BAB II
LANDASAN TEORI

2.1 Tinjauan Studi

Banyak penelitian dilakukan dalam menganalisis keranjang pasar untuk rekomendasi produk. Hal ini dapat dilihat dari banyaknya buku-buku, jurnal ilmiah dan *conference* yang melakukan penelitian untuk menganalisis keranjang pasar. Beberapa penelitian yang terkait dengan penelitian ini yaitu penelitian mengenai analisis keranjang pasar pada *e-commerce* [7], penelitian untuk analisis keranjang pasar dengan menggunakan *FP-Growth* [9], dan penelitian menentukan *association rule* dengan menggunakan *matrix database* [10].

Berikut penjabaran dari penelitian-penelitian yang terkait :

1. Judul: *Exploring Customer Preferences with Probabilistic Topics Models* [7]
   State of The Art:
   Christidist et al menganalisis keranjang pasar pada *e-commerce* untuk menentukan rekomendasi produk kepada pelanggan. Pendekatan yang umum digunakan adalah aturan asosiasi, tetapi ada sejumlah masalah, yaitu: aturan asosiasi cenderung mengabaikan *item*set yang besar dan rekomendasi produk kurang tepat karena tidak ada informasi tentang produk ritel. Mereka mencoba menerapkan metode *latent topic*, karena metode ini lebih memperhatikan *item*set dan juga histori penjualan produk.
   Metode yang digunakan adalah 5 topic model dan asosiasi menggunakan algoritma *FP-Growth*. Topik-topik model yang digunakan, yaitu:
   a. *Latent Baskets – Gibbs Sampler* untuk memprediksi prilaku pelanggan berdasarkan dari produk-produk yang ada di keranjang pasar pelanggan.
   b. *Latent Baskets Theasurus* untuk mengkalkulasi produk yang mirip dengan histori keranjang pasar pelanggan tersebut sampai produk yang berbeda. Kemudian membandingkan produk relevan dengan produk yang benar-benar dipilih oleh pelanggan tersebut.

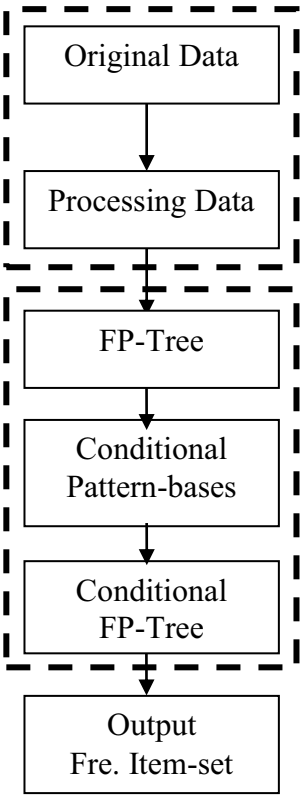
c. *Latent Basket with Co-occurence Boosting*. Hampir sama dengan *Theasurus*, yaitu mengkalkulasi jumlah produk yang mirip dengan histori keranjang pasar pelanggan.
d. *Latent Users Theasurus* untuk memprediksi produk yang akan dibeli yaituberdasarkan produk-produk yang ada di keranjang pasar pelanggan.
e. *Latent Baskets combined with Latent Users*.

Hasil evaluasi menyimpulkan bahwa metode *Latent Baskets – Gibbs Sampler* adalah yang paling efektif untuk rekomendasi produk kepada pelanggan dibandingkan dengan metode lainnya.

2. Judul: *Improved Algorithms Research for Association Rule Based on Matrix* [10]
State of the art:
Meskipun algoritma *Apriori* menggunakan *cut-technology* ketika *generate candidate itemset*, hal tersebut harus dilakukan setiap saat dengan cara melakukan scan database pada setiap transaksi. Kecepatan *scanning* akan menjadi lambat untuk jumlah data yang besar. Peningkatan algoritma *Apriori* berbasis *algoritma matrix*, memiliki ide dasar dengan mengubah kejadian dalam database ke dalam *database matriks* sehingga untuk mendapatkan *item matriks* ditetapkan *itemset* yang maksimal. Ketika menemukan *frequent k-item set*dari *frequent k-1item set*, hanya set matriks ditemukan. Jadi hanya data yang sesuai dihitung untuk mendapatkan *frequent k-item set* yang ditetapkan. Oleh karena itu peningkatan algoritma *Apriori* membutuhkan waktu komputasi sangat cepat. Hasil percobaan membuktikan efisiensi peningkatan algoritma *Apriori*.

3. Judul: *FP-Growth Algorithm for Application in Research of Market Basket Analysis* [9]
State of the art:
Association Rules adalah konten penelitian yang penting dalam data mining. Tetapi produksi *frequent set* adalah langkah pertama sebelum produksi aturan asosiasi. Pada kenyataannya banyak orang umumnya menggunakan algoritma seperti *Apriori*. *Apriori* memiliki kekurangan yang sangat besar, kita harus mendapatkan *frequent sets* yang diproduksi *candidate frequent sets* tanpa henti.

Namun, *cost* dari *candidate frequent sets* sangat luar biasa. Kelebihan algoritma *FP-Growth* adalah bahwa hal itu dapat menghemat waktu dan ruang penyimpanan dan penggunaan sarana partisi untuk menghindari database skala besar. Untuk algoritma *FP-Growth*, biasanya membangun *frequent sets* melalui *FP-Tree*.



Gambar 2.1.1 Metode yang diusulkan Yongmei Liu [11]

Ketika database yang begitu besar dimana *FP-Tree* masih belum dibangun dalam memori setelah database akan dipecah menjadi subset data yang dengan *frequent 1-set*, maka penggalian data akan sulit untuk dilaksanakan dengan lancar. Ide meningkatkan adalah bahwa bagian data yang dengan *frequent 1-set* dipecah menjadi bagian data dengan *frequent 2-set*. Jika *FP-Tree* masih belum dibangun dalam memori setelah dipecah, subset data tidak akan rusak sampai *FP-Tree* dibangun.

2.1.1 Rangkuman Penelitian Terkait

Rangkuman penelitian yang dilakukan pada tinjauan studi di atas dapat dilihat pada tabel berikut.

Tabel 2.1.1 Perbandingan Penelitian terkait

No	Judul dan Peneliti	Tahun	Kesimpulan
1	**Judul:** *Exploring Customer Preferences with Probabilistic Topics Models* **Peneliti:** - Konstantinos Christidis - Dimitris Apostolou - Gregoris Mentzas	**2010**	**Kekurangan:** - Penggunaan *latent topic* tidak berpengaruh pada *itemset* yang besar. **Kelebihan:** - Menerapkan metode *latent topic*, yang lebih memperhatikan *item*set dan juga histori penjualan produk.
2	**Judul:** *Improved Algorithms Research for Association Rule Based on Matrix* **Peneliti:** - LUO XianWen - WANG WeiQing	**2010**	**Kekurangan:** - Perlu mengkonversi database ke dalam bentuk matriks yang rumit sebelum melakukan *Association Rules*. **Kelebihan:** - Mempercepat waktu *scan* database yang dilakukan algoritma *apriori*.
3	**Judul:** *FP-Growth Algorithm for Application in Research of Market Basket Analysis* **Peneliti:** - Yongmei Liu - Yong Guan	**2008**	**Kekurangan:** - Masih diperlukan cara untuk mengatasi dekomposisi database yang besar. **Kelebihan:** - Mengatasi kelemahan Algoritma Apriori dalam melakukan *Association Rules* dengan menggunakan Algoritma *FP-Growth*.

2.1.2 Analisis Keranjang Pasar

Analisis keranjang pasar (juga dikenal sebagai *Association Rule Mining*) merupakan salah satu metode data mining yang berfokus pada menemukan pola pembelian dengan mengekstraksi asosiasi atau kejadian dari data transaksional sebuah toko [1].

Analisis keranjang pasar bermula dari transaksi-transaksi yang berisi satu atau lebih barang/*item*, dan beberapa informasi sementara dari transaksi tersebut. Untuk melakukan analisis keranjang pasar, berikut langkah-langkah:

1. Tentukan nilai *Minimum Support* yang diinginkan. *Minimum Support* merupakan ambang batas minimum jumlah *itemset* yang diperbolehkan, jika jumlah *item*nya di bawah ambang batas maka *item* tersebut akan dieliminasi.

2. Menetapkan *frequent itemset* (kumpulan *item* yang muncul secara bersamaan), dengan cara mengambil *itemset* yang memiliki frekuensi *itemset* minimal sebesar *Minimum Support* sebelumnya.
3. Dari semua *frequent itemset*, hasilkan aturan asosiasi yang memenuhi nilai *Minimum Support.*

Analisis keranjang pasar didasarkan pada tiga matrik: *Support*, *Confidence* dan *Lift*. Ketiga matrik tersebut berasal dari catatan transaksi untuk bisnis [12].

1. *Support*

   Matrik pertama ditetapkan untuk analisis keranjang pasar adalah *Support*, yang merupakan probabilitas dari asosiasi (probabilitas dari dua *item* yang dibeli bersama-sama). *Support* dihasilkan dari berapa kali jumlah *item* A dan B terjadi bersamaan dalam transaksi yang sama dibagi dengan jumlah total dari transaksi tersebut. *Support* dapat dirumuskan sebagai berikut:

$$Support = \mathrm{P}\,(\mathrm{A} \cap \mathrm{B}) = \frac{\text{jumlah transaksi yang memuat A dan B}}{\text{total jumlah transaksi}}$$

2. *Confidence*

   Confidence dihasilkan dari seberapa kuat hubungan produk yang sudah dibeli. *Confidence* dapat dirumuskan sebagai berikut:

$$Confidence = \mathrm{P}\,(\mathrm{B}/\mathrm{A}) = \frac{Support\ (\mathrm{A} \cap \mathrm{B})}{\mathrm{P}\,(\mathrm{A})}$$

3. *Lift*

   Lift Ratio mengukur seberapa penting *rule* yang teleh terbentuk berdasarkan nilai *support* dan *confidence*. *Lift Ratio* merupakan nilai yang menunjukkan kevalidan proses transaksi dan memberikan informasi apakah benar produk A dibeli bersamaan dengan produk B. *Lift Ratio* dapat dirumuskan sebagai berikut:

$$Lift\ Ratio = \frac{\mathrm{Support}\ (\mathrm{A} \cap \mathrm{B})}{\mathrm{Support}\ (\mathrm{A}) * \mathrm{Support}\ (\mathrm{B})}$$

Beberapa keuntungan menggunakan analisis keranjang pasar [13], antara lain:

1. Mengeksplorasi data transaksi.
2. Menentukan hubungan antar produk.
3. Mendeteksi perilaku dan perubahan perilaku pelanggan.

Selain keuntungan tersebut, analisis keranjang pasar juga memiliki beberapa manfaat lain, yaitu:

1. Membuat iklan dan promosi lebih menguntungkan dikarenakan sudah diprediksi bagaimana respon pembeli.
2. Target penawaran yang dilakukan sales menjadi lebih tepat.
3. Meningkatkan lalu lintas pembeli ke toko.
4. Meningkatkan ukuran dan nilai dari keranjang pasar, dapat mengidentifikasi apakah pelanggan akan membeli semua barang atau tidak.
5. Menentukan harga produk dalam toko.
6. Mengoptimasi penjulan produk di toko.

Tujuan utama dari analisis keranjang pasar adalah untuk mengeksploitasi kedekatan produk dengan menginduksi konsumen untuk membeli produk tambahan yang tidak direncanakan berdasarkan pembelian yang sudah dilakukan. Analisis keranjang pasar sering gagal untuk membedakan antara melengkapi (pelanggan produk cenderung untuk membeli bersama-sama) dan pengganti (sepasang produk di mana pelanggan cenderung untuk mengganti satu produk untuk yang lain, maka tidak membeli keduanya bersama-sama). Hal ini membuat fungsional produk pengganti menjadi pelengkap yang aktual. Masalah lainnya adalah sejumlah besar transaksi dalam urusan ukuran yang lebih besar, yang dapat membuat sulit untuk melaksanakan analisis keranjang pasar kecuali subsampel dirancang dengan baik dari himpunan transaksi digunakan sebagai pengganti seluruh database [12].

Untuk menganalisis keranjang pasar, pendekatan yang biasa digunakan adalah aturan asosiasi. Tetapi ada sejumlah masalah teknis yang berhubungan dengan teknik rekomendasi yang paling umum. Aturan asosiasi cenderung mengabaikan *itemset* besar, dan rekomendasi *item* kurang tepat karena informasi tentang produk ritel tidak tersedia [7], sehingga untuk data yang besar hasilnya menjadi kurang akurat. Untuk mengatasi masalah tersebut, atribut yang ada di*cluster* untuk membentuk kelompok atribut yang sama dan kemudian menentukan pola asosiasi pada masing-masing kelompok [5], sehingga dapat mempermudah proses mencari rekomendasi produk.

Pola hubungan antar produk ini berupa *Interesting Rules*. Analisis keranjang pasar memberikan informasi produk apa saja yang sering dibeli oleh konsumen secara bersamaan. Contohnya jika seorang konsumen membeli senter, maka ia juga akan membeli baterai. Informasi-informasi inilah yang digali dari data mining dengan analisis keranjang pasar. Produk-produk yang sering dibeli secara bersamaan dapat ditempatkan secara berdekatan sehingga konsumen dapat dengan mudah menemukan apa yang ia cari. Dengan demikian para konsumen akan merasa puas dan penjualan juga akan meningkat.

2.1.3 Clustering

Mengelompokan *record*, pengamatan dan membentuk kelas obyek-obyek yang memiliki kemiripan. Tujuan dari algoritma *cluster* adalah dengan memecahkan setiap datadalam *dataset* menjadi kelompok kelompok yang homogen. Kelompok data ini biasanya disebut sebagi *cluster*. Setiap *cluster* yang terbentuk akan terdiri dari data yang sejenis dan berbeda dengan data pada *cluster* lainnya [16]. Pengelompokkan ini sama dengan cara kerja otak manusia, dimana ilmu pengetahuan dikelompokkan dalam setiap bidangnya. Dengan adanya pengelompokkan, data yang dapat diolah dengan lebih spesifik sesuai dengan tujuan penelitian. Pemecahan data kedalam *cluster* data juga diterapkan pada tahappengolahan awal data dalam proses data mining, sehingga dapat diterapkan metode data mining data mining kedalam setiap *cluster* data. Proses *clustering* juga dapat mengurangi jumlah ataupun dimensi data yang diolah.

Beberapa alasan dalam pemilihan algoritma *clustering* [16]:

1. Flexibilitas

   Algoritma *clustering* harus dapat mengolah berbagai jenis data, sehingga dapat diterapkan dalam ruang lingkup yang lebih besar.

2. Handal

   Algoritma *clustering* harus dapat menghasilkan *cluster* yang stabil sehingga tidak menimbulkan banyak perbedaan dalam setiap observasi yang dilakukan. Algoritma juga tidak boleh mudah terganggu oleh data yang

tidak relevan. Attribut dan ciri dari setiap *cluster* harus tetap stabil jika diteliti dengan cara yang sama.

3. Efisiensi

   Algoritma *clustering* harus dapat mengolah data baik dalam jumlah besar dengan waktu pengolahan yang sedikit. Algoritma *clustering* juga harus dapat menyesuaikan jumlah attribute yang digunakan dalam *dataset* agar waktu pengolahan dapat diminimalisasi, namun tetap memiliki hasil yang konsisten.

Metode *clustering* dapat diklasifikasikan menjadi beberapa jenis berdasarkan logika *clustering* [16]:

a. Metode partisi

   Baik dalam mengelompokkan data menjadi kelompok yang sudahditentukan lebih dahulu

b. Metode hierarki

   Memecah data kedalam *cluster* dengan struktur hirarki. Data setiap *cluster* tetap homogen, namun memiliki tingkatan antara satu *cluster* dengan *cluster* lainnya

c. Metode berbasis kepadatan

   Merupakan perpaduan kedua metode sebelumnya, metode ini memecah data kedalam partisi berdasarkan jarak data terhadap setiap *cluster*. Namun setiap *cluster* memiliki batasan jarak, sehingga nilai tidak boleh lebih kecil dari nilai minimum *cluster*.

d. Metode grid

   Cluster terbentuk bersarakan sturktur ruangan yang seperti sel. Metode ini dapat mengolah data besar dengan cepat, namun memiliki akurasi yang rendah

2.1.4 Asosiasi

Analisis asosiasi atau association *Rule* mining adalah teknik data mining untuk menemukan aturan assosiatif antara suatu kombinasi *item*. Contoh dari aturan *assosiatif* dari analisa pembelian di suatu pasar swalayan adalah dapat diketahuinya

berapa besar kemungkinan seorang pelanggan membeli roti bersamaan dengan susu. Dengan pengetahuan tersebut pemilik pasar swalayan dapat mengatur penempatan barangnya atau merancang kampanye pemasaran dengan memakai kupon diskon untuk kombinasi barang tertentu. Karena analisis asosiasi menjadi terkenal karena aplikasinya untuk menganalisa isi keranjang belanja di pasar swalayan, analisis asosiasi juga sering disebut dengan istilah analisis keranjang pasar (*market basket analysis)* [17].

Analisis keranjang pasar adalah analisis dari kebiasaan membeli *customer* dengan mencari asosiasi dan korelasi antara item-item berbeda yang diletakkan customer dalam keranjang belanjaannya. Hal tersebut digunakan untuk menganalisa data dalam rangka keperluan strategi pemasaran, desain katalog, dan proses pembuatan keputusan bisnis.

Aturan asosiasi mengcapture item atau kejadian dalam data berukuran besar yang berisi data transaksi. Dengan kemajuan teknologi, data penjualan dapat disimpan dalam jumlah besar yang disebut dengan "*basket* data." Aturan asosiasi yang didefinisikan pada basket data, digunakan untuk keperluan promosi, desain katalog, segmentasi customer dan target pemasaran. Secara tradisional, aturan asosiasi digunakan untuk menemukan trend bisnis dengan menganalisa transaksi customer.

Algoritma asosiasi yang paling populer dikenal sebagai *Apriori* dengan paradigma *generate* and *test*, yaitu pembuatan kandidat kombinasi *item* yang mungkin berdasar aturan tertentu lalu diuji apakah kombinasi *item* tersebut memenuhi syarat support *minimum*. Kombinasi *item* yang memenuhi syarat tersebut disebut *frequent itemset*, yang nantinya dipakai untuk membuat aturan-aturan yang memenuhi syarat *confidence minimum*. Algoritma baru yang lebih efisien bernama *FP-Growth*. Asosiasi digunakan untuk menemukan atribut yang berkaitan atau disebut juga "*go together*". Sebagian besar bisnis menganalisis keranjang pasar menggunakan pola asosiasi untuk menemukan pola hubungan antara dua atribut atau lebih.

Contohnya yaitu pada sebuah supermarket ditemukan 1000 pelanggan yang berbelanja pada hari kamis malam [15], dimana:

- 200 membeli popok
- dari 200 yang membeli popok, 50 dari mereka juga membeli bir.

Sehingga pola asosiasinya yaitu menjadi "Jika beli popok, maka beli bir" dengan perbandingan 200/1000 = 20% dan 50/200 = 25 %.

Untuk mengukur hasil pendekatan asosiasi dalam *data mining*, yaitu:

1. *Support* adalah suatu ukuran yang menunjukan seberapa besar dominasi suatu *item*set dari keseluruhan *dataset*.
2. *Confidence* adalah suatu ukuran yang menunjukan hubungan antar dua *item*.

2.1.5 Algoritma FP-Growth

Algoritma yang biasa dipakai dalam mencari *frequent itemset* antara lain algoritma *Apriori* dan algoritma *FP-Growth*. Pada penelitian ini akan dibahas bagaimana pencarian *frequent itemset* mengunakan algoritma *FP-Growth*. *FP-Tree* (*Frequent Pattern Tree*) digunakan bersamaan dengan algoritma *FP-Growth* untuk menentukan *frequent itemset* (data yang paling sering muncul) dari sebuah *dataset*.

Algoritma *Apriori* memerlukan langkah *candidate generation*, yaitu dengan melakukan *scanning dataset* secara berulang-ulang untuk menentukan *frequent itemset*. Algoritma *FP-Growth* adalah salah satu cara alternatif untuk menemukan himpunan data yang paling sering muncul (*frequent itemset*) tanpa menggunakan generasi kandidat [18].

FP-Growth membangun konstruksi data (*FP-Tree*) yang sangat dikompresi, dan mengurangi data asli. Algoritma *FP-Growth* melakukan *scan* database yang sama sebanyak dua kali. *Scanning* database yang pertama, kita dapat memperoleh *frequent 1-item-set,* dan *scanning* database yang kedua, kita dapat memfilter database *non-frequent item,* selebihnya, *FP-Tree* dihasilkan secara bersamaan. Akhirnya, dapat diperoleh aturan asosiasi dengan menggunakan *FP-Tree* [9].

Kelemahan *Apriori* yang selalu melakukan *scanning* database secara berulang-ulang membuat *Apriori* ini kurang efektif. Berbeda dengan *FP-Tree* yang digunakan bersamaan dengan algoritma *FP-Growth* yang hanya memerlukan dua kali scanning database untuk membuat *frequent itemset*. Dengan menggunakan *FP-Tree*, algoritma *FP-Growth* dapat langsung mengekstrak *frequent Itemset* dari *FP-Tree* yang telah terbentuk.

FP-Tree didefinisikan [19] sebagai berikut:

1. Sebuah *root* yang diberi label *null*, sekumpulan *sub-tree* yang beranggotakan *item-item* tertentu, dan sebuah tabel *frequentheader*.
2. Setiap simpul dalam *FP-Tree* mengandung tiga *field*, yaitu:
   a. *Item-name*: menginformasikan *item* yang dipresentasikan oleh simpul tersebut.
   b. *Count*: mempresentasikan jumlah transaksi yang melewati simpul tersebut.
   c. *Node-Link*: penghubung yang menghubungkan simpul-simpul dengan *item-name* yang sama, atau *null* jika kosong.

Contoh kasus:

Tabel 2.1.2 Data Pembelian

TID	Item bought
100	***{f, a, c, d, g, i, m, p}***
200	***{a, b, c, f, l, m, o}***
300	***{b, f, h, j, o}***
400	***{b, c, k, s, p}***
500	***{a, f, c, e, l, p, m, n}***

Dari tabel di atas bisa dilihat contoh 5 buah transaksi pembelian barang, misalnya: TID 100 membeli barang {f, a, c, d, g, i, m, p}. Untuk menentukan *frequent itemset* pada data transaksi tersebut, dapat dilakukan langkah-langkah berikut ini:

1. Menentukan *Minimum Support*

   Minimum Support merupakan ambang batas minimum jumlah *itemset* yang diperbolehkan, jika jumlah *item*nya di bawah ambang batas maka *item* tersebut akan dieliminasi. Misalnya: min_sup=3/5=60%.

2. Menentukan *Header Frequent Itemset*

   Untuk mendapatkan *header itemset*, *scan itemset* dan hitung frekuensi masing-masing *itemset*. Dari data di atas akan menghasilkan: { c(4), f(4), a(3), b(3), m(3), p(3) }.

Tabel 2.1.3 *Header* Tabel

Item	***frequency***
c	*4*
f	*4*
a	*3*
b	*3*
m	*3*
p	*3*

Sedangkan *itemset* yang dieliminasi dikarenakan tidak memenuhi *Minimum Support*, yaitu: { l(2), o(2), d(1), g(1), i(1), h(1), j(1), k(1), s(1), e(1) }

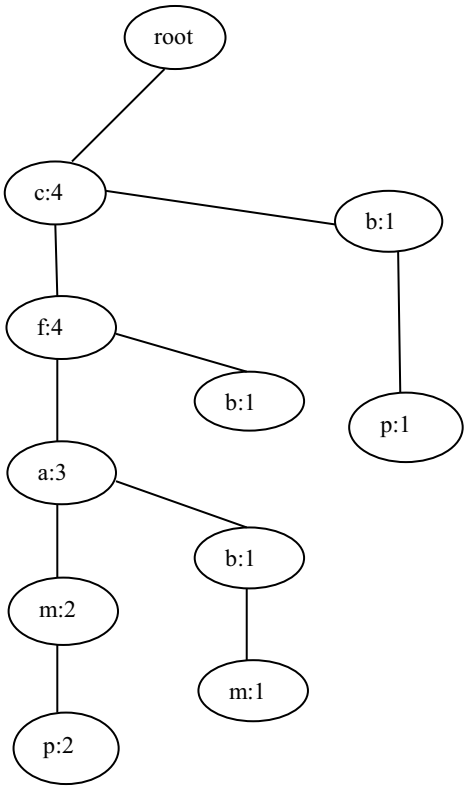
3. Membuat *FP-Tree*

Tabel 2.1.4 Data Transaksi

TID	**Item bought**	***(ordered) frequent items***
100	***{f, a, c, d, g, i, m, p}***	***{c, f, a, m, p}***
200	***{a, b, c, f, l, m, o}***	***{c, f, a, b, m}***
300	***{b, f, h, j, o}***	***{f, b}***
400	***{b, c, k, s, p}***	***{c, b, p}***
500	***{a, f, c, e, l, p, m, n}***	***{c, f, a, m, p}***

FP-Tree dibangun dengan mencari *item* sesuai urutan pada *item* yang *frequent*. Data transaksi tidak perlu diurutkan, dan untuk tiap *item* yang ditemukan bisa langsung dimasukkan ke dalam *FP-Tree*. Sesudah membuat *root*, tiap *item* yang ditemukan dimasukkan berdasarkan path pada *FP-Tree*. Jika *item* yang ditemukan sudah ada, maka nilai *support item* tersebut yang ditambahkan. Namun jika *path* belum ada, maka dibuat *node* baru untuk melengkapi *path* baru pada *FP-Tree* tersebut. Hal ini dilakukan selama *item* pada transaksi masih ada yang *qualified*, artinya memenuhi nilai *Minimum Support*. Jadi, *item-item* yang ditemukan dalam transaksi akan berurutan memanjang kebawah. Dalam struktur *FP-Tree*, diterapkan alur *path* dari *child* hingga ke *root*. Jadi, suatu *path* utuh dalam *FP-*

Tree adalah dari *child* terbawah hingga ke *root*. Tiap *node* pada *FP-Tree* memiliki *pointer* ke *parent*, sehingga pencarian harus dimulai dari bawah.



Gambar 2.1.2 *FP-Tree* [11]

4. Membuat *Conditional Pattern* berdasarkan *FP-Tree*

   Misalnya pada *node* p:1 pada *FP-Tree*, berarti terdapat *pattern* c-b bernilai *support* 1. Kemudian bila ada *pattern* c-b lagi bernilai *support n* yang ditemukan dari *FP-Tree* maka nilai *support* 1 tersebut menjadi *n+1*. Contoh hasil lengkap dari *Pattern Tree* tersebut:

   a. c:4 menggambarkan bahwa ada *pattern* c sebanyak 4
   b. f:4 menggambarkan bahwa ada *pattern* c-f sebanyak 4
   c. a:3 menggambarkan bahwa ada *pattern* c-f-a sebanyak 3
   d. b:1 menggambarkan bahwa ada *pattern* c-f-a-b sebanyak 1
   e. b:1 menggambarkan bahwa ada *pattern* c-f-b sebanyak 1
   f. b:1 menggambarkan bahwa ada *pattern* c-b sebanyak 1
   g. m:2 menggambarkan bahwa ada *pattern* c-f-a-m sebanyak 2
   h. m:1 menggambarkan bahwa ada *pattern* c-f-a-b-m sebanyak 1

i. p:2 menggambarkan bahwa ada *pattern* c-f-a-m-p sebanyak 2

j. p:1 menggambarkan bahwa ada *pattern* c-b-p sebanyak 1

Berikut *Conditional Pattern Base* bila dimulai dari *Head Item* yang ada:

Tabel 2.1.5 *Conditional Pattern Base*

Head Item	***Condition pattern base***
f	*c:4*
a	*cf:3*
b	*cfa:1, cf:1, c:1*
m	*cfa:2, cfab:1*
P	*cfam:2, cb:1*

5. Menentukan *Frequent Item-set*

Pada *condition pattern base*, dari awal *p-item*, setiap *item* dari *condition pattern base* di *scan*. *Pattern* yang tidak memenuhi *Minimum Support*, dihapus dari daftar *pattern*. *Pattern-pattern* yang tersisa kemudian diurutkan untuk memudahkan pembuatan *rules*. Pada saat yang sama, jumlah *item* yang sesuai dihitung dan kondisi *FP-Tree* dihasilkan. Dan kemudian *FP-Tree* terhubung dengan *Head-item*, dan akhirnya menghasilkan *frequent item-set*.

Tabel 2.1.6 *Frequent Item-set*

Head Item	***condition pattern base***	***condition FP-Tree***	***Frequency Item***
p	*cfam:2, cb:1*	*<c:3>*	*cp:3*
m	*cfa:2, cfab:1*	*<c:3, f:3, a:3, cf:3, ca:3, fa:3, fa:3>*	*cm:3, fm:3, am:3, cfm:3, cam:3, fam:3, cfam:3*
b	*cfa:1, cf:1, c:1*	*<c:3>*	*cb:3*
a	*cf:3*	*<cf:3>*	*ca:3, fa:3, cfa:3*
f	*c:4*	*<c:4>*	*cf:4*
c	ϕ	ϕ	ϕ

2.1.6 Algoritma K-Medoids

Untuk melakukan *clustering* dengan metode partisi dapat menggunakan *K-Means* dan *K-Medoids*. *K-Means* merupakan suatu algoritma peng*cluster*an yang cukup sederhana yang mempartisi *dataset* kedalam beberapa *cluster* k. Algoritmanya cukup

mudah untuk diimplementasi dan dijalankan, relatif cepat, mudah disesuaikan dan banyak digunakan [18].

Kelemahan-kelemahan dari algoritma *K-Means* [20] yaitu:

1. Ketika jumlah data yang tidak begitu banyak, pengelompokan awal akan menentukan *cluster* secara signifikan.
2. Jumlah *cluster* K harus ditentukan terlebih dahulu.
3. *Cluster* yang asli tidak diketahui, dengan menggunakan data yang sama, jika dimasukkan dalam urutan yang berbeda dapat menghasilkan *cluster* yang berbeda jika jumlah data sedikit.
4. Kelemahan dari aritmatika mean tidak kuat untuk *outlier*, sangat jauh data dari *centroid* memungkinkan mempengaruhi *centroid* yang asli.

Algortima *K-Medoids*, juga dikenal sebagai *partitioning around Medoids*, adalah varian dari metode *K-Means.* Hal ini didasarkan pada penggunaan *Medoids* bukan dari pengamatan *mean* yang dimiliki oleh setiap *cluster,* dengan tujuan mengurangi sensitivitas dari partisi yang dihasilkan sehubungan dengan nilai-nilai ekstrim yang ada dalam *dataset* [16].

Algoritma *K-Medoids* hadir untuk mengatasi kelemahan Algoritma *K-Means* yang sensitif terhadap *outlier* karena suatu objek dengan suatu nilai yang besar mungkin secara substansial menyimpang dari distribusi data [21].

Menurut Han dan Kamber, algoritma *K-Medoids* adalah sebagai berikut [21].

1. Secara acak pilih *k* objek pada sekumpulan *n* objek sebagai *medoid.*
2. Ulangi:
3. Tempatkan objek *non-medoid* ke dalam *cluster* yang paling dekat dengan *medoid.*
4. Secara acak pilih Orandom: sebuah objek *non-medoid.*
5. Hitung total *cost*, *S*, dari pertukaran *medoid* oj dengan Orandom.
6. Jika $S < 0$ maka tukar oj dengan Orandom untuk membentuk sekumpulan *k* objek baru sebagai *medoid.*
7. Hingga tidak ada perubahan.

Contoh kasus:

Tabel 2.1.7 Data Objek

X_1	2	6
X_2	3	4
X_3	3	8
X_4	4	7
X_5	6	2
X_6	6	4
X_7	7	3
X_8	7	4
X_9	8	5
X_{10}	7	6

Langkah-1:

1. Inisialisasi K sebagai *Medoid.*

   Misal:

   c_1 = (3,4) and c_2 = (7,4)

2. Hitung jarak terdekat setiap objek dengan *Medoid.*

   Contoh perhitungan:

   c_1 = (3,4) and X_1= (2,6)

   Cost (*distance*) = (3-2) + (4-6)

   = 1 + 2

   = 3

Tabel 2.1.8 *Distance* c_1 = (3,4) dengan Data Objek

i	c_1		Data Objects (X_i)		Cost (distance)
1	3	4	2	6	**3**
3	3	4	3	8	**4**
4	3	4	4	7	**4**
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	3	5
9	3	4	8	5	6
10	3	4	7	6	6

Tabel 2.1.9 *Distance* c_2 = (7,4) dengan Data Objek

i	c_2		Data Objects (X_i)		Cost (distance)
1	7	4	2	6	7
3	7	4	3	8	8
4	7	4	4	7	6
5	7	4	6	2	**3**
6	7	4	6	4	**1**
7	7	4	7	3	**1**
9	7	4	8	5	**2**
10	7	4	7	6	**2**

Maka, *cluster* yang dihasilkan dari langkah-1:

$Cluster_1$ = {(3,4)(2,6)(3,8)(4,7)}

$Cluster_2$ = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}

Dikarenakan poin (2,6), (3,8) dan (4,7) lebih dekat ke c1 maka mereka membentuk satu *cluster* sementara poin yang tersisa membentuk *cluster* lain.

3. Hitung total jarak dari langkah-1.

Untuk menghitung total jarak dengan rumus:

$$cost(x, c) = \sum_{i=1}^{d} | Xi - Ci|$$

Keterangan:

x = data objek

c = *medoid*

d = dimensi dari objek

Maka, total jarak dari langkah-1:

Total cost = {cost ((3,4),(2,6)) + cost ((3,4),(3,8)) + cost ((3,4),(4,7))}
+ {cost ((7,4),(6,2)) + cost ((7,4),(6,4)) + cost ((7,4),(7,3))
+ cost ((7,4),(8,5)) + cost ((7,4),(7,6))}
= (3+4+4) + (3+1+1+2)
= 20

Langkah-2:

1. Tentukan non *Medoids* O'

   Misal:

   O'= (7,3)

   Jadi sekarang *Medoids* adalah c1 (3,4) dan O'(7,3)'

2. Hitung total jarak seperti langkah-1.

Tabel 2.1.10 *Distance* c_1 = (3,4) dengan Data Objek

i	c_1		Data Objects (X_i)		Cost (distance)
1	3	4	2	6	**3**
3	3	4	3	8	**4**
4	3	4	4	7	**4**
5	3	4	6	2	5
6	3	4	6	4	3
7	3	4	7	4	4
9	3	4	8	5	6
10	3	4	7	6	4

Tabel 2.1.11 *Distance* O'= (7,3) dengan Data Objek

i	O’		Data Objects (X_i)		Cost (distance)
1	7	3	2	6	8
3	7	3	3	8	9
4	7	3	4	7	7
5	7	3	6	2	**2**
6	7	3	6	4	**2**
8	7	3	7	4	**1**
9	7	3	8	5	**3**
10	7	3	7	6	**3**

total cost = 3 + 4 + 4 + 2 + 2 + 1 + 3 + 3

= 22

3. Hitung jarak pertukaran *medoid* dari c_2 ke O′

   S = current total cost – past total cost

   = 22 – 20

   = 2 > 0.

Dikarenakan S > 0 maka pertukaran *Medoid* selesai dan *cluster* yang dihasilkan kembali pada langkah-1, yaitu:

$Cluster_1$ = {(3,4)(2,6)(3,8)(4,7)}

$Cluster_2$ = {(7,4)(6,2)(6,4)(7,3)(8,5)(7,6)}

2.1.7 Evaluasi Support, Confidence, Lift Ratio

Support dihasilkan dari berapa kali jumlah *item* A dan B terjadi bersamaan dalam transaksi yang sama dibagi dengan jumlah total dari transaksi tersebut [12]. *Support* dapat dirumuskan sebagai berikut:

$$Support\ (\mathrm{A} \cap \mathrm{B}) = \frac{\text{jumlah transaksi yang memuat A dan B}}{\text{total jumlah transaksi}}$$

Confidence dihasilkan dari seberapa kuat hubungan produk yang sudah dibeli [12]. *Confidence* dapat dirumuskan sebagai berikut:

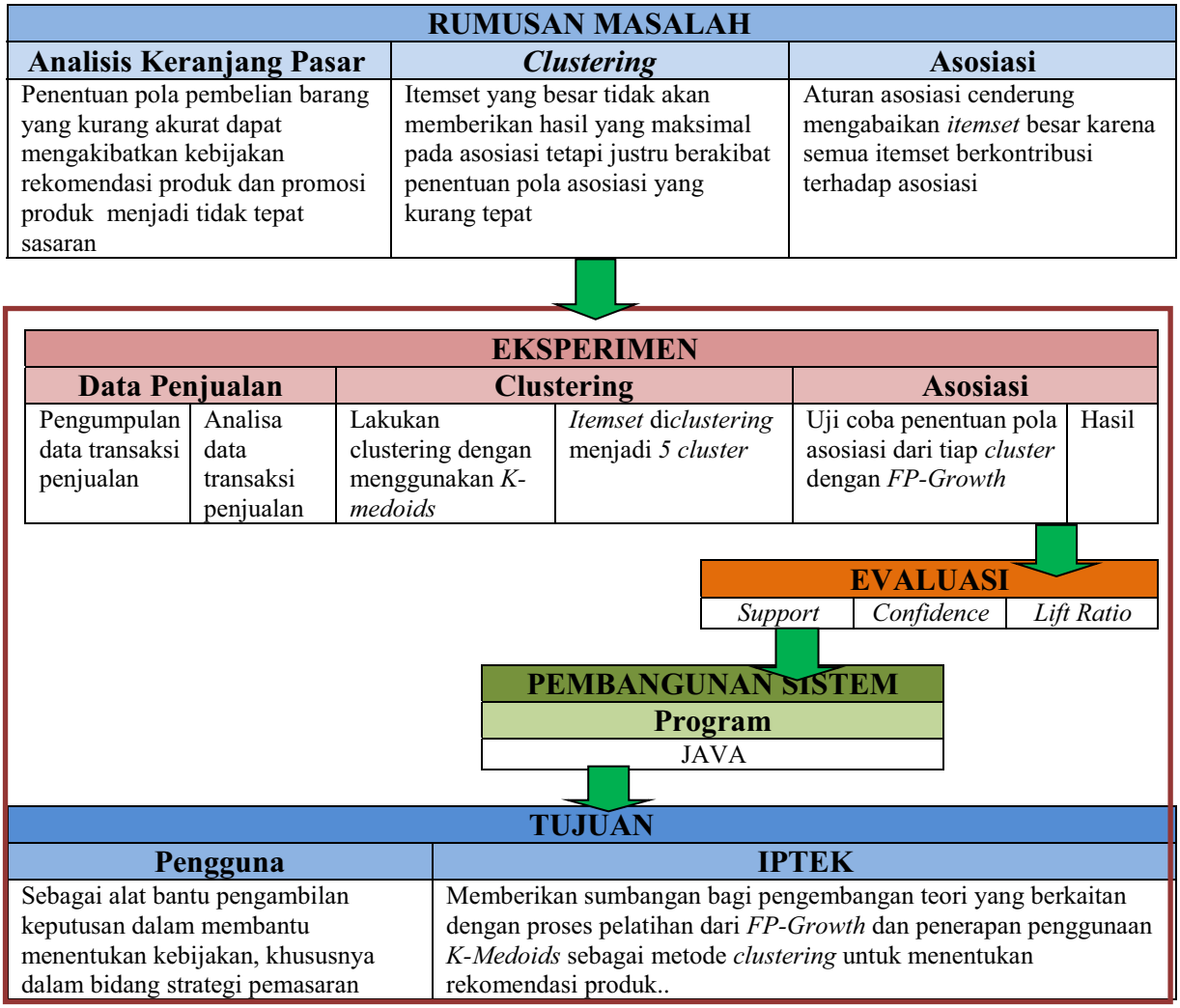
$$Confidence\ (\mathrm{A} \rightarrow \mathrm{B}) = \frac{Support\ (\mathrm{A} \cap \mathrm{B})}{\mathrm{Support\ (A)}}$$

Lift Ratio mengukur seberapa penting *rule* yang telah terbentuk berdasarkan nilai *support* dan *confidence*. *Lift Ratio* merupakan nilai yang menunjukkan kevalidan proses transaksi dan memberikan informasi apakah benar produk A dibeli bersamaan dengan produk B. Sebuah transaksi dikatakan valid jika mempunyai nilai *Lift Ratio* lebih dari 1, yang berarti bahwa dalam transaksi tersebut, produk A dan B benar-benar dibeli secara bersamaan [12] . *Lift Ratio* dapat dirumuskan sebagai berikut:

$$Lift\ Ratio\ (A - B) = \frac{\mathrm{Support\ (A} \cap \mathrm{B)}}{\mathrm{Support\ (A)} * \mathrm{Support\ (B)}}$$

2.1.8 Kerangka Pemikiran

Berdasarkan pandangan diatas maka kerangka pemikiran yang dihasilkan adalah sebagai berikut:



Gambar 2.1.3 Kerangka Pemikiran

Pada penelitian ini, digunakan data transaksi penjualan dari http://inf.abdn.ac.uk/~hnguyen/teaching/CS5553/prac05.php yang bersifat *public*. Metode yang diusulkan adalah menggunakan algoritma *K-Medoids* untuk *clustering* pada data penjualan dan menerapkan algoritma *FP-Growth* untuk pendekatan asosiasi pada setiap *cluster*.

Untuk meningkatkan akurasi menggunakan aturan asosiasi dengan *FP-Growth* pada *dataset* yang besar, maka *dataset* di*clustering* dahulu menjadi *5 cluster*. Tujuannya untuk mengecilkan *dataset* dalam proses asosiasi.

Masing-masing dari *cluster* yang terbentuk akan dilakukan proses asosiasi menggunakan algoritma *FP-Growth* untuk menentukan rekomendasi produk kepada pelanggan. Hasil dari proses asosiasi ini diukur menggunakan *Support*, *Confidence*, *Lift Ratio*. Akurasi dari penerapan algoritma *FP-Growth* yang di*clustering* dahulu menggunakan *K-Medoids* akan dibandingkan dengan akurasi dari penerapan algoritma *FP-Growth* saja.