# Analisis Komparatif Algoritma Apriori dan FP-Growth: Implementasi Manual vs Pustaka

**Abstrak**

Laporan ini menyajikan analisis komparatif komprehensif dua algoritma fundamental dalam penambangan *itemset* sering dan pembelajaran aturan asosiasi: Apriori dan FP-Growth. Studi ini menyelidiki prinsip-prinsip teoretis, mekanisme operasional, dan karakteristik kinerja praktisnya di seluruh implementasi manual dan pendekatan berbasis pustaka yang dioptimalkan. Dengan memeriksa metrik utama seperti efisiensi komputasi (waktu eksekusi) dan pemanfaatan memori, penelitian ini bertujuan untuk menjelaskan pertukaran yang melekat pada setiap algoritma dan strategi implementasi. Temuan ini memberikan wawasan penting bagi peneliti dan praktisi dalam memilih algoritma dan pendekatan pengembangan yang paling sesuai untuk berbagai tugas penambangan data, terutama yang berkaitan dengan skalabilitas dan manajemen sumber daya dalam *dataset* besar.

## 1. Pendahuluan

### 1.1. Tinjauan Umum Penambangan *Itemset* Sering dan Pembelajaran Aturan Asosiasi

Penambangan *itemset* sering adalah tugas inti dalam penambangan data yang bertujuan untuk menemukan pola *item* yang sering muncul bersama dalam suatu *dataset* (Tan et al., 2019). Teknik ini menjadi dasar untuk memahami hubungan dalam data transaksional, seperti kebiasaan pembelian pelanggan dalam analisis keranjang belanja (*market basket analysis*). Misalnya, dalam analisis keranjang belanja, penambangan *itemset* sering dapat mengungkapkan bahwa pelanggan yang membeli roti dan mentega juga sering membeli susu.

Pembelajaran aturan asosiasi memperluas penambangan *itemset* sering dengan menurunkan pernyataan "jika-maka" yang menyoroti tren dan hubungan umum (Han et al., 2022). Aturan-aturan ini sangat penting untuk pengambilan keputusan di berbagai domain, termasuk ritel, analitik web, deteksi penipuan, dan bioinformatika. Dalam konteks ini, penambangan *itemset* sering berfungsi sebagai prasyarat penting untuk menghasilkan aturan asosiasi yang bermakna.

### 1.2. Pengantar Algoritma Apriori dan FP-Growth

Algoritma Apriori, yang diajukan oleh Agrawal dan Srikant (1994), adalah algoritma perintis dan banyak digunakan untuk penambangan *itemset* sering. Algoritma ini menggunakan pendekatan "bawah-atas" (*bottom-up*) dan sifat anti-monoton dari *support* untuk secara efisien menemukan *itemset* sering.

Algoritma FP-Growth (*Frequent Pattern Growth*), yang diperkenalkan oleh Han, Pei, dan Yin (2000), muncul sebagai alternatif Apriori, yang secara khusus dirancang untuk mengatasi ketidakefisienannya. Algoritma ini mencapai tujuannya dengan menggunakan struktur data yang ringkas yang disebut FP-tree dan menghindari generasi kandidat yang mahal secara komputasi. Perkembangan ini mencerminkan evolusi alami dalam desain algoritmik, di mana Apriori meletakkan dasar teoretis, dan FP-Growth membangun di atas fondasi tersebut dengan memperkenalkan struktur data dan strategi penambangan yang lebih efisien.

### 1.3. Rasionalitas Perbandingan Implementasi Manual dan Pustaka

Perbandingan antara implementasi manual dan pustaka menyoroti titik keputusan penting bagi praktisi dan peneliti: apakah akan mengimplementasikan algoritma dari awal untuk pemahaman yang lebih mendalam atau memanfaatkan pustaka yang ada yang dioptimalkan untuk efisiensi dan keandalan.

* **Implementasi Manual:** Menawarkan pemahaman mendalam tentang mekanisme internal algoritma, aliran data, dan langkah-langkah komputasi. Proses ini sangat berharga untuk tujuan pendidikan dan bagi peneliti yang ingin memodifikasi atau memperluas algoritma inti.
* **Implementasi Pustaka:** Pustaka seperti mlxtend untuk Python (Raschka, 2018) biasanya sangat dioptimalkan, kuat, dan dirancang untuk aplikasi praktis pada *dataset* besar. Pustaka ini mengabstraksi sebagian besar kompleksitas, memungkinkan pengguna untuk fokus pada persiapan dan interpretasi data.

Perbandingan ini menggarisbawahi dilema antara pembelajaran dan penerapan: implementasi manual lebih berharga untuk tujuan pembelajaran, sementara pustaka hampir selalu lebih unggul untuk lingkungan produksi karena kekokohan dan optimasinya.

## 2. Latar Belakang Teoretis

### 2.1. Algoritma Apriori

* **Prinsip Inti:** Didasarkan pada "sifat Apriori" atau "sifat anti-monoton dari *support*": jika sebuah *itemset* sering, maka semua *subset*-nya juga harus sering. Sebaliknya, jika sebuah *itemset* tidak sering, semua *superset*-nya juga tidak sering. Sifat ini sangat penting untuk memangkas ruang pencarian secara efisien (Agrawal & Srikant, 1994).
* **Proses:** Bekerja dengan pendekatan "bawah-atas" (*bottom-up*). Dimulai dengan menghasilkan dan menguji *itemset* tunggal, kemudian secara iteratif menggabungkan *itemset* yang sering untuk menghasilkan kandidat *itemset* yang lebih besar. Setiap iterasi memerlukan pemindaian penuh pada *database* untuk menghitung *support* dari kandidat.
* **Kelebihan:** Secara konseptual lugas, mudah dipahami, dan mudah diimplementasikan.
* **Kekurangan:** Menghasilkan jumlah *candidate itemset* yang sangat besar dan memerlukan banyak pemindaian *database*. Hal ini menyebabkan biaya komputasi dan penggunaan memori yang tinggi, sehingga tidak efisien untuk *dataset* besar (Han et al., 2022).
* **Kompleksitas:** Kompleksitas waktu dan ruang dalam kasus terburuk bersifat eksponensial terhadap jumlah *item* unik dalam *dataset*, .

### 2.2. Algoritma FP-Growth

* **Prinsip Inti:** Mengatasi kelemahan Apriori dengan menghindari generasi kandidat secara eksplisit. FP-Growth menggunakan struktur data pohon prefiks yang ringkas, yaitu FP-tree, untuk menyimpan informasi frekuensi *itemset* (Han et al., 2000).
* **Proses:** Bekerja dalam dua fase utama. Pertama, *database* dipindai dua kali: sekali untuk menemukan *item* sering dan mengurutkannya, dan kedua untuk membangun FP-tree. Kedua, algoritma menambang *itemset* sering secara rekursif dari FP-tree menggunakan pendekatan "bagi dan taklukkan" (*divide and conquer*).
* **Kelebihan:** Secara signifikan lebih cepat dan lebih efisien dalam penggunaan memori dibandingkan Apriori untuk *dataset* besar, karena hanya memerlukan dua pemindaian *database* dan tidak ada generasi kandidat (Han et al., 2022).
* **Kekurangan:** Algoritma dan struktur datanya lebih kompleks untuk diimplementasikan dari awal. Dalam kasus *dataset* yang sangat padat atau memiliki banyak pola panjang, FP-tree bisa menjadi besar dan rumit, yang berpotensi mengurangi keunggulan kinerjanya.
* **Kompleksitas:** Meskipun tidak sepenuhnya linier, kompleksitas waktunya secara umum jauh lebih baik daripada Apriori dan lebih dekat dengan proporsionalitas terhadap ukuran *database* dan jumlah *itemset* sering yang ditemukan (Tan et al., 2019).

### 2.3. Metrik Evaluasi Utama

Aturan asosiasi dievaluasi menggunakan tiga metrik utama (Tan et al., 2019):

* ***Support***: Mengukur frekuensi kemunculan sebuah *itemset* dalam *dataset*.
  + Formula:
* ***Confidence***: Mengukur seberapa sering aturan tersebut terbukti benar. Ini adalah probabilitas kondisional dari konsekuen, mengingat anteseden.
  + Formula:
* ***Lift***: Mengukur seberapa besar kemungkinan konsekuen terjadi ketika anteseden terjadi, dibandingkan dengan probabilitas terjadinya konsekuen secara acak.
  + Formula:
  + Interpretasi: *Lift* > 1 menunjukkan korelasi positif, *Lift* = 1 menunjukkan independensi, dan *Lift* < 1 menunjukkan korelasi negatif. *Lift* sangat penting untuk menemukan aturan yang benar-benar menarik dan bukan hanya hasil dari tingginya frekuensi masing-masing *item*.

## 3. Pengaturan Eksperimen

### 3.1. Deskripsi *Dataset*

Eksperimen dilakukan pada *dataset* transaksional, yang merupakan ciri khas dari analisis keranjang belanja. Kinerja algoritma sangat bergantung pada karakteristik data seperti jumlah transaksi (ukuran), jumlah *item* unik (lebar), dan panjang transaksi rata-rata (kepadatan). Oleh karena itu, perbandingan yang kuat harus menguji algoritma di berbagai jenis *dataset* untuk menilai skalabilitas dan kekokohan (Han et al., 2022).

### 3.2. Pendekatan Implementasi

* **Implementasi Manual:** Dikembangkan dari awal menggunakan Python untuk secara eksplisit mereplikasi logika langkah demi langkah dari algoritma Apriori dan FP-Growth. Pendekatan ini memberikan transparansi penuh terhadap proses komputasi.
* **Implementasi Pustaka:** Memanfaatkan pustaka mlxtend di Python, yang menyediakan fungsi apriori dan fpgrowth yang sangat dioptimalkan. Pustaka ini diakui karena efisiensinya dan didokumentasikan dalam literatur akademis (Raschka, 2018).

### 3.3. Metodologi Pengukuran Kinerja

Kinerja diukur berdasarkan:

* **Waktu Eksekusi:** Diukur menggunakan modul time atau timeit Python untuk fase penambangan *itemset* sering.
* **Pemanfaatan Memori:** Diukur pada puncak eksekusi menggunakan pustaka seperti memory\_profiler. Eksperimen dijalankan di lingkungan yang terkontrol dan diulang beberapa kali dengan nilai ambang min\_support yang bervariasi untuk memastikan hasil yang valid dan dapat direproduksi.

## 4. Hasil dan Diskusi

### 4.1. Perbandingan Kinerja

* **Apriori (Manual vs. Pustaka):** Implementasi mlxtend.apriori secara konsisten jauh mengungguli versi manual. Kesenjangan kinerja ini disebabkan oleh optimasi tingkat rendah dalam pustaka yang sulit ditiru dalam implementasi Python murni (Raschka, 2018). Namun, kedua implementasi pada dasarnya dibatasi oleh kebutuhan untuk menghasilkan dan menyimpan *candidate set* yang besar.
* **FP-Growth (Manual vs. Pustaka):** Serupa dengan Apriori, mlxtend.fpgrowth secara signifikan lebih cepat daripada implementasi manual. Efisiensi ini berasal dari implementasi FP-tree dan proses penambangan rekursif yang sangat dioptimalkan dalam pustaka.
* **Apriori vs. FP-Growth (Lintas Implementasi):** Secara keseluruhan, implementasi pustaka FP-Growth adalah yang paling efisien dalam hal waktu eksekusi dan umumnya lebih hemat memori. Hal ini menegaskan keunggulan desain fundamental FP-Growth dalam menghindari *bottleneck* generasi kandidat Apriori (Han et al., 2000).

### 4.2. Implikasi Pemilihan

* **Untuk Tujuan Pembelajaran:** Implementasi manual, meskipun lambat, sangat berharga untuk membangun pemahaman yang mendalam tentang cara kerja internal algoritma.
* **Untuk Aplikasi Produksi:** Implementasi berbasis pustaka adalah pilihan yang jelas. Mereka menawarkan kecepatan, keandalan, dan skalabilitas yang diperlukan untuk menangani *dataset* dunia nyata yang besar dan kompleks.
* **Berdasarkan Karakteristik Data:** FP-Growth umumnya lebih unggul untuk *dataset* besar dan padat (Han et al., 2022). Namun, Apriori mungkin masih cukup memadai untuk *dataset* yang lebih kecil atau jarang di mana kesederhanaan implementasi menjadi prioritas.

## 5. Kesimpulan

Laporan ini mengonfirmasi bahwa algoritma FP-Growth secara teoretis dan praktis lebih efisien daripada Apriori, terutama untuk *dataset* skala besar. Keunggulan ini diperkuat secara signifikan saat menggunakan implementasi pustaka yang dioptimalkan seperti mlxtend, yang secara konsisten mengungguli implementasi manual. Pilihan antara algoritma dan pendekatan implementasi harus selaras dengan tujuan proyek. Untuk pemahaman konseptual dan tujuan akademis, membangun algoritma secara manual memberikan wawasan yang tak ternilai. Namun, untuk aplikasi praktis yang menuntut kinerja dan skalabilitas tinggi, memanfaatkan pustaka FP-Growth yang dioptimalkan adalah pendekatan yang paling logis dan efektif.

**Daftar Pustaka**

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. In J. B. Bocca, M. Jarke, & C. Zaniolo (Eds.), *Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94* (pp. 487–499). Morgan Kaufmann Publishers Inc.

Han, J., Kamber, M., & Pei, J. (2022). *Data mining: Concepts and techniques* (4th ed.). Morgan Kaufmann Publishers Inc.

Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (pp. 1–12). Association for Computing Machinery. <https://doi.org/10.1145/342009.335372>

Raschka, S. (2018). MLxtend: Providing machine learning and data science utilities and extensions to Python’s scientific computing stack. *Journal of Open Source Software, 3*(24), 638. <https://doi.org/10.21105/joss.00638>

Tan, P.-N., Steinbach, M., & Kumar, V. (2019). *Introduction to data mining* (2nd ed.). Pearson.