**Goal**:

The goal of the project is to implement and analyze runtimes of popular frequent pattern mining algorithms. Classical Apriori, FP-growth, and improved Apriori were used to perform frequent pattern mining on UCI’s Adult dataset, available at <http://archive.ics.uci.edu/ml/machine-learning-databases/adult/>.

**Data preprocessing**:

The main class reads the relative file path for adult.data. Names for comma-separated attributes and numeric concept hierarchies were manually entered.

The DataScanner class takes in the relative file path for adult.data and the comma-separated attribute names. A 2D array was created to store the attribute names in the first row and the data points in the following rows.

The MissingValueHandler class takes in the data frame. Instances of missing values for for attributes are counted. Because in adult.data, all missing values are for nonnumeric attributes and are small in counts, we simply delete all rows with missing values.

The HierarchySort class takes in the data frame and the numeric concept hierarchy names and discretized numeric nonbinary attributes as “low”, “medium”, or “high”.

The frequent pattern mining algorithm takes in the data frame and the support count value and finds frequent patterns.

**Analysis**:

I originally implemented all three algorithms in one run. However, the garbage collection generated by each algorithm seemed to impact the runtime for the following algorithm. I then implemented all three algorithms individually.

Runtimes were recorded six times for all algorithms for accuracy. The average runtimes and the natural logs of the average runtimes are located below.

|  |  |  |  |
| --- | --- | --- | --- |
| Support | Apriori Runtime (in ns) | FP-Growth Runtime (in ns) | Improved Apriori Runtime (in ns) |
| 0.15 | 3.94233E+11 | 69329210660 | 3.90845E+11 |
| 0.2 | 1.5791E+11 | 44644648226 | 1.54258E+11 |
| 0.25 | 81624877393 | 32840825456 | 78399164130 |
| 0.35 | 32569431278 | 16526595672 | 29309070326 |
| 0.45 | 14637015026 | 8301331909 | 12881264912 |
| 0.55 | 9779264941 | 6759628153 | 8194790079 |
| 0.65 | 5447379244 | 4935631035 | 4645994304 |
| 0.75 | 4407473858 | 3852866603 | 3789249584 |
| 0.85 | 3685973830 | 1689335326 | 3041088067 |
| 0.95 | 2662469406 | 976664846.8 | 2224388452 |

The improvement of improved Apriori pertains to transaction reduction. For improved Apriori, every transaction line is parsed once into a 2D array before scanning, which is different from classical Apriori, for which one individual transaction line is parsed after scanning and subjected to candidate generation. Since information about every transaction line is accessible before scanning, we can mark which transaction lines lack infrequent itemsets and therefore will undergo candidate generation and which transaction lines contain infrequent itemsets will skip candidate generation. Because candidate generation is expensive, improved Apriori is faster than classical Apriori.

The magnitude of improvement for improved Apriori decreases as the support decreases. This behavior can be attributed to the fact that less support results in less infrequent itemsets to be marked for skipping candidate generation. There are some deviations in the overall trend, noticeably at supports 0.55 and 0.85, which could either be due to the low number of trials resulting in high variance in runtimes or be due to the discretization of continuous data points biasing toward certain frequent patterns (such that less infrequent itemsets are marked for skipping candidate generation) at different supports.

The runtimes for FP-growth were the lowest at most support levels since the data frame is scanned only once. At supports 0.65 and 0.75, FP-growth is beaten by improved Apriori although the runtime difference is not statistically significant. This observation can be best explained by the low number of trials resulting in high variance in runtimes.

All algorithms generate the same frequent patterns although the output from FP-growth is sequenced differently compared to the outputs from classical Apriori and improved Apriori.