Objective:

- To research Apriori Algorithm, its application, and implementations.
- To enhance upon the serialized Apriori Algorithms and research ideas which includes parallelization and gear like SparkRDD, Map-Reduce, etc.
- The serial Apriori algorithm is both data-intensive and computing-intensive. This makes the serial Apriori algorithm very time-consuming while running on big data. Our objective is to resolve this hassle by using parallelization. We can parallelize it with the assistance of Map-reduce, SparkRDD, etc.

Work done:

- In this project, we have focused and worked on the parallelization of the Apriori Algorithm using PySpark and concepts of Partitioning.
- So the task was to now parallelize it and hence it can provide us efficient execution time for large datasets.
- Before implementing the algorithm, we created a function which helps us to generate a new candidate itemset ($\mathbf{c}_k$) using previously generated frequent itemsets ($\mathbf{f}_{k-1}$). Also for doing this we have used a set data structure.
- Using the candidate itemsets generated above, we will generate the frequent itemsets by counting how many times the items come together in the transactions.
- Using the functions and the technique of parallelization we generated all the frequent itemsets possible by using PySpark in a parallelized algorithm.
- We also tried comparing the above-parallelized algorithm(implemented using PySpark) with the serial apriori algorithm.

Evaluation:

- We executed the above method in both parallelized and linear apriori algorithms. And we observed that parallel is sufficiently faster compared to the linear algorithm.
- We used chess and pumsb datasets for comparison between both algorithms.
- The chess dataset was used to compare the two algorithms on datasets of specific sizes. A change was made to truncate rows (transactions) and columns (unique items) to get different-sized datasets.
- We also used graphs to more systematically and logically compare both algorithms on a modified chess dataset, and concluded our results.

Final Outcomes:

- And finally using the execution time and graphs plotted from the modified chess dataset we can conclude that the parallelized apriori algorithm is quite faster compared to the linear apriori algorithm.

[GitHub Link]