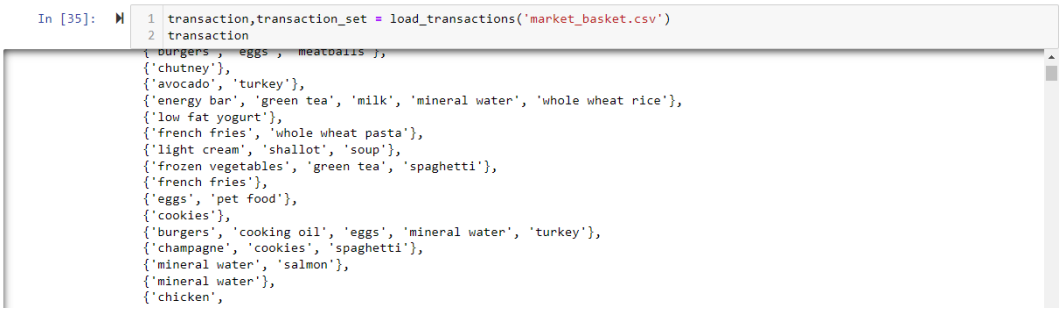


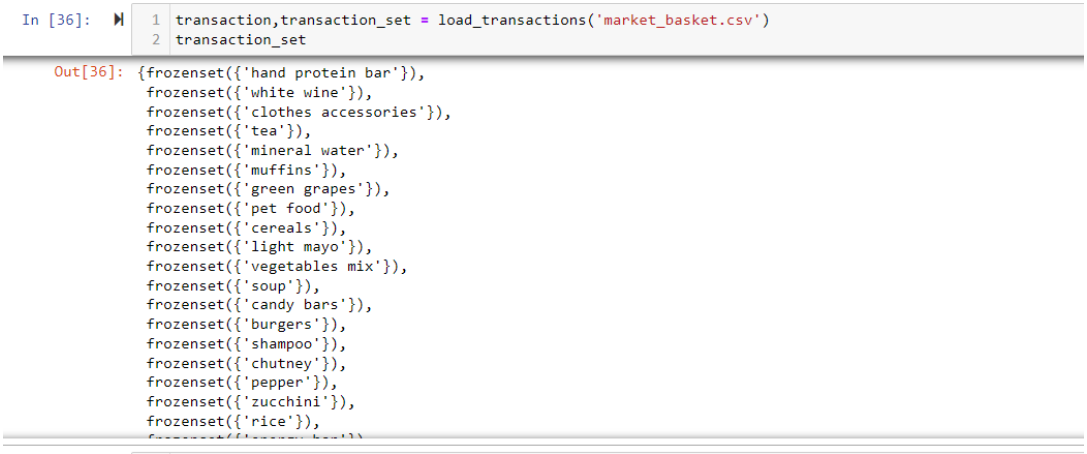
Fig 1.1: A process for Apriori Algorithm [1]

Design Choices:

1. The dataset used in this project is 'Market basket.csv'. The programming language used is python. Please use a Jupyter notebook to run the script.

2. The dataset D is read in using the load_transactions function. Each line of the file is considered as a transaction or a basket which contains various items bought by the consumer itself. The basket may contain multiple instances of the same object which will affect our final result and give us results based on frequency of item instead of how frequently the item is bought. To prevent this set is used. The items in each basket are stored as a set which are then added to the list. The list contains all the transactions of the database. To keep track of all unique items in the mentioned file frozen set is used. Frozen can be hashed, hence making it an ideal choice over sets in this situation. The method will return two data structures transaction (list) and transaction_set(set)





3. To generate first set of candidate pairs generateSetC1 function is used. To accumulate the counts of each item in transaction (list) generated previously, a dictionary is implemented. To make it visually easier to interpret, the dictionary is converted into a data frame. Each key value of the dictionary is a frozenset , not a string. The transaction_set returned by load_transactions is passed into generateSetC1 where frozen set is checked against the transaction list (also returned by load_transactions). Based on the membership relation of the set item , the count of the dictionary is updated

```
diction,it_set = generateSetC1(transaction,transaction_set)
df = convertToDataFrame(diction)
df
```

]:

	Item Set	Support Count
0	(hand protein bar)	39
1	(white wine)	124
2	(clothes accessories)	63
3	(tea)	29
4	(mineral water)	1788
...	...	...
115	(shrimp)	536
116	(champagne)	351
117	(cake)	608
118	(mint green tea)	42
119	(eggs)	1348

120 rows × 2 columns

4. Similarly for L1, Ck_generation and Lk_generation_with_minimum_Support dicitionary and frozen sets has been used.

5. For user inputs please refer the following template. A Confidence level of atleast 10 % should be selected or else too many rules will get generated.

```
-----------------Please enter filename, Minimum Support and Confidence--------------------

Please enter minimum support and confidence level as integer number between 0-100.

File Name :
market_basket.csv
 Enter Minimum Support. Recommendation : As golden rule set support high such as 1% of the total items----

 Minimum Support :
1
 Confidence :
4
C1 Item Sets
```

Problem Encountered:

1. The first problem encountered was to find a dataset such that it has enough data for analysis and also such that the data provided is not too large as the algorithm was being implemented with an iterative approach. Using a huge dataset would have increased the number of pairs to compare and increase computational time.
2. Secondly, as it was my first time dealing with frozen set knowledge of basic implementation was required. Some properties applicable to set were not applicable to frozen set as they are immutable.
3. Thirdly, generating rules was a little tricky. Counts of all the frequent item set (1 frequent, 2 frequent, 3 frequent , 4 frequent ,……, n frequent) had to be stored so that confidence rule can be generated in the end.

Conclusion:

The APriori Algorithm using Iterative Approach was extremely fun to implement. The result obtained were matched with the results produced by apriori model algorithms found apyori package provided by Python Package Index. However minor differences in some rules could be observed, which were discarded as they were not significant.

The Apriori Algorithm implemented here however is very naïve approach. Various method such as triangular matrix can be used to increase the efficiency of the algorithm. Other techniques such as FP growth (Frequent pattern growth referred in *[1]*) can be used which eliminates the costly generation of candidate pairs.

References

[1] J. Han, M. Kamber, and J. Pei, “Mining Frequent Patterns, Associations, and Correlations: Basic Concepts and Methods,” in *Data Mining Concepts and Technique*, Third., Waltham, MA, USA: Morgan Kaufmann Publishers, 2012, pp. 250–252.

[2] R. Agrawal and R. Srikant, “Fast algorithms for Mining Association rules.” [Online]. Available: https://rakesh.agrawal-family.com/papers/vldb94apriori_rj.pdf. [Accessed: 11-Dec-2021].