BAB II
TINJAUAN PUSTAKA

2.1 E-Commerce (Electronic Commerce)

Electronic Commerce (E-Commerce) adalah bagian dari *E-Bussiness,* definisi *E-Commerce* adalah penggunaan media telekomunikasi elektronik untuk pembelian dan penjualan barang atau jasa, contohnya seperti internet dan media telekomunikasi lainya. *E-Commerce* melibatkan semua jenis transaksi yang membutuhkan transmisi digital dari pertukaran informasi. Ketika transaksi terjadi melalui media elektronik, produk yang dibeli mungkin dikirimkan menggunakan alur pengiriman secara tradisional seperti jasa pengiriman atau mekanisme digital seperti mengunduh produk dari internet (Greenstein, 2002).

2.2 Sistem Rekomendasi

Sistem rekomendasi digunakan oleh *web Commerce* untuk merekomendasikan produk kepada konsumen mereka dan menyediakan konsumen dengan informasi yang membantu mereka memilih produk mana yang akan dibeli. Produk dapat direkomendasikan berdasarkan pada tingkat penjualan tertinggi website, sesuai dengan kondisi demografis dari konsumen atau dari analisis perilaku transaksi pembelian sebagai prediksi perilaku pembelian (Schafer, 2000).

Bentuk dari rekomendasi termasuk menyarankan produk kepada konsumen, menyediakan informasi produk, dan review dari pengalaman konsumen yang telah membeli. Proses yang termasuk dalam sistem rekomendasi seperti secara manual membentuk daftar penjualan atau secara automatis dibentuk oleh sistem komputer atau yang disebut *Automatic Recommender System*. *Automatic Recomender System* adalah sistem data mining yang telah dioptmiasi untuk berinteraksi kepada konsumen daripada penjual. Sistem rekomendasi meningkatakan penjualan pada *web Commerce* dengan 3 jalan (Schafer, 2000):

1. *Converting Browsers into Buyers*: pengunjung web sering kali melihat-lihat pada *website* tanpa membeli apapun, sistem rekomendasi dapat membantu konsumen untuk mencari produk yang ingin mereka beli.
2. *Increasing Cross-Sell:* sistem rekomendasi meningkatkan cross-sell produk dengan menyarankan produk tambahan pada konsumen untuk dibeli, jika rekomendasinya bagus maka rata-rata jumlah

pemesanan akan meningkat. Contohnya, *website* mungkin menambahkan produk tambahan pada proses *checkout* berdasarkan produk yang telah ada pada *shopping cart*.

3. *Building Loyalty*: memperoleh loyalitas konsumen adalah penting untuk strategi bisnis ditengah persaingan *web Commerce*, sistem rekomendasi meingkatkan loyalitas konsumen dengan membentuk hubungan anatara *website* dengan konsumen, dengan mempelajari informasi yang berhubungan dengan konsumen mereka, kemudian menggunakan sistem rekomendasi untuk memproses informasi tersebut dan menghadirkan tampilan produk yang cocok dengan kebutuhan konsumen, sehingga konsumen dapat membeli produk yang sesuai dengan yang mereka butuhkan.

2.3 Data Mining

Kemajuan dalam pengumpulan data dan teknologi penyimpanan yang cepat memungkinkan organisasi menghimpun jumlah data yang sangat luas. Alat dan teknik analisis data tradisional tidak dapat digunakan untuk mengekstrak informasi dari data yang sangat besar. Untuk itu diperlukan suatu metode baru yang dapat menjawab kebutuhan tersebut.

Data Mining merupakan teknologi yang menggabungkan metode analisis tradisional dengan algoritma yang canggih untuk memproses data dengan jumlah besar. Ada beberapa definisi dari *Data Mining* yang dikenal antara lain:

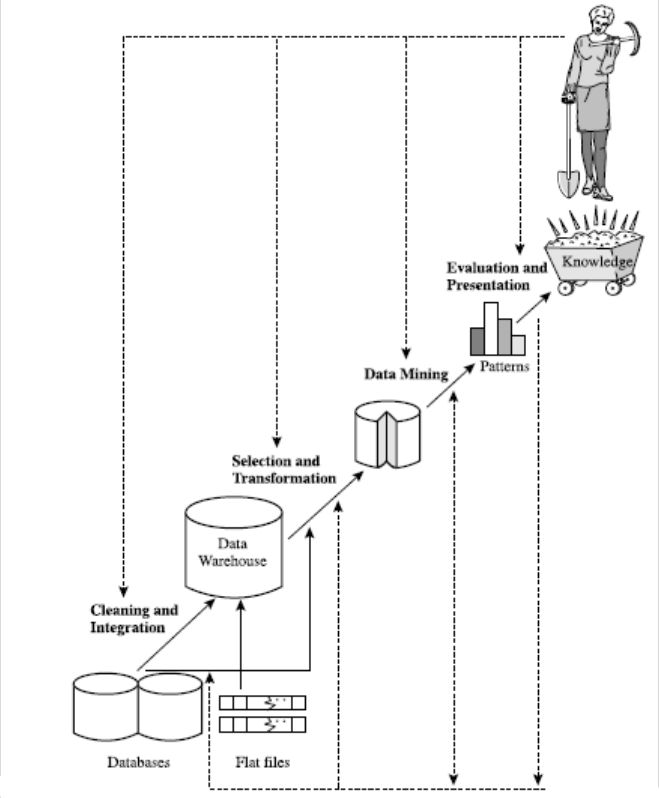
1. *Data Mining* adalah proses menemukan pengetahuan yang menarik dari data dengan jumlah yang besar yang disimpan dalam *database*, *data warehouse* atau penyimpanan informasi lainya (Han, 2006).
2. *Data Mining* adalah proses iteratif dimana kemajuannya ditentukan melalui penemuan melalui metode otomatis atau metode manual (Kantardzic, 2003).

Banyak orang-orang yang memperlakukan *Data Mining* sebagai sinonim untuk istilah lain yang populer seperti *Knowledge Discovery From Data (KDD)*. Pada pandangan yang lain orang mengangap *Data Mining* hanya sebuah langkah penting dalam proses *Knowledge Discovery*. *Knowledge Discovery* terdiri dari urutan iteratif dari langkah-langkah dibawah (Han, 2006):

1. *Data Cleaning* yaitu proses untuk menghilangkan *noise* dan data yang tidak konsisten.
2. *Data Integration* yaitu proses dimana beberapa sumber data dapat dikombinasikan.

3. *Data Selection* dimana data yang sesuai dengan proses analisis diambil dari *database*.
4. *Data Transformation* yaitu proses dimana data ditransformasikan ke bentuk yang sesuai untuk proses *mining*.
5. *Data Mining* yaitu sebuah proses penting dimana metode cerdas diterapkan untuk mengekstrak pola data.
6. *Pattern Evaluation* yaitu proses untuk mengidentifikasi tingkat *interesting patterns* berdasarkan pada beberapa *interestingness measure*.
7. *Knowledge Presentation* yaitu teknik visualisasi dan representasi *knowledge* digunakan untuk menyajikan hasil dari *mining knowledge* kepada pengguna.

Ilustrasi langkah langkah penemuan *Knowledge Discovery* dapat dilihat pada gambar 2.2.



Gambar 2.1 Langkah Penemuan Knowledge Discovery (Han, 2006)

2.4 Association Rule

Association Rule adalah teknik *Data Mining* untuk menemukan aturan *associative* antara suatu kombinasi *item* dalam suatu data set yang ditentukan (Han, 2006). Proses Association Rule meliputi dua tahap:

a. Mencari kombinasi yang paling sering terjadi dari suatu *itemset* (Frequent *Item*set).
b. Membangkitkan *association rule* dari *frequent itemset* yang telah dibuat sebelumnya.

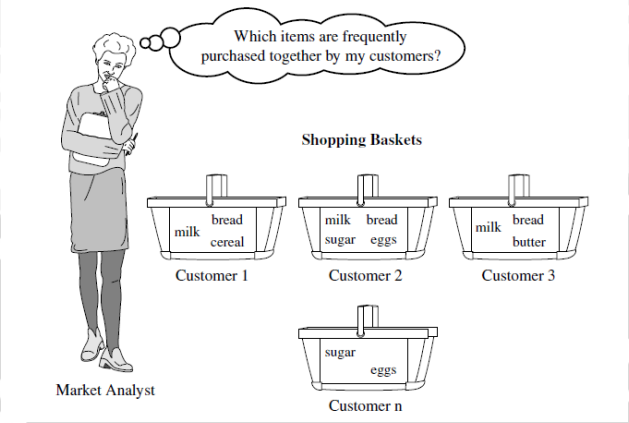
2.5 Frequent Patterns

Frequent patterns adalah pola-pola seperti kumpulan *item* (*Itemsets)* yang sering muncul dalam sebuah data, contohnya sekumpulan *item* seperti susu dan roti yang sering muncul bersama dalam sebuah kumpulan data transaksi adalah *frequent itemset* (Han, 2006).

Pencarian *frequent itemset* memainkan peranan penting dalam proses pencarian *rule, correlations*, dan banyak hubungan menarik lainya dalam data. Selain itu, *frequent itemset* juga membantu dalam klasifikasi data, *Clustering*, dan berbagai metode *Data Mining* lainya.

Salah satu contoh dari *frequent itemset* adalah analisis dari keranjang belanja. Proses ini menganalisa kebiasaan membeli pelanggan dengan mencari hubungan diantara *item-item* berbeda yang ditempatkan oleh pelanggan dalam keranjang mereka, ilustrasi analisis keranjang belanja ini dapat dilihat pada gambar 2.3.

Penemuan hubungan ini dapat membantu para penjual untuk mengembangkan strategi pemasaran mereka, sebagai contoh jika pelanggan membeli susu, bagaimana mereka juga membeli roti pada transaksi yang sama? Informasi seperti ini dapat membawa pada peningkatan keuntungan penjualan dengan membantu penjual dalam pemasaran dan perencanaan posisi barang.



Gambar 2.2 Ilustrasi Analisis Keranjang Belanja (Han,2006)

Pola pembelian yang mencerminkan *item* yang dibeli secara bersamaan dapat diwakilkan dalam bentuk dari *asscociation rule.* Sebagai contoh, informasi bahwa pelanggan yang membeli susu cenderung membeli roti pada saat bersamaan dilambangkan dalam *association rule*:

Susu => Roti [*Support* 2%, *Confidence* 60%].

Support *rule* dan Confidence adalah dua ukuran dari *interestingness rule*. Mereka masing masing mencerminkan manfaat dan kepastian dari *rule* yang ditemukan. Dari persamaan diatas diketahui Support 2% berarti bahwa dari semua transaksi pembelian yang dianalisis peluang pembelian susu dan roti yang dibeli bersamaan adalah 2% dan Confidence 60% menunjukkan bahwa 60% dari pelanggan yang membeli susu juga membeli roti.

Biasanya, *rule* dianggap menarik (*Interesting*) jika mereka memenuhi diantara batas Support minimal dan batas Confidence minimal, batas seperti itu dapat di atur oleh pengguna atau pakar. Analisis lebih lanjut dapat dilakukan untuk menemukan informasi penting lainya. Salah satu algoritma popular yang membangkitkan *rule* dari *frequent itemset* yang dihasilkan adalah Algoritma Apriori.

2.6 Algortima Apriori

Algoritma Apriori menghitung *frequent itemset* dalam *database* melalui beberapa iterasi. Terdapat dua tahapan pada setiap iterasinya yaitu: pembangkitan kanditat dan perhitungan dan seleksi kandidat (Kantardzic, 2003).

Dalam tahapan pertama dari iterasi pertama, kumpulan kandidat *frequent itemset* yang dihasilkan mengandung semua *1-itemsets*. Dalam tahap penghitungan, algoritma Apriori menghitung semua nilai Support dengan melakukan pencarian kembali di dalam *database*. Terakhir, hanya 1-*item*sets dengan Support diatas batas yang dikehendaki dipilih sebagai *frequent itemset*.

Berdasarkan pengetahuan tentang *infrequent itemsets* yang didapat dari iterasi sebelumnya, algoritma Apriori mengurangi *itemset* dengan melakukan pemangkasan (*pruning)* pada kandidat *itemset* yang tidak bisa menjadi *frequent itemset*.

Tabel 2.1 Contoh Database Transaski

TID	*Item*s
001	ACD
002	BCE
003	ABCE
004	BE

Berdasarkan Tabel 2.1, diasumsikan bahwa Support minimal Sup=50% maka sebuah *itemset* dikatakan *frequent* jika mengandung kurang lebih 50% dari transaksi seperti dalam contoh. Di setiap iterasi, algoritma Apriori membentuk kandidat *itemset* di setiap iterasinya, menghitung jumlah kemunculan setiap kandidat, dan menentukan *frequent itemset* berdasarkan Support minimal yang telah ditentukan sebelumnya *Sup*=50%.

Algoritma Apriori sederhana dan mudah untuk diterapkan. Akan tetapi, terdapat beberapa masalah dalam algoritma Apriori, antara lain (Deng, 2010):

1. Algoritma ini membutuhkan berulangkali melakukan *database scanning* untuk membentuk kandidat *itemset*. Dalam proses *scanning*, untuk menghasilkan L_k, dibutuhkan *database scan* sebanyak k kali, sehingga sangat mempengaruhi kinerja algoritma.
2. Algoritma ini kurang effisien karena Algoritma ini menghasilkan banyak perulangan *rule* sehingga dibutuhkan waktu komputasi yang lebih lama.

2.7 Algoritma Matrix and Interestingness based Association Rule Mining (MIbARM)

Untuk meningkatkan kinerja metode Association Rule dan meningkatkan kualitas rekomendasi maka algoritma MIbARM diterapkan. Algoritma ini hanya sekali melakukan *database scannning* kemudian membuang *non-frequent items* dalam proses *mining* untuk kompresi ruang pencarian dan menghindari perulangan kandidat *item*set. Algoritma MIbARM dalam prosesnya mempunyai dua tahap yaitu: *frequent itemset mining* dan *association rule generation*.

2.7.1 Frequent *Item*set Mining berdasarkan Transaction Matrix

Untuk menghindari terjadinya *database scanning* secara berulang-ulang dalam pembentukan kandidat *itemset* maka digunakan Transaction Matrix *(T)* untuk menyimpan data transaksi, dimana dalam prosesnya metode MIbARM hanya melakukan *database scanning* sekali yang disimpan dalam sebuah Transaction Matrix *(T)* kemudian menghilangkan *non-frequent items* dalam proses *mining* untuk menghemat ruang pencarian dan untuk menghindari pengulangan kandidat *itemset*. Algoritma MIbARM ini telah ditingkatkan untuk menangani *frequent k-Itemset (k>2).* Dalam pembentukan *frequent itemset* terdapat beberapa tahapan, antara lain (Deng, 2010):

2.7.1.1 Transaction Matrix (T)

Proses pertama ini merubah data transaksi dalam *database* menjadi bentuk Transaction Matrix *(T)* dua dimensi yang akan digunakan dalam proses *mining* selanjutnya. Pembentukan matriks ini ditunjukkan pada persamaan 2.1, dimana elemen baris dalam matriks transaksi adalah transaksi-transaksi dalam *database (D)* dan $I=\{I_j|j=1,2,\ldots,n\}$ adalah kumpulan *n item* yang berbeda dalam *database*. Jika baris (*i-th)* transaksi (T_i) termasuk pada *item* (I_j), maka elemen pada *matrix* $T(t_{ij})=1$, jika tidak maka elemen $t_{ij}=0$.

$$T=(T_1,T_2,\ldots,Tm)'=(t_{ij})_{mxn} \quad (2.1)$$

Dimana:

T = Transaction Matrix

$T_1,T_2,\ldots,T_m$= transaksi yang berbeda pada *database*

t_{ij}= *entry* yang terjadi pada intreseksi di baris ke *(i)* dan kolom ke *(j)* pada matriks T

m= jumlah baris

n = jumlah kolom

$$T=(t_{ij})_{mxn}\begin{bmatrix} t1 \\ t2 \\ \vdots \\ \vdots \\ \vdots \\ tm \end{bmatrix}\begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1j} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2j} & & t_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{i1} & t_{i2} & \cdots & t_{ij} & \cdots & t_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{mj} & & t_{mn} \end{bmatrix} \quad (2.2)$$

2.7.1.2 Absolute Support (SupA)

Absolute Support (*SupA)* adalah jumlah kemunculan *item* (I_j) dalam matriks *T*. Setelah Transaction Matrix *(T)* terbentuk, langkah selanjutnya adalah menghitung semua Absolute Support dengan menjumlahkan semua element dari kolom ke *j* (*j-th)* pada matriks *T*, penghitungan Absolute Support ditunjukkan pada persamaan 2.3**.**

$$SupA(Ij) = \sum_{i=1}^{m} t_{ij,}\ i = 1,2,\dots,m \quad (2.3)$$

Dimana:

SupA = Absolute Support atau jumlah kemunculan *item*

SupA(I_j)= Absolute Support dari *item* I_j

I_j = kumpulan *item* yang berbeda dalam database *j=1,2,…n*

2.7.1.3 Support (Sup)

Setelah didapat SupA kemudian dihitung juga Support *(Sup)* setiap *item*(i_j) dalam matriks *T*. Support dari sebuah *item* adalah *SupA* dibagi dengan jumlah semua transaksi *m*. Untuk menghitung *Sup* setiap *item* dapat dilihat pada persamaan 2.4.

$$Sup(Ij) = \frac{SupA(I_j)}{m} = \frac{\sum_{i=1}^{m} t_{ij}}{m} \quad (2.4)$$

Dimana:

Sup = Support

Sup(I_j)=Support dari *item*(I_j)

SupA(I_j)= Absolute Support *item*(I_j)

m=jumlah transaksi

2.7.1.4 Mining Frequent 1-*Item*set

Frequent 1-itemset (L_1) didapatkan dari proses pencarian kandidat *Frequent 1-Itemset* (C_1) dengan menghilangkan kolom dari *item* (i_j) pada *T* yang memiliki Support *(Sup)* yang kurang dari minimal Support *(*Sup_{min}*)* yang ditetapkan. Persaman dari pencarian L_1 dapat dilihat pada persamaan 2.5.

$$L1 = \{I_j | Sup(I_j) \geq Sup_{min}, j = 1,2,\dots,n\} \quad (2.5)$$

Dimana:
L_1= *Frequent 1-itemset*
I_j= *item* dalam matriks *T*
Sup= Support *item*
Sup_{min} = minimal Support

Setelah dihilangkan kolom termasuk elemen *item* (I_j) yang mempunyai Support yang kurang dari Support minimal maka didapatkan *frequent 1-itemset* (L_1).

$$T=(t_{ij})_{mxn} \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_i \\ \vdots \\ t_m \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1j} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2j} & & t_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{i1} & t_{i2} & \dots & t_{ij} & \dots & t_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \dots & t_{mj} & & t_{mn} \end{bmatrix}$$

Langkah selanjutnya adalah menghilangkan elemen baris dalam matriks *T* yang jumlahnya kurang kurang dari 2 ($\sum_{j=1}^{n} t_{ij} < 2$).

$$T=(t_{ij})_{mxn} \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_i \\ \vdots \\ t_m \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & \dots & t_{1j} & \dots & t_{1n} \\ t_{21} & t_{22} & \dots & t_{2j} & & t_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{i1} & t_{i2} & \dots & t_{ij} & \dots & t_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ t_{m1} & t_{m2} & \dots & t_{mj} & & t_{mn} \end{bmatrix}$$

Jumlah baris dinotasikan m_2 dan kolom dengan dan n_2 ($m_2 \leq m$, $n2 \leq n$).

2.7.1.5 Mining Frequent 2-*Item*sets

Untuk mencari *frequent 2-itemset* (C_2) yang pertama adalah membentuk sebuah Hermite *matrix* (*H*) dari matriks *T*. Persamaan ini ditunjukkan pada persamaan 2.6.

$$H = T'T = \begin{bmatrix} \sum_{i=1}^{m2} t_{i1}t_{i2} & \sum_{i=1}^{m2} t_{i1}t_{i2} & \cdots & \sum_{i=1}^{m2} t_{i1}t_{in_2} \\ \sum_{i=1}^{m2} t_{i2}t_{i1} & \sum_{i=1}^{m2} t_{i2}t_{i2} & \cdots & \vdots \\ \vdots & \cdots & \ddots & \vdots \\ \sum_{i=1}^{m2} t_{in_2}t_{i1} & \cdots & \cdots & \sum_{i=1}^{m2} t_{in_2}t_{in_2} \end{bmatrix} \quad (2.6)$$

Setiap *non-diagonal element* pada matriks *H* adalah Absolute Support *(SupA)* dari setiap anggota kandidat *frequent 2-item*set *(C_2) SupA(C_2),* sedangkan diagonal *element* adalah Absolute Support (*SupA*) setiap *item* pada *frequent 1-itemset Sup(L_1)*.

Setelah terbentuk matriks *H*, proses selanjutnya adalah kemudian membentuk *symetric matrix G=(g_{ij})* yang dibentuk sesuai dengan Hermite *matrix (H)* untuk mendapatkan *frequent 2-itemset (L_2),* persamaan ini dapat dilihat pada persamaan 2.7.

$$\mathrm{G} = g_{ij} = g_{ji} = \begin{cases} 1 & h_{ij} \geq Sup_{min} \quad i \neq j \\ 0 & h_{ij} < Sup_{min} \quad i \neq j \\ 0 & i = j \end{cases} \quad (2.7)$$

Dimana:
- Nilai 1 diberikan pada elemen g_{ij} matriks *G* jika elemen h_{ij} pada matriks *H* kurang dari Sup_{min} dengan catatan baris (*i)* tidak sama dengan kolom (*j).*
- Nilai 0 diberikan pada element g_{ij} jika elemen h_{ij} pada matriks *H* kurang dari Sup_{min} dengan catatan baris (*i)* tidak sama dengan kolom *(j).*
- Nilai 0 diberikan pada element g_{ij} matriks *G* jika *i* sama dengan *j*.

$$\mathrm{G}=g_{ij}=g_{ji} \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1j} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2j} & \cdots & g_{2n} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{i1} & g_{i2} & \cdots & g_{ij} & \cdots & g_{in} \\ \vdots & \vdots & \ddots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nj} & \cdots & g_{nn} \end{bmatrix} \quad (2.8)$$

Frequent 2-itemset (L_2) didapatkan dari matriks *G*, dimana elemen g_{ij} bernilai 1, $L_2=\{C_2|g_{ij}=1,i,j=1,2,,,,P_1,i\neq j\}$. Langkah terakir baris dalam *T* yang jumlahnya kurang dari 3, $\sum_{j=1}^{n} t_{ij} < 3$ dibuang. Banyaknya baris dan kolom kolom dinotasikan dengan m_3 dan n_3.

2.7.1.6 Mining Frequent k-Itemsets

Pertama sebuah matriks $\boldsymbol{U^k}$ dibentuk untuk pembentukan kandidat k-*Item*sets (C_k) yang berasal dari *frequent itemset* sebelumnya, *frequent (k-1)-itemsets*. Persamaan ini dapat dilihat pada persamaan 2.9.

$$Uk = \left(U^k_{1,}U^k_{2,\dots,}U^k_1\right)' = \left(C_{k,}C_{k,\dots,}C_{k,}\right)' \tag{2.9}$$

Kemudian sebuah matriks *W* dibentuk dengan persamaan 2.10.

$$W=(w_{ij})_{mxm}=U^kT_K{}' \tag{2.10}$$

Simbol:
W= matriks *W*
$U^{k=}$ matriks *U* dengan k indek
T= *Transaction Matrix (T*) dengan k indeks

Fungsi *F(W)* dibentuk dari matriks *W* untuk memberikan nilai pada element matriks $W(w_{ij})$ dengan persamaan 2.11.

$$F(w_{ij}) = \begin{cases} 1, w_{ij} \geq k \\ 0, w_{ij} < k' \end{cases} \text{ i,j } = 1,2,\dots,m \tag{2.11}$$

Dimana :
- Nilai 1 diberikan jika jumlah nilai $w_{ij}>= k$ dimana w_{ij} merupakan Absolute Support *(SupA)* dari kandidat *k-itemset* yang dibentuk *(C_k), k* adalah indek dan nilainya sesuai dengan *index k* pada *frequent k-itemset*.
- Nilai 0 diberikan jika jumlah w_{ij} tidak sama dengan *k*.

Matrix fungsi *F(W)* ditunjukkan pada persamaan dibawah 2.12.

$$F(W) = \begin{bmatrix} f\left[\sum_{i=t}^{n} u_{1i}t_{1i}\right] & \cdots & f\left[\sum_{i=1}^{n} u_{1i}t_{mi}\right] \\ \vdots & \ddots & \vdots \\ f\left[\sum_{i=1}^{n} u_{mi}t_{1i}\right] & \cdots & f\left[\sum_{i=1}^{n} u_{mi}t_{mi}\right] \end{bmatrix} \tag{2.12}$$

Setelah terbentuk matriks *W* kemudian dibentuk matriks vektor *v* untuk menyimpan nilai *Sup* dan *SupA* kandidat *k-item*set sesuai matrix *W*, ditunjukkan pada persamaan 2.13.

$$V=(v_i)_{mx1}=[\sum_{i=1}^{m} f(w_{1i}), \ldots, \sum_{i=1}^{m} f(W_{mi})] \quad (2.13)$$

Dalam kolom vektor *v*, setiap elemen v_i ($v_i > 0$) sama dengan *Absolute Support* dari kandidat k-*Item*set, $SupA(C_k)$ dan $Sup(C_k)$. Kandidat *item*set yang kurang dari Support minimal (Sup_{min}) kemudian dibuang, maka didapatkan *frequent k-itemset* (L_k). Proses terakhir membuang elemen baris t_{ij} di *T* yang kurang dari *k+1* ($\sum_{j=1}^{n} t_{ij} < k+1$).

Ketika *k=k+1*, jika jumlah anggota dari *frequent(k-1)-itemset* (L_{k-1}) kurang dari *k*, kandidat *k-item*sets yang dihasilkan oleh L_{k-1} akan jadi kosong atau *null* sehingga proses dari *frequent k-itemset*s akan berhenti, jika tidak maka proses berlajut ke pencarian *item*set selanjutnya mining *frequent (k+1)-itemset*.

2.7.2 Interestingness Measure

Piatetsky-Shapiro mengajukan tiga prinsip Interestingness Rule (Lenca, 2008):

1. Interestingness Rule adalah *independence*, jika hasilnya=0.
2. Interestingness Rule meningkat jika $\frac{n_a n_b}{n}$.
3. Interestingness Rule menurun ketika -$\frac{n_a n_b}{n}$.

Untuk membangkitkan *association rule* dari *frequent item*set yang telah terbentuk, digunakan Interestingness Measure dari *G. Piatetsky-Shapiro* untuk mendapatkan nilai Interestingenss dari kandidat *rule* (*R:A→B)* yang dibentuk dari *frequent k-itemset* (k>2) yang telah dibentuk dapat dilihat pada persamaan 2.14.

$$I=n_{ab} - \frac{n_a n_b}{n} \quad (2.14)$$

Dimana

I= Interestingness Rule

n_{ab}= jumlah kemunculan bersama Antecedent dan Consequent (SupA)

n_a= Absolute Support Antecedent, SupA (Antecedent)

$n_{b=}$ Absolute Support Consequent, SupA (Consequent)

2.7.3 Algoritma MIbARM

Algoritma yang digunakan untuk menerapkan semua metode yang telah dijelaskan sebelumnya adalah:

```
Input:
Database=D
Banyak Transaksi yang berbeda=m
Set of n distinct itemss=I
Minimum Support=Sup_min
Minimum Interestingness= I_min
Set
candidate itemset C_k = Φ
Frequent itemset L_k = Φ
Association rules set R= Φ
Transform database D to matrix T
Begin
Remove columns that ∑_{i=1}^{m} tij < m.supmin
Let C_1=I, L_1=Φ
For j=1; j ≤ n ; j++
    Calculate the Sup(I_j)
     If Sup(Ij) < Sup_min
       Delete I_j from C_1
     Endif
Set L_1=C_1,
number of rows: m_1
number of columns: n_1.
Remove rows that ∑_{j=1}^{n} tij < 2
Let C_2={I_iI_j|i,j=1,2,…,m_1}, L_2=Φ
Do{
    Construct Hermite matrix H
    If Sup(I_iI_j) < Sup_min
      Delete I_iI_j from C_2
    Endif
    }
While(Length of C_2 Doesn't change )
Let L_2=C_2
number of rows: m_2
number of coloumns: n_2
Remove rows that ∑_{j=1}^{n} tij < 3
Let C_k={c_1|1=1,2,…m_2},L_k=Φ
 do{
    do{
        Construct matrix W
        Calculate the Sup(C_k)
        If Sup(c_1) < Sup_min
```

```
42            Delete C1 from Ck
43          endif
44          }while (Length of Ck doesn't change)
45          Set Lk =Ck
46          number of rows: mk
47          number of columns: nk
48          Remove rows that ∑_{j=1}^{n} tij < k + 1
49   }While (Ck=Φ)
50  Generate rules Rc by L2,L3,…..,Lk
51   Do {
52          Calculate the I(ri)
53          If I(ri) >Imin
54              Add ri  into R
55          endif
56  }While(Rc ≠Φ)
57  Output R
58  end
```

Penjelasan Algoritma MIbARM:

- Baris 1,2,3,4,5,6 mendefiniskan notasi inputan:
- Baris 7,8,9,10 menetapkan Nilai C_k,L_k,R.
- Baris 11 adalah proses pertama *Transaction Matrix (T)* yaitu mengubah transaksi dalam *database D* ke matiks T.
- Baris 13,14,15,16,17,18,19,20,21,22,23 adalah proses Pencarian *Mining Frequent 1-Itemset*.
- Baris 24 sampai 34 adalah proses pencarian *Frequent 2-Itemset.*
- Baris 35-47 pencarian *Frequent itenset >2 k-Itemset*
- Baris 48-57 pembentukan *rule* berdasarkan *Frequent Itemset*.

2.8. Evaluasi Performa Rule menggunakan Lift Ratio

Dengan menggunakan nilai Lift Ratio, *rule* yang dihasilkan dapat diketahui tingkat kekuatan *rule*. Nilai Lift adalah rasio dari Confidence dan Confidence yang diharapkan. Confidence yang diharapkan didefinisikan sebagai Support dari produk dan nilai Support dari Antecedent dan Consequent *rule* dibagi dengan Support dari *Consequent* (Lenca, 2008).

Nilai Lift dari *rule* didefinisiakan sebagai berikut.

$$Lift = \frac{confidence}{expected\ Confidence} \quad (2.14)$$

$$expected\ Confidence = \frac{SupA(Consequent)}{m} = Sup(consequent) \quad (2.15)$$

$$Confidence = \frac{SupA(A,B)}{SupA(A)} \quad (2.16)$$

Dimana:

- SupA(Consequent)=Absolute Support atau kemunculan *item* di semua transaksi *rule* Consequent
- m= jumlah semua transaksi
- SupA(A,B)= jumlah kemunculan bersama *item* A dan B pada transaksi
- SupA(A)= jumlah kemunculan *item Antecedent* pada transaksi

Lift Ratio adalah tingkat *dependency* antara *rule* Antecedent dan *rule* Consequent. Nilai Lift berkisar antara 0 dan tidak terbatas, yang didefinisikan sebagai (Lenca, 2008):

- Nilai Lift yang lebih besar dari 1 menandakan bahwa kejadian pada *rule* Antecedent mempunyai efek positif pada kejadian di *rule* Consequent.
- Nilai Lift yang lebih kecil dari 1 menandakan bahwa kejadian pada *rule* Atencedent mempunyai efek negatif terhadap kejadian di *rule* Consequent.