

DATA MINING

Pertemuan 7: Association

Topik

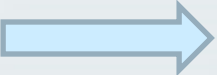
- Definisi Association Rule Mining
- Mining Single-Dimensional Boolean Association Rule
- Mining multilevel association rule
- Mining multidimensional association rule

Aturan Asosiasi (*Association Rule*)

- Mining Aturan Asosiasi:
 - Pencarian *frequent patterns* (pola yg sering muncul), asosiasi, korelasi di antara sekumpulan objek pada basis data transaksional, relasional, dan media penyimpanan informasi lainnya
- *Market Basket Analysis*:
 - Terdapat sejumlah transaksi yang masing-masingnya diwakili oleh sejumlah *item*. Permasalahan *Market Basket Analysis* adalah menganalisis kebiasaan pembelian dari pelanggan dengan menemukan **asosiasi antara item-item yang berbeda** yang dibeli oleh pelanggan di dalam keranjang belanja mereka.

Aturan Asosiasi (*Association Rule*)

- Bentuk rule:

Antecedent  Consequent

- Contoh Rule: {Donat, ...}  {Keripik kentang}

- **Keripik kentang sebagai consequent:**

- Dapat digunakan untuk menentukan apa yang harus dilakukan untuk meningkatkan penjualan keripik kentang

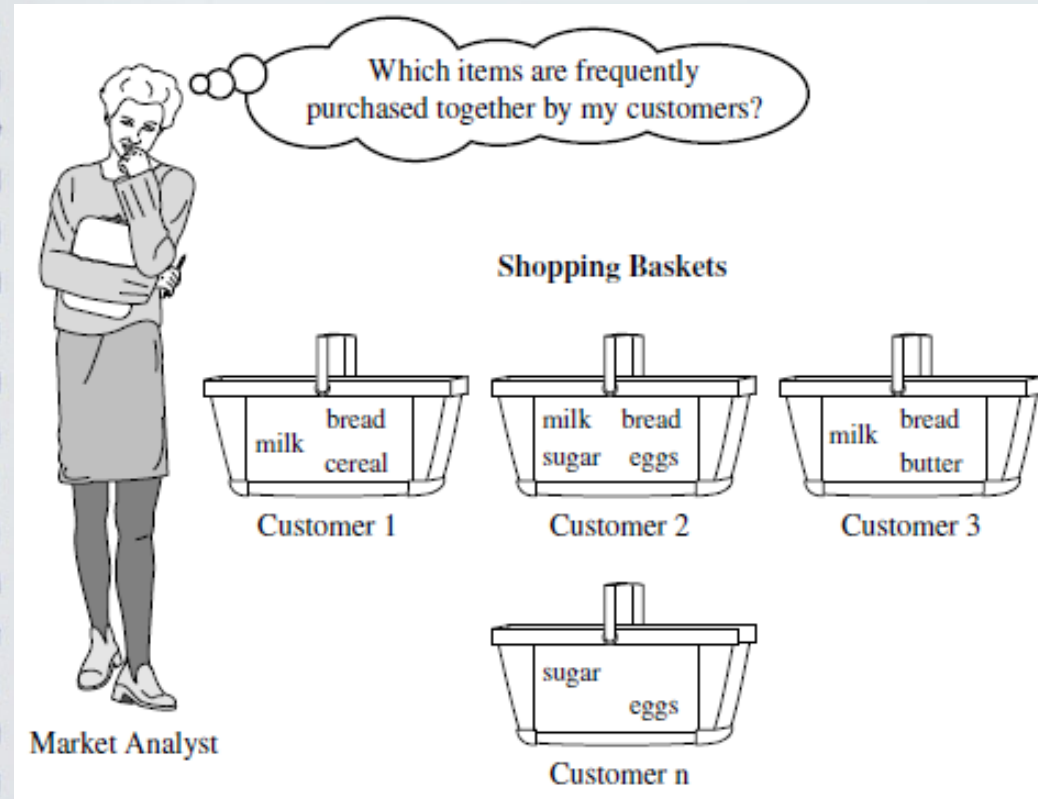
- **Donat sebagai antecedent:**


- Dapat digunakan untuk melihat produk apa yang akan terdampak jika toko berhenti menjual donat

- **Donat sebagai antecedent dan keripik kentang sebagai consequent:**

- Dapat digunakan untuk melihat produk apa yang harus dijual bersamaan dengan donat untuk meningkatkan penjualan keripik kentang

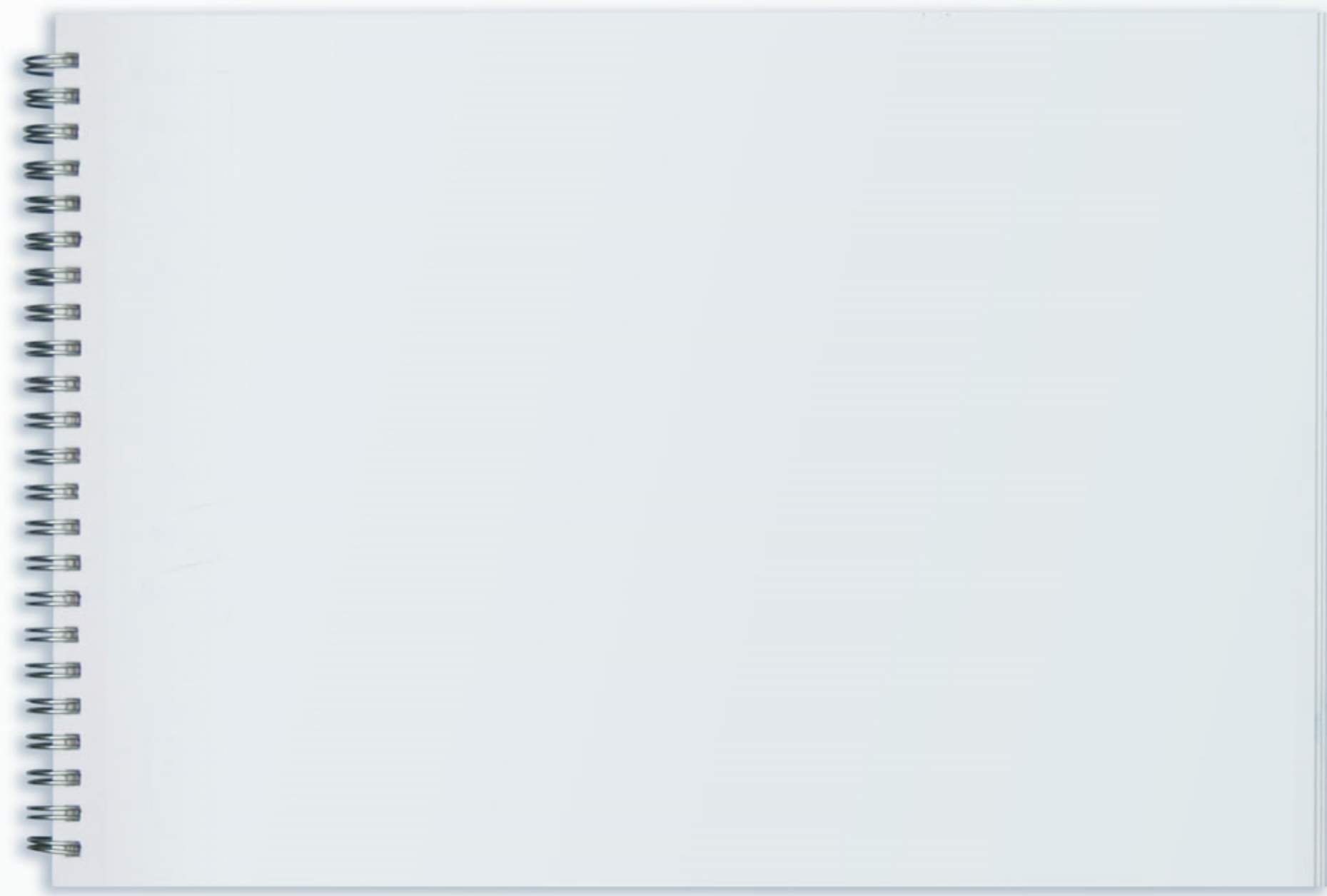
Contoh



- Adakah hal menarik yang dapat ditemukan?
 - Milk → bread (100%)
 - Sugar → eggs (100%)
 - Cereal → milk (100%)
 - Pelanggan yang membeli susu juga akan membeli roti
- 
- Identifikasi peluang potensial untuk *cross-selling* antara item yang saling berhubungan

Motivasi untuk Market Basket Analysis

- Jika pelanggan membeli susu, berapa besar kemungkinan mereka juga akan membeli roti?
- Aturan tersebut akan membantu penjual untuk:
 - Merencanakan ruang untuk lorong item: dengan menempatkan susu dekat dengan roti untuk meningkatkan penjualan
 - Merencanakan iklan produk/rekomendasi bagi pelanggan
 - Menggabungkan item yang sering dibeli bersamaan dengan memberikan diskon untuk meningkatkan penjualan



Pernyataan Permasalahan

- Input:

- Database dari transaksi
- Setiap transaksi terdiri dari sejumlah item yang dibeli oleh pelanggan dalam 1 kunjungan

- Output:

- Semua aturan/rule yang mengkorelasikan kemunculan sebuah item dengan item lainnya
- Contoh: 100% pelanggan yang membeli {susu} juga membeli {roti}

Data Transaksi

- Transaksi keranjang belanja:
 - $t1 = \{\text{roti, keju, susu}\}$
 - $t2 = \{\text{apel, telur, garam, yoghurt}\}$
 -
 - $t_n = \{\text{biskuit, susu, telur}\}$
- Konsep:
 - **Item**: sebuah item pada keranjang belanja
 - **I**: himpunan semua item yang dijual di toko
 - **Transaksi**: item-item yang dibeli dalam keranjang belanja, memiliki TID (transaction ID)
 - **Dataset transaksional**: Sekumpulan transaksi

Data transaksi: sekumpulan dokumen → Text Mining

- Data set sebuah dokumen teks. Setiap dokumen dianggap sebagai sebuah “keranjang” dari beberapa kata kunci (*keywords*)
 - doc1: Student, Teach, School
 - doc2: Student, School
 - doc3: Teach, School, City, Game
 - doc4: Basketball, Baseball
 - doc5: Basketball, Player, Spectator
 - doc6: Baseball, Coach, Game, Team
 - doc7: Basketball, Team, City, Game

Contoh Database Transaksi

Transaction Database	Items
100	1 2 3 4 5
200	6 2 3 5
300	1 2 4
400	2 9 4 5
500	1 2 6 4
600	6 3 5
...	...

Notasi Formal

- Sekumpulan item $\mathcal{I} = \{i_1, i_2, \dots, i_n\}$
- Database transaksi $\mathcal{D} = \{t_1, t_2, \dots, t_m\}$

$t_i \subseteq \mathcal{I}$ (sebuah transaksi merupakan sekumpulan item)



TID: $t_i \rightarrow$ sebuah nomor transaksi

- Sebuah transaksi t berisi X , sekumpulan item (**itemset**) pada \mathcal{I} , jika $X \subseteq t$.
 - Sebuah **itemset** adalah sekumpulan item
 - Contoh: $X = \{\text{susu}, \text{roti}, \text{sereal}\}$ adalah sebuah itemset
 - Sebuah **k -itemset** adalah itemset dengan item sebanyak k
 - Contoh: $\{\text{susu}, \text{roti}, \text{sereal}\}$ adalah 3-itemset

Notasi Formal (lanjutan)

- Sebuah aturan asosiasi adalah implikasi dalam bentuk:
 - $X \rightarrow Y$, dimana $X, Y \subseteq I$, dan $X \cap Y = \emptyset$
 - $X \rightarrow Y$

Mining Aturan Asosiasi

- Boolean Association Rule
 - Memperhatikan asosiasi antara kemunculan dan ketidakmunculan sejumlah item
 - Computer  Software_manajemen_keuangan
[support = 2%, confidence = 60%]
- Quantitative Association Rule
 - Nilai kuantitatif diskrit untuk item atau atribut dibagi ke dalam sejumlah interval
 - Age(X, "30...39") \wedge income(X, "42K...48K") 
buys(X, high resolution TV)
[support = 14%, confidence = 40%]

Mining Aturan Asosiasi (lanjutan)

- Single Dimensional Association Rule

- Hanya 1 dimensi (atribut) yang terlibat

$\text{buys}(\text{Computer}) \Rightarrow \text{buys}(\text{financial_management_software})$
[support = 2%, confidence = 60%]

- Multi-dimensional Association Rule

- Melibatkan 2 atau lebih dimensi (atribut)

$\text{age}(X, \text{"30 .. 39"}) \wedge \text{income}(X, \text{"42K .. 48K"}) \Rightarrow \text{buys}(X, \text{high resolution TV})$
[support = 14%, confidence = 40%]

- Single or Multi-dimensional Association Rule

$\text{age}(X, \text{"30 .. 39"}) \Rightarrow \text{buys}(X, \text{"laptop computer"})$
[support = 10%, confidence = 60%]
 $\text{age}(X, \text{"30 .. 39"}) \Rightarrow \text{buys}(X, \text{"computer"})$
[support = 15%, confidence = 80%]

Notasi Formal

- Rule Support:

- $\text{Support } (X \rightarrow Y) = P(X \cup Y)$
- $s\%$ dari transaksi pada \mathcal{D} mengandung $X \cup Y$

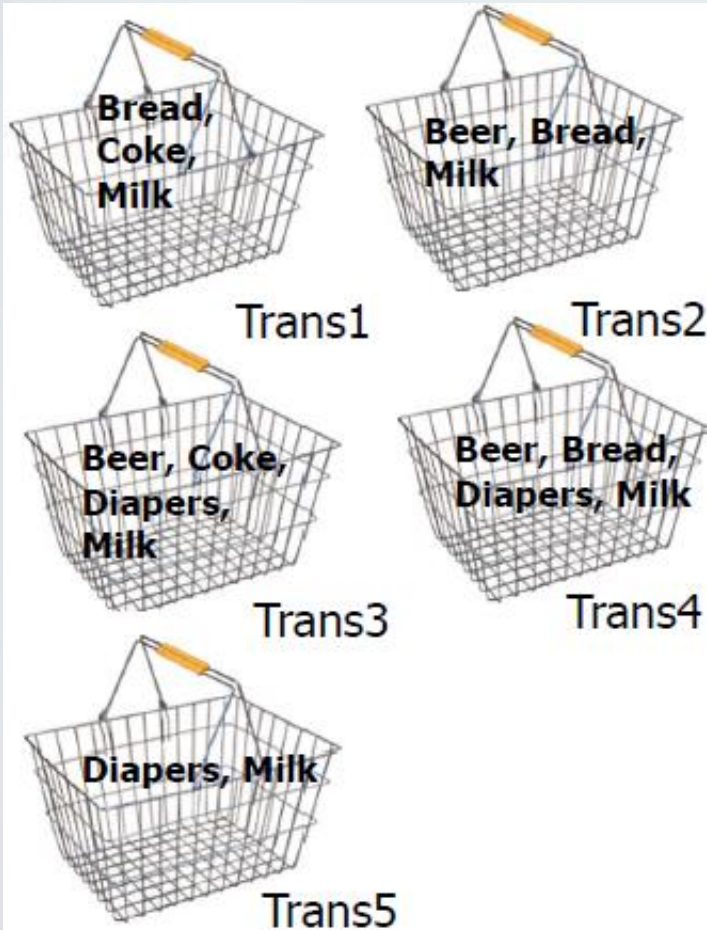
- Rule Confidence:

- $\text{Confidence } (X \rightarrow Y) = P(Y|X) = P(X \cup Y) / P(X)$
- Transaksi pada \mathcal{D} yang mengandung X memiliki $c\%$ yang mengandung Y

- Threshold (batasan) yang didefinisikan user):

- min_sup
- min_conf
- Aturan yang memenuhi min_sup dan min_conf disebut *strong*

Contoh



Bread → Milk
Diapers → Beer
Diapers → Milk

Support_count(Bread⇒Milk) = ?
Confidence (Bread⇒Milk) = ?

Support_count(Diapers⇒Beer) = ?
Confidence (Diapers⇒Beer) = ?

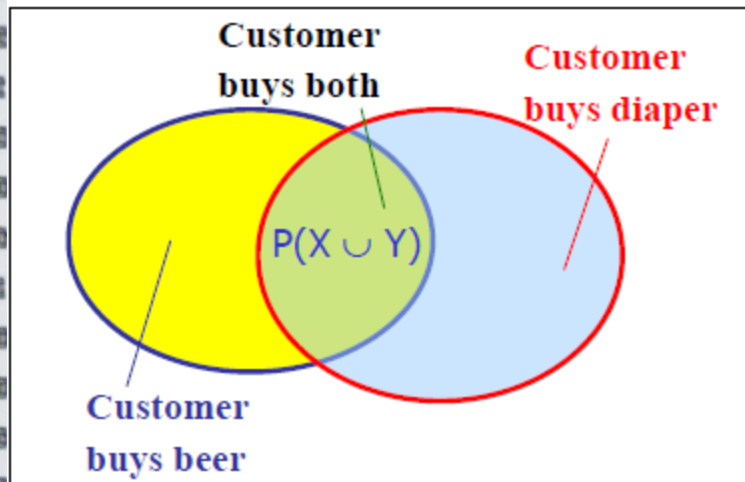
Pernyataan Permasalahan

- Cari semua itemset yang memiliki support transaction di atas *min_sup* (cari semua itemset besar)
- Gunakan itemset besar tersebut untuk menghasilkan aturan asosiasi

Besar = sering muncul:
 $\text{count} \geq \text{min_sup}$

Contoh

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F



- Itemset $X = \{x_1, \dots, x_k\}$
- Temukan semua rule $X \rightarrow Y$ dengan min confidence dan support
 - support, s, probabilitas sebuah transaksi berisi $X \cup Y$
 - confidence, c, probabilitas kondisional sebuah transaksi yang memiliki X juga memiliki Y
- Misal min_sup = 50%, min_conf = 50%:
 - $A \rightarrow C$ (50%, 66,7%)
 - $C \rightarrow A$ (50%, 100%)

support

confidence

Contoh (lanjutan)

Transaction-id	Items bought
10	A, B, C
20	A, C
30	A, D
40	B, E, F

Min support 50%
Min confidence 50%

Frequent pattern	Support
{A}	75%
{B}	50%
{C}	50%
{A, C}	50%

Untuk Rule $A \rightarrow C$:

Support = support ($\{A\} \cup \{C\}$) = 50%

Confidence = support ($\{A\} \cup \{C\}$) / support ($\{A\}$) = 66,67%

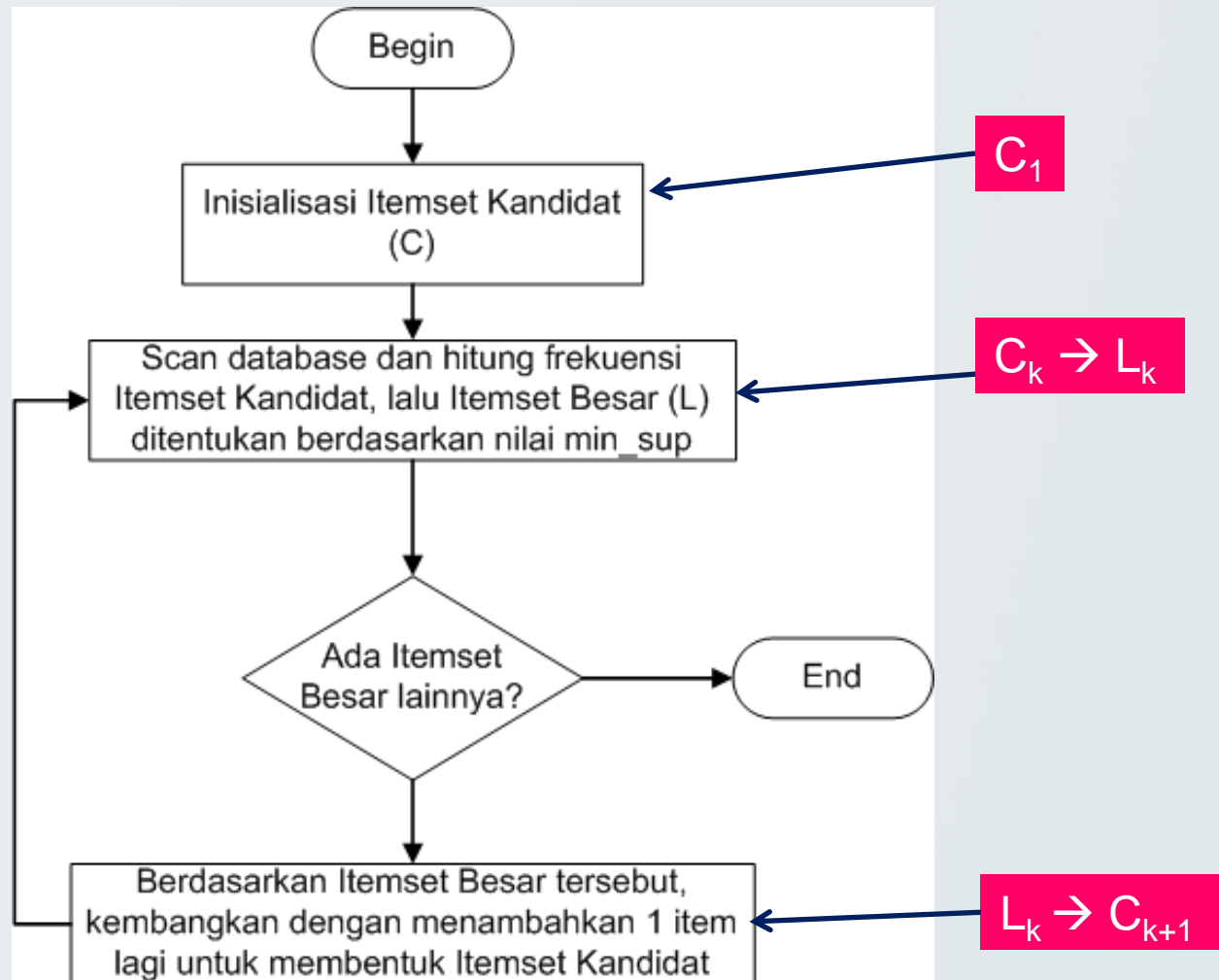
Algoritma yang Digunakan

- Algoritma Apriori
- Frequent Pattern Tree (FPT) Algorithm

1. Algoritma Apriori

- Pencarian itemset yang sering muncul dengan pembuatan itemset kandidat
 - $C_1 \rightarrow L_1 \rightarrow \dots \rightarrow C_k \rightarrow L_k \rightarrow C_{k+1} \rightarrow L_{k+1} \rightarrow \dots$
- Himpunan **kandidat** k-itemset. Notasi: **C**
 - C_k (itemset)
 - Contoh: $C_2 = \{\{\text{Roti, Susu}\}, \{\text{Popok, Bir}\}, \{\text{Popok, Susu}\}\}$
- Itemset besar (itemset yang sering muncul). Notasi: **L**
 - $L_k(\text{itemset, support_count})$
 - $\text{support_count} > \text{min_sup}$
 - Contoh $L_2 = \{\{\text{Roti, Susu}\}, \{\text{Popok, Susu}\}\}$ dimana $\text{min_sup} = 3$

Algoritma Apriori



Pseudocode Apriori

- 1) $L_1 = \{\text{large 1-itemsets}\}$
- 2) for ($k=2; L_{k-1} \neq \emptyset; k++$) do begin
- 3) $C_k = \text{apriori-gen}(L_{k-1});$ // New candidates
- 4) for each transactions $t \in \mathcal{D}$ do begin
- 5) $C_t = \text{subset}(C_k, t)$ // get the subsets of t that are candidates
- 6) for each candidate $c \in C_t$ do
- 7) $c.\text{count}++;$
- 8) end
- 9) $L_k = \{c \in C_k \mid c.\text{count} \geq \text{min_sup}\}$
- 10) end
- 11) $\text{Answer} = \bigcup_k L_k;$

Contoh Apriori



Database

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C₁

Itemset	Support Count
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L₁

Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3



C₂

Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

Min_Sup = 2

Contoh Apriori (lanjutan)

L_2	
Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

Min_Sup = 2

→

C_3	
Itemset	Support
{2 3 5}	2
{1 2 3}	1

L_3

Itemset	Support
{2 3 5}	2

- Perhatikan bahwa {1,2,3} **tidak** akan muncul di C_3 karena salah satu subsetnya {1,2} tidak ada dalam L_2

Permasalahan pada algoritma ini:

Ketika database di-scan untuk mengecek C_k yang akan membentuk L_k , banyak transaksi lain yang akan di-scan meskipun mereka tidak berisi k-itemset

Jumlah Itemset yang mungkin terbentuk

- Frequent pattern $\{a_1, \dots, a_{100}\} \rightarrow \sum_{k=1}^{100} \binom{100}{k} = 2^{100} - 1 = 1.27 \times 10^{30}$ frequent pattern!

Pembentukan Aturan Asosiasi dari Frequent Itemsets

- Buat aturan untuk Frequent Itemsets yang paling maksimum
 - Untuk setiap frequent itemset L , buat semua subset L yang tidak kosong
 - Untuk setiap subset S dari L yang tidak kosong, buat aturan: $S \rightarrow (L-S)$, lalu hitung nilai confidence: $c = \text{support_count}(L) / \text{support_count}(S)$
 - Jika $c \geq \text{min_conf}$, maka aturan tersebut adalah aturan asosiasi yang kuat (*strong association rule*)

Aturan Asosiasi yang dihasilkan

Database	
TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

C ₁	
Itemset	Support Count
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L ₁	
Itemset	Support
{1}	2
{2}	3
{3}	3
{5}	3

C ₂	
Itemset	Support
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

Min_Sup = 2

L ₂	
Itemset	Support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C ₃	
Itemset	Support
{2 3 5}	2
{1 2 3}	1

Min_Sup = 2

L ₃	
Itemset	Support
{2 3 5}	2

Dari L₂:

1 → 3: conf = 2/2 = 100% ✓

3 → 1: conf = 2/3 = 66.6%

2 → 3: conf = 2/3 = 66.6%

3 → 2: conf = 2/3 = 66.6%

2 → 5: conf = 3/3 = 100% ✓

5 → 2: conf = 3/3 = 100% ✓

3 → 5: conf = 2/3 = 66.6%

5 → 3: conf = 2/3 = 66.6%

Dari L₃:

{2,3} → 5: conf = 2/2 = 100% ✓

{2,5} → 3: conf = 2/3 = 66.6%

{3,5} → 2: conf = 2/2 = 100% ✓

2 → {3,5}: conf = 2/3 = 66.6%

3 → {2,5}: conf = 2/3 = 66.6%

5 → {2,3}: conf = 2/3 = 66.6%

Min sup = 50%

Min conf = 75%

Tugas

- Sebuah database memiliki 5 transaksi. Misal $\text{min_sup}=40\%$ dan $\text{min_conf}=75\%$

TID	Items
T100	{Roti, Keju, Jus}
T200	{Susu, Roti, Yoghurt}
T300	{Roti, Jus, Susu}
T400	{Telur, Roti, Keju, Jus}
T500	{Keju, Jus, Susu}

- Tentukan (tunjukkan langkah2 pengerjaannya):
 - Frequent Items (L_1, L_2, L_3, \dots)
 - Aturan asosiasi yang didapat dengan algoritma Apriori

Permasalahan pada Algoritma Apriori

- Biaya yang dikeluarkan untuk mengelola set kandidat besar sangat banyak.
- Misal, ada 10^4 1-itemset besar, algoritma Apriori akan menghasilkan 10^7 2-itemset kandidat. Untuk 100-itemset, perlu dihasilkan lebih dari $2^{100} \approx 10^{30}$ kandidat
- Pembuatan set kandidat merupakan biaya (besar) yang tidak dapat dilepaskan dari algoritma Apriori

Kesimpulan

- Penggunaan algoritma Apriori **tidak** tepat untuk melakukan mining pada database besar untuk mencari **pola/pattern yang panjang**

Meningkatkan Efisiensi Apriori

- Mengurangi jumlah transaksi
 - Transaksi yang tidak berisi *k-itemset* yang sering muncul bisa dihilangkan pada iterasi scan database berikutnya
- Sampling: melakukan mining pada sebagian data
 - Sampel yang diambil harus dapat disimpan di memori
 - Menggunakan threshold support yang lebih rendah untuk mengurangi kemungkinan hilangnya sejumlah itemset
 - Bagian data yang tidak di-sampling digunakan untuk menentukan banyaknya count itemset

Meningkatkan Efisiensi Apriori

- Partisi database
 - Database dibagi ke dalam n partisi yang tidak overlap dan di masing-masing partisi dilakukan pencarian frequent itemset lokal (menggunakan parallel algorithm)
 - Hasil digabung untuk membentuk frequent itemset global
 - Nilai support diturunkan untuk menyesuaikan dengan ukuran data
- Dynamic itemset counting
 - mempartisi database dan menggunakan estimasi nilai support dari itemset untuk dimasukkan ke itemset kandidat

2. Frequent Pattern Tree Algorithm (FP-Growth)

- Item pada transaksi diurutkan berdasarkan **frekuensi** di database (secara **menurun/descending**)
- Struktur tree (Prefix-tree) digunakan untuk mencatat pola yang sering muncul secara *top-down*. Hanya **item yang memiliki frekuensi besar yang akan memiliki node di dalam tree** (dengan kata lain, database dikompresi)
- Setiap pengaksesan pada tree adalah untuk mendapatkan pola yang sering muncul yang berasosiasi dengan item dan prosedurnya **rekursif**.

Algoritma FP Tree

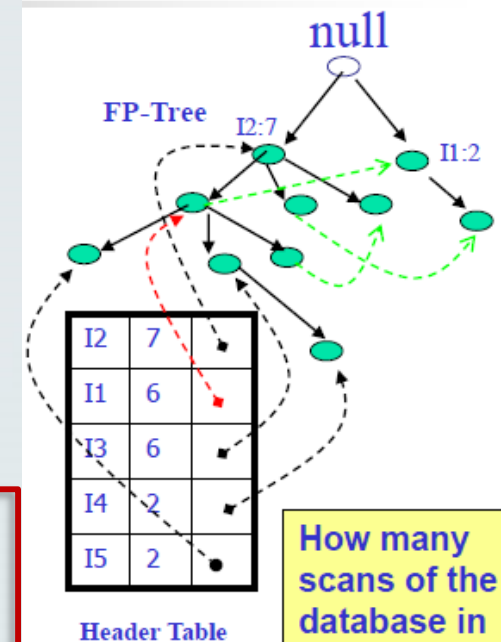
- Pembuatan FP Tree:

Inisialisasi frequent itemset sebagai sebuah item tunggal pada database. Urutkan 1-itemset secara menurun

Buat root dari FP-Tree dan beri label "null"

Buat tree dengan men-scan transaksi pada database menurut urutan 1-itemset. Buat cabang pada tree jika tidak ada prefix yang sama pada jalur tree. Proses penghitungan dilakukan untuk item pada transaksi sepanjang jalur pada tree

Sebuah tabel **item header** digunakan untuk menghubungkan semua node pada list



How many scans of the database in order to build the FP-Tree?

Contoh

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

- min_support = 3

1. Scan database 1x, cari *frequent* 1-itemset (*single item pattern*)
2. Urutkan frequent item secara menurun, f-list
3. Scan database sekali lagi, buat FP-tree

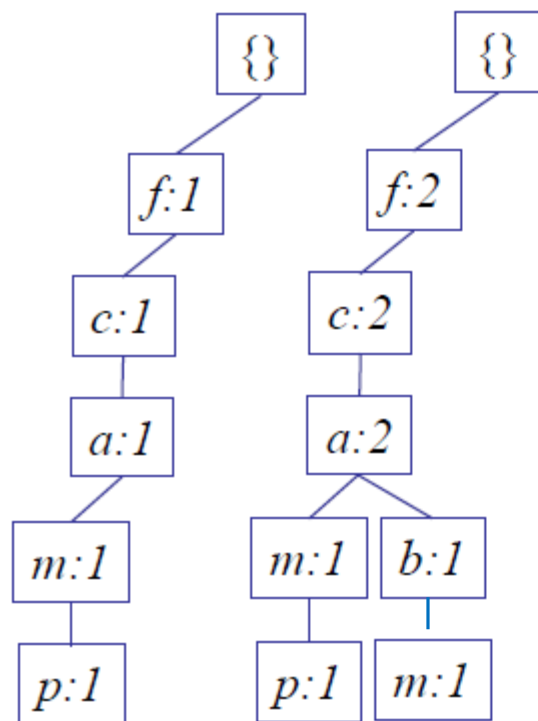
Header Table

Item	Frequency Head
f	4
c	4
a	3
b	3
m	3
p	3

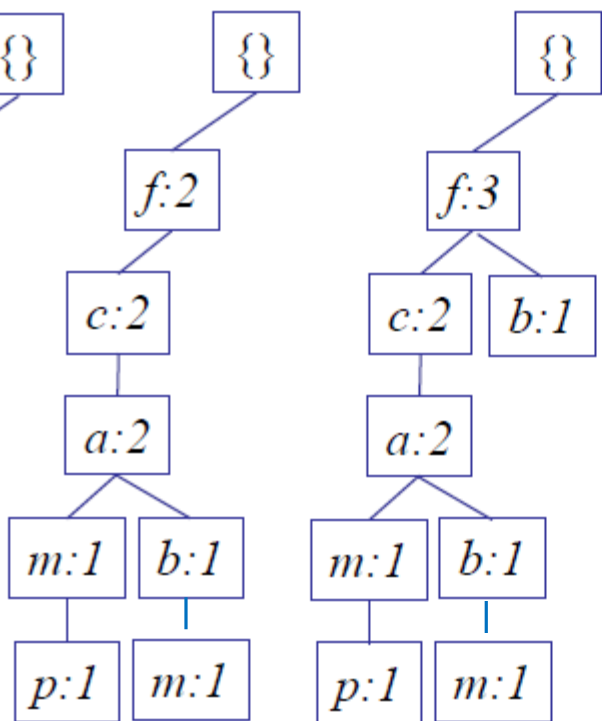
F-list = f-c-a-b-m-p

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

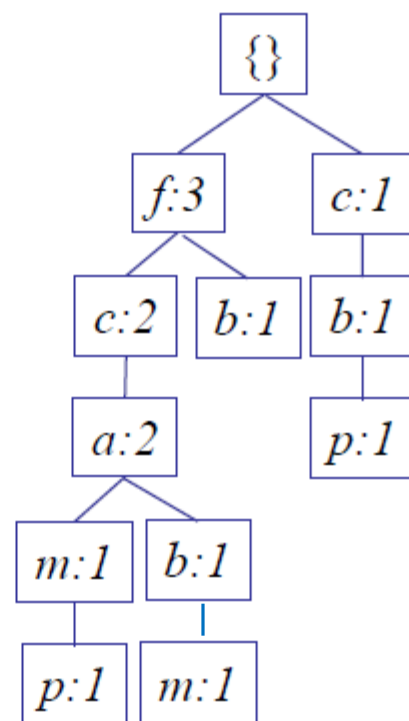
- min_support = 3



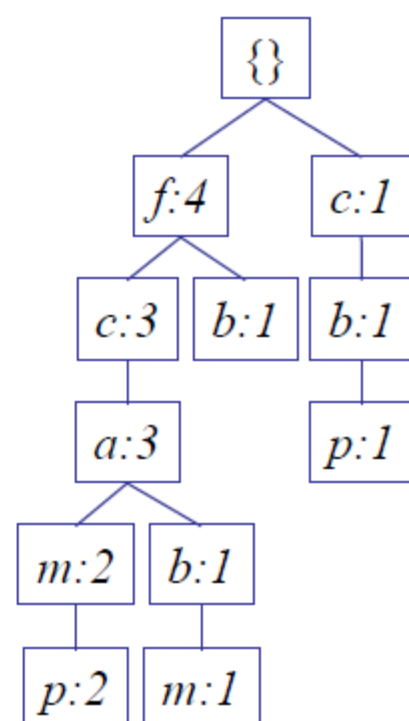
step1



step3



step4



step5

Algoritma FP-Tree (lanjutan)

- Melakukan mining pada frequent pattern dari FP-tree:

Mulai dari item terakhir dari header table sebagai pola akhiran, buat basis conditional pattern. Sepanjang jalur dari prefix, semua akhiran item akan memiliki nilai support yang sama

Berdasarkan basis conditional pattern, buat conditional FP-tree dari item, dan lakukan secara rekursif

Pertumbuhan dari pattern dicapai dengan menggabungkan pattern akhiran dengan frequent pattern yang dihasilkan oleh FP-Tree

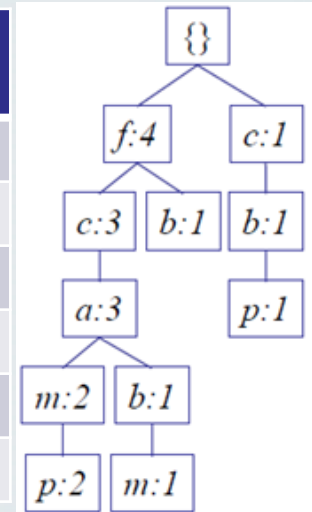
Algoritma berhenti ketika penelusuran header table selesai. Semua frequent pattern (itemset besar) sudah dibuat pada saat pertumbuhan pattern

Contoh 1

Frequent Item Header Table

Item	Frequency Head
f	4
c	4
a	3
b	3
m	3
p	3

FP-Tree



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

- min_support = 3

- Mulai dari frequent item header table pada FP-Tree
- Telusuri FP-Tree dengan mengikuti jalur untuk setiap frequent item p (dengan kata lain, buat rangkaian dari semua jalur yang berakhiran di item p)
- Akumulasi semua jalur prefix dari item p untuk membentuk conditional pattern base dari p

Item	Conditional Pattern Base	Frequent Pattern yang dihasilkan
c	f:3	fc: 3
a	c:3, f:3, fc:3	ca:3, fa:3, fca:3
b	f:2, c:2, a:1, ca:1, fc:1, fa:1, fca:1	-
m	a:3, c:3, f:3, ca:3, fc:3, fa:3, fca:3, b:1, ab:1, cab:1, fcab:1	am:3, cm:3, fm:3, fcm:3, fam:3, cam:3, fcam:3
p	m:2, a:2, c:3, f:2, am:2, cam:2, fcam:2, b:1	cp:3

Contoh 1 (lanjutan)

- Jika diketahui $\text{min_conf}=75\%$, aturan asosiasi yang mana saja yang dapat dibentuk?

Keunggulan Struktur FP-Tree

✓ Kelengkapan

- Mempertahankan informasi yang lengkap untuk melakukan mining pada frequent pattern
- Tidak pernah memutus sebuah pattern yang panjang

✓ Keringkasan

- Mengurangi info yang tidak relevan → item yang jarang muncul akan dihilangkan
- Frekuensi kemunculan item diurut menurun → semakin sering sebuah item muncul, semakin mungkin item tersebut masuk dalam frequent item bersama item lain
- Ukuran tidak akan pernah lebih besar dari database transaksi

Keunggulan Struktur FP-Tree (lanjutan)

✓ Menggunakan metode *divide-and-conquer*

- Mendekomposisi pekerjaan mining dan Database berdasarkan frequent pattern yang telah didapat
- Mengarah pada pencarian terfokus pada database yang lebih kecil

✓ Faktor lain

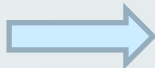
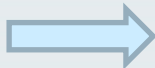
- Tidak ada pembuatan set kandidat, tidak ada pengetesan set kandidat
- Database yang terkompresi → struktur FP-Tree
- Tidak ada scan berulang-ulang pada database
- Operasi dasar hanyalah menghitung frekuensi item dan membuat FP-tree. Tidak ada pencarian dan pencocokan pattern

Pengukuran *Interestingness*

- SUBYEKTIF [Silberschatz & Tuzhilin, KDD95]
 - Sebuah rule dianggap menarik jika:
 - Di luar dugaan (*unexpected*); dan/atau
 - Dapat dilakukan sesuatu terhadapnya (*actionable*)
 - Hanya user yang dapat menilai tingkat *interestingness* dari sebuah rule
- OBYEKTIF
 - Support
 - Confidence/strength
 - Lift/Interest/Correlation
 - Conviction
 - Leverage/Piatetsky-Shapiro
 - Coverage

Kritik terhadap Support dan Confidence

	basketball	not basketball	sum(row)	
cereal	2000	1750	3750	75%
not cereal	1000	250	1250	25%
sum(col.)	3000	2000	5000	
	60%	40%		

- Dari 5000 siswa:
 - 3000 main basket
 - 3750 makan sereal
 - 2000 main basket dan makan sereal
- Rule: main basket  makan sereal [40%, 66.7%]
 - Menyesatkan, karena persentase keseluruhan yang makan sereal adalah 75%, yang lebih besar dari 66.7%
- Rule: main basket  tidak makan sereal [20%, 33.3%]
 - Lebih akurat, meskipun dengan nilai support dan confidence yang rendah

Lift dari sebuah Rule

Rasio *confidence* terhadap *expected confidence*

- Lift (Correlation, Interest)

$$Lift(A \rightarrow B) = \frac{\text{confidence}(A \rightarrow B)}{\text{support}(B)}$$

- A dan B berkorelasi negatif jika nilai Lift kurang dari 1; dan sebaliknya berkorelasi positif jika nilai Lift lebih dari 1
- Jika $Lift(A \rightarrow B) > 1$, maka artinya kedua kejadian tersebut saling bergantung (*dependant*) dan dapat digunakan untuk memprediksi konsekuensi (akibat) pada dataset selanjutnya
- Jika $Lift(A \rightarrow B) = 1$, maka kedua kejadian *independent* dan tidak ada rule yang dapat dibuat untuk kejadian tersebut
- Jika $Lift(A \rightarrow B) < 1$, artinya kedua kejadian saling bergantung secara negatif, di mana kejadian B memberikan efek negatif terhadap kejadian A

Lift

	basketball	not basketball	sum(row)	
cereal	2000	1750	3750	75%
not cereal	1000	250	1250	25%
sum(col.)	3000	2000	5000	
	60%	40%		

- Dalam contoh di atas, perhitungan Lift dari rule main basket (B) → makan cereal (C):

$$Support(C) = \frac{3750}{5000} = 0,75$$

$$Confidence(B \rightarrow C) = P(C|B) = \frac{P(C \cup B)}{P(B)} = \frac{2000}{3000} = 0,67$$

$$Lift(B \rightarrow C) = \frac{confidence(B \rightarrow C)}{support(C)} = \frac{0,67}{0,75} = 0,89$$

- Karena nilai $Lift(B \rightarrow C) < 1$, maka B berkorelasi negatif dengan C. Dengan kata lain, kejadian B memberikan efek negatif terhadap kejadian C, yakni jika Main Basket maka tidak makan Cereal adalah Rule yang lebih tepat

Riset Terkini dalam Association Rule Mining

- Penjelasan dan interpretasi
 - Hubungan sebab-akibat, penemuan item-item yang memiliki tingkat ketergantungan tinggi
- Menentukan minimum support dan minimum confidence secara otomatis
- Mining pada data streaming
- Mining aturan negatif
 - $A \rightarrow \neg B, \neg A \rightarrow B, \neg A \rightarrow \neg B$

Penilaian

- NTS:
 - 40% UTS
 - 50% tugas
 - 10% kehadiran
- NAS:
 - 30% tugas
 - 60% Final Project
 - 10% kehadiran

Catatan: Untuk dapat lulus MK ini, semua komponen di atas tidak boleh kosong

$$\text{Nilai Akhir} = (\text{NTS} + \text{NAS}) / 2$$