FP-Growth Association Rule Tutorial

September 21, 2025

1 Introduction

The FP-Growth algorithm mines frequent itemsets without candidate generation by compressing transactions into a frequent-pattern tree (FP-tree). After a single database scan builds the FP-tree, recursive conditional mining discovers patterns efficiently, making FP-Growth a popular alternative to Apriori for large and dense datasets.

2 Theory and Formulas

2.1 FP-tree Construction

Given transactions $\mathcal{D}$ and a minimum support threshold `min_supp`, FP-Growth counts item frequencies, removes infrequent items, and orders each transaction by descending frequency before insertion. Each transaction is mapped into the FP-tree, merging common prefixes and incrementing node counts. A header table maintains links to nodes containing the same item to support traversal.

2.2 Conditional Pattern Bases

For each item i in the header table, FP-Growth gathers all prefix paths ending at i to form the conditional pattern base. These weighted paths define a conditional FP-tree from which frequent patterns that include i are mined recursively. The support of an itemset is the sum of counts along paths leading to the item. Because the search explores only frequent prefixes, the algorithm avoids combinatorial explosion.

2.3 Rule Generation

Once frequent itemsets are known, association rules $X \Rightarrow Y$ are derived with the same support, confidence, and lift metrics as Apriori:

$$\mathrm{supp}(X) = \frac{|\{T \mid X \subseteq T,\ T \in \mathcal{D}\}|}{|\mathcal{D}|}, \tag{1}$$

$$\mathrm{conf}(X \Rightarrow Y) = \frac{\mathrm{supp}(X \cup Y)}{\mathrm{supp}(X)}, \tag{2}$$

$$\mathrm{lift}(X \Rightarrow Y) = \frac{\mathrm{conf}(X \Rightarrow Y)}{\mathrm{supp}(Y)}. \tag{3}$$

Additional measures such as conviction or leverage help prioritize the most actionable rules.

3 Applications and Tips

- **Retail analytics**: identify product bundles and plan promotions in large point-of-sale datasets.
- **Web usage mining**: uncover navigation patterns or co-accessed resources from clickstreams.
- **Industrial monitoring**: detect co-occurring alarms or events in sensor logs.
- **Best practices**: tune `min_supp` to limit tree size, normalize item ordering for deterministic results, limit recursion depth for very dense data, and validate rules against business constraints.

4 Python Practice

The script `gen_fp_growth_figures.py` synthesizes transactional data, implements a compact FP-Growth miner, and visualizes the support–confidence distribution and lift histogram for the resulting rules.

Listing 1: Excerpt from gen$_{f}p_{g}rowth_{f}igures.py$

```python
frequent_itemsets = fpgrowth(transactions, min_support=0.06)
rules = derive_rules(frequent_itemsets, min_confidence=0.5)

for lhs, rhs, support, confidence, lift in rules:
    # store metrics for plotting
    support_vals.append(support)
    confidence_vals.append(confidence)
    lift_vals.append(lift)
```

5 Result

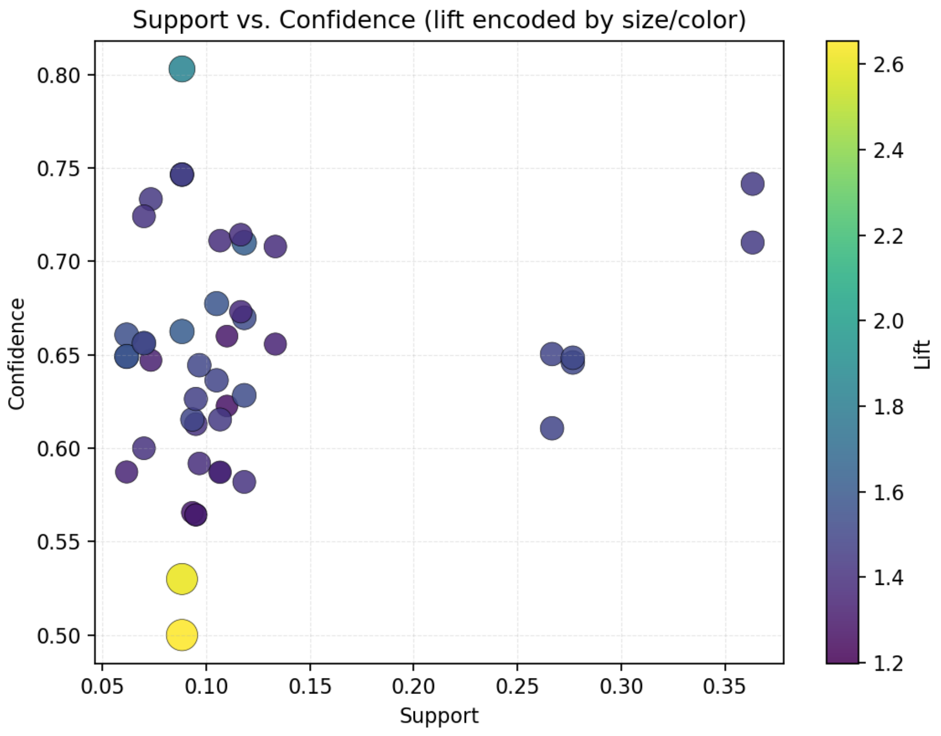

Figure 1: Support vs. confidence for FP-Growth rules (marker size indicates lift)

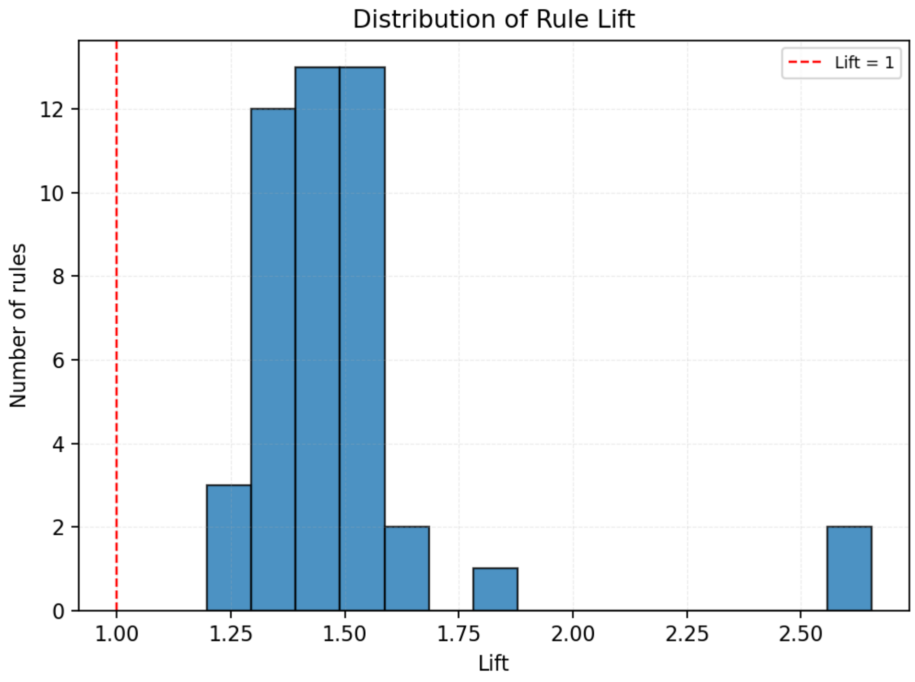

Figure 2: Lift distribution highlighting strong associations

6 Summary

FP-Growth compresses transactions into an FP-tree and mines frequent patterns without exhaustive candidate generation. By adjusting support thresholds, item ordering, and rule metrics, analysts can extract interpretable association rules even from large datasets. The synthetic example demonstrates how visualization aids in evaluating mined patterns and tuning parameters.