Machine Learning

Lecture 20: Association Rule Mining

COURSE CODE: CSE451

2021

Course Teacher

Dr. Mrinal Kanti Baowaly

Associate Professor

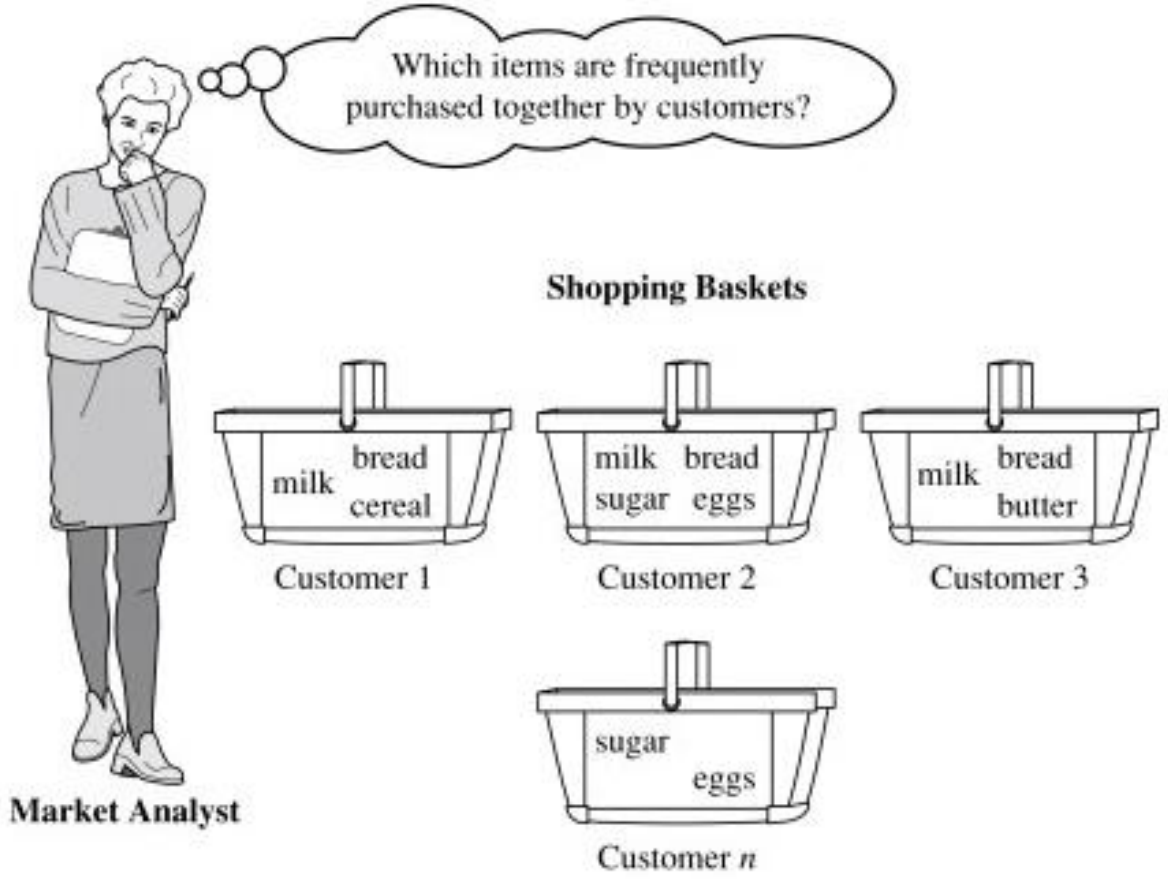
Department of Computer Science and Engineering, Bangabandhu Sheikh Mujibur Rahman Science and Technology University, Bangladesh.

Email: baowaly@gmail.com

Association Rule Mining

- Association Rule Mining is a type of unsupervised learning to find some interesting relations or associations among the variables of dataset.
- It finds:
  - features (variables) which occur together
  - features (variables) which are “correlated”
- Initially used for Market Basket Analysis to discover the associations between items purchased by customers.

Market Basket Analysis



Market Basket Analysis (cont.)

- It is a technique used by the big retailers to discover the associations between items purchased by customers.
- It works by looking for combinations of items that occur together frequently in transactions.
- The retailers keep the frequent items together in the store in order to increase sales.

Applications of Association Rule Mining

- **Market Basket Analysis:** It is one of the popular examples and applications of association rule mining. This technique is commonly used by big retailers to determine the association between items.
- **Medical Diagnosis:** With the help of association rules, patients can be cured easily, as it helps in identifying the probability of illness for a particular disease.
- **Protein Sequence:** The association rules help in determining the synthesis of artificial Proteins.
- It is also used for the **Catalog Design** and **Loss-leader Analysis** and many more other applications.

Definition of Association Rule Mining

- Given a set of transactions, find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Market-Basket transactions

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Association Rules

$\{Bread\} \rightarrow \{Milk\}$

$\{Diaper\} \rightarrow \{Beer\}$

$\{Bread, Milk\} \rightarrow \{Diaper\}$

Definition: Frequent Itemset

Itemset
- A collection of one or more items
  - Example: {Milk, Bread, Diaper}
- k-itemset
  - An itemset that contains k items

Support count (σ)
- Frequency of occurrence of an itemset
- E.g. σ({Milk, Bread,Diaper}) = 2

Support
- Fraction of transactions that contain an itemset
- E.g. s({Milk, Bread, Diaper}) = 2/5

Frequent Itemset
- An itemset whose support is greater than or equal to a minimum support (*minsup)* threshold

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Definition: Association Rule

Let $I = \{i_1, i_2, i_3, \ldots, i_n\}$, the set of all items

Let $D = \{t_1, t_2, t_3, \ldots, t_n\}$, a set of transactions called the database, $T \subseteq D$.

- **Association rule** is defined as an implication of the form: $X \rightarrow Y$, with respect to T where $X,Y \subseteq I$ and $X \cap Y = \varnothing$
  - Example: {Milk, Diaper} $\rightarrow$ {Beer}
    This means that if a customer buys milk and diapers, then he will high likely buy a beer.
- Association rule learning works on the concept of If and Else statement, such as if X then Y.
- Here the If element (X) is called **antecedent**, and then statement (Y) is called as **consequent**.

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Association Rule Evaluation Metrics

- Support (s)
  - Fraction of transactions that contain both X and Y.

$$Support\ (X \rightarrow Y) = \frac{Transactions\ involving\ both\ X\ and\ Y}{Total\ transactions} = \frac{\sigma(X,Y)}{|T|}$$

- Confidence (c)
  - Measures how often items in Y appear in transactions that contain X.

$$Confidence\ (X \rightarrow Y) = \frac{Transactions\ involving\ both\ X\ and\ Y}{Transactions\ involving\ only\ X} = \frac{\sigma(X,Y)}{\sigma(X)} = \frac{Support\ (X \rightarrow Y)}{Support\ (X)}$$

TID	*Items*
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example:

$$\{\text{Milk}, \text{Diaper}\} \Rightarrow \text{Beer}$$

$$s = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{|\text{T}|} = \frac{2}{5} = 0.4$$

$$c = \frac{\sigma(\text{Milk}, \text{Diaper}, \text{Beer})}{\sigma(\text{Milk}, \text{Diaper})} = \frac{2}{3} = 0.67$$

Lift

- It is the strength of any rule, which can be defined as below formula:

$$Lift(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{Support\ (Y)} = \frac{Support\ (X \rightarrow Y)}{Support\ (X) \times Support\ (Y)}$$

- It is the ratio of the observed support measure and expected support if X and Y are independent of each other. It has three possible values:
  - If Lift=1: The probability of occurrence of antecedent and consequent is independent of each other, no rule can be drawn involving those two itemsets.
  - Lift>1: It determines the degree to which the two itemsets are dependent to each other, and makes those rules potentially useful for predicting the consequent in future data sets.
  - Lift<1: It tells us that one item is a substitute for other items. This means that presence of one itemset has negative effect on presence of other itemset and vice versa.

Association Rule Mining Task

- Given a transaction data set T, and a minimum support (*minsup)* and a minimum confidence (*minconf)*. The goal of association rule mining is to find all rules having
  - support ≥ *minsup* threshold
  - confidence ≥ *minconf* threshold

- Brute-force approach:
  - List all possible association rules
  - Compute the support and confidence for each rule
  - Prune rules that fail the *minsup* and *minconf* thresholds

$\Rightarrow$ Computationally prohibitive!

Brute-force approach: An Example

Given:

1. minimum support 2/4 or 0.5,
2. minimum confidence 2/3 or 0.67

Association rules:

- {b} → {e}, support = 75%, confidence = 100%
- {e} → {b}, support = 75%, confidence = 100%
- {a} → {c}, support = 50%, confidence = 100%
- {c} → {a}, support = 50%, confidence = 66%
- {b, c} → {e}, support = 50%, confidence = 100%
- {e} → {b, c}, support = 50%, confidence = 66%
- {c, e} → {b}, support = 50%, confidence = 100%
- {b, e} → {c}, support = 50%, confidence = 66%
- …

TID	*Items*
1	a, c, d
2	b, c, e
3	a, b, c, e
4	b, e

How many association rules are generated ?

Computationally expensive!!

Observations: Mining Association Rules

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Example of Rules:

{Milk,Diaper} → {Beer} (s=0.4, c=0.67)
{Milk,Beer} → {Diaper} (s=0.4, c=1.0)
{Diaper,Beer} → {Milk} (s=0.4, c=0.67)
{Beer} → {Milk,Diaper} (s=0.4, c=0.67)
{Diaper} → {Milk,Beer} (s=0.4, c=0.5)
{Milk} → {Diaper,Beer} (s=0.4, c=0.5)

Observations:

- All the above rules are binary partitions of the same itemset:
  {Milk, Diaper, Beer}
- Rules originating from the same itemset have identical support but can have different confidence
- Thus, we may separate the support and confidence requirements

Mining Association Rules

Two-step approach:

1. Frequent Itemset Generation
   - Generate all itemsets whose support $\geq$ minsup

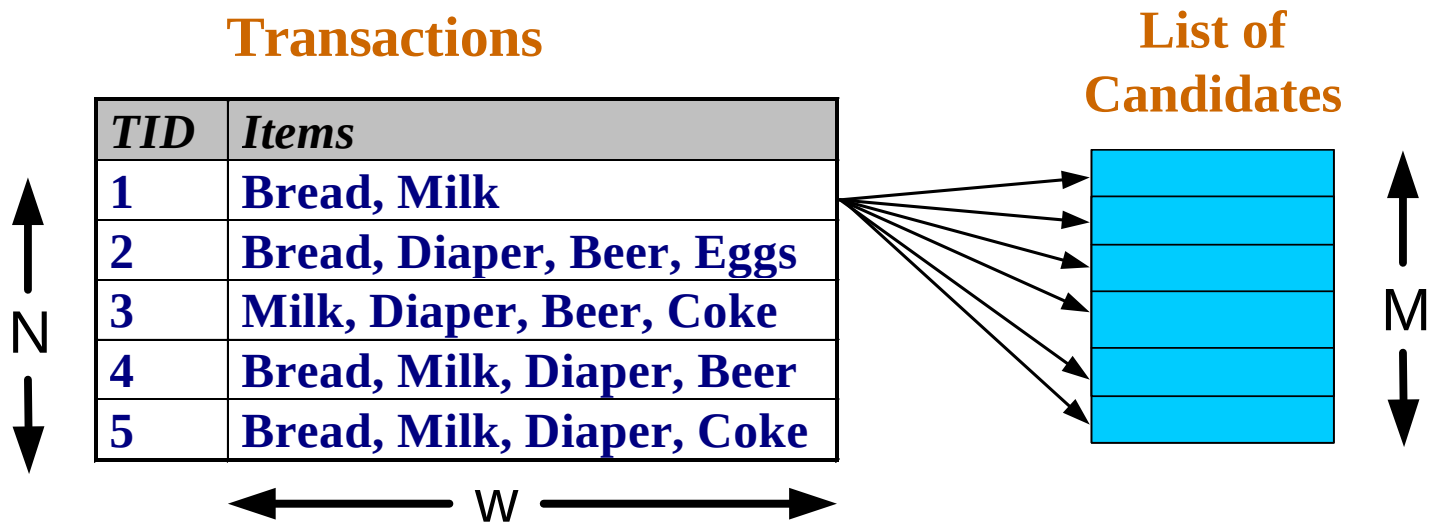
2. Rule Generation
   - Generate high confidence rules from each frequent itemset, where each rule is a binary partitioning of a frequent itemset

Frequent itemset generation is still computationally expensive

Frequent Itemset Generation

Brute-force approach:

- Each itemset in the lattice is a candidate frequent itemset
- Count the support of each candidate by scanning the database



- Match each transaction against every candidate
- Complexity ~ O(NMw) => Expensive since $M = 2^d$!!! (d = No. of unique items)

Computational Complexity

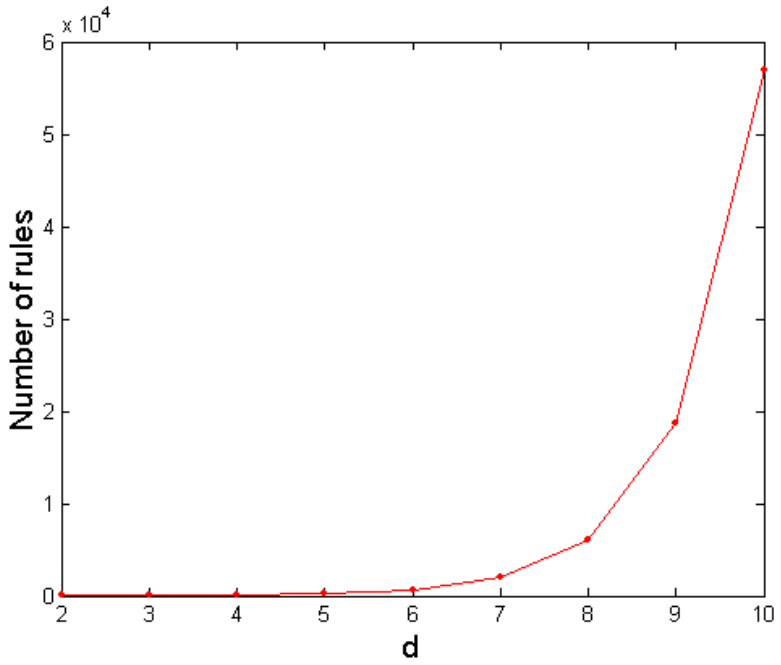
Given d unique items:

- Total number of itemsets = 2^d
- Total number of possible association rules:

$$R = \sum_{k=1}^{d-1}\left[\binom{d}{k} \times \sum_{j=1}^{d-k}\binom{d-k}{j}\right]$$

$$= 3^d - 2^{d+1} + 1$$

If d=6, R = 602 rules



Frequent Itemset Generation Strategies

Reduce the number of candidates (M)

- Complete search: $M=2^d$
- Use pruning techniques to reduce M

Reduce the number of transactions (N)

- Reduce size of N as the size of itemset increases
- Used by DHP and vertical-based mining algorithms

Reduce the number of comparisons (NM)

- Use efficient data structures to store the candidates or transactions
- No need to match every candidate against every transaction

Reducing Number of Candidates

- Apriori principle:
  - If an itemset is frequent, then all of its subsets must also be frequent

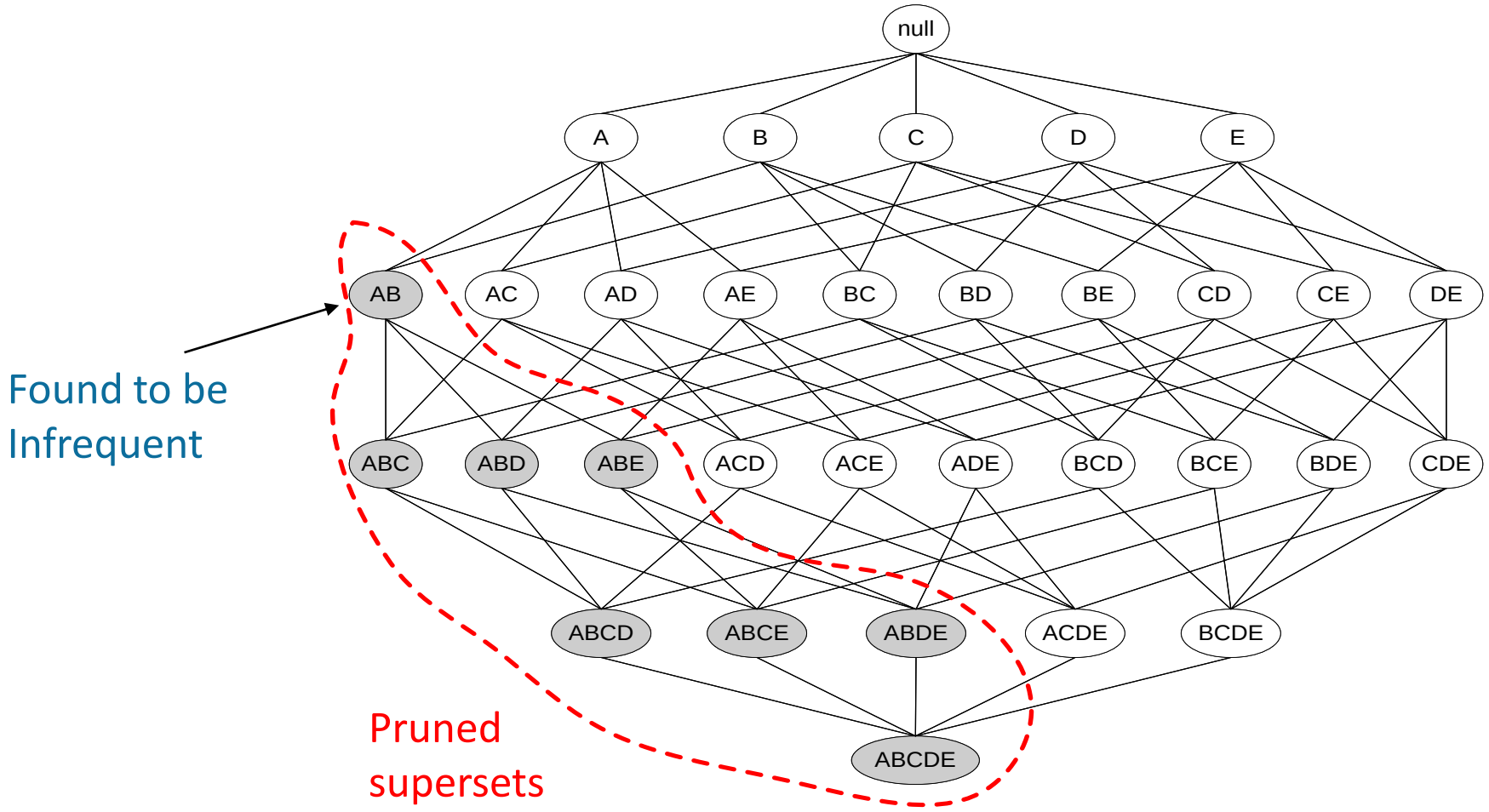
- Apriori principle holds due to the following property of the support measure:

$$\forall X, Y : (X \subseteq Y) \Rightarrow s(X) \geq s(Y)$$

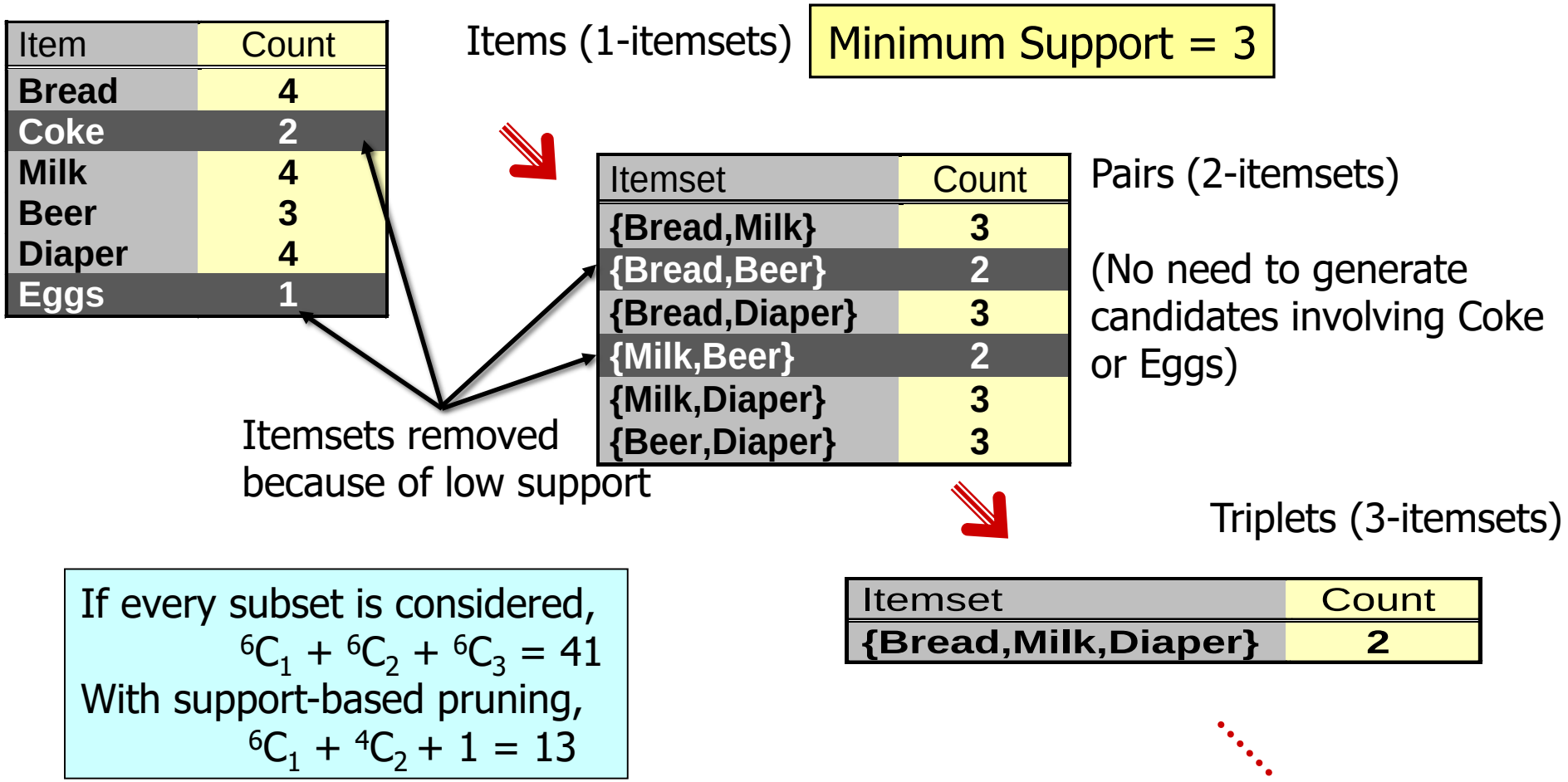
  - Support of an itemset never exceeds the support of its subsets
  - This is known as the anti-monotone property of support
  - All the supersets of an infrequent itemset must also be infrequent

    e.g. {d} is not frequent, {a, d} is not frequent either.

Illustrating Apriori Principle



Illustrating Apriori Principle



Candidate itemsets generation

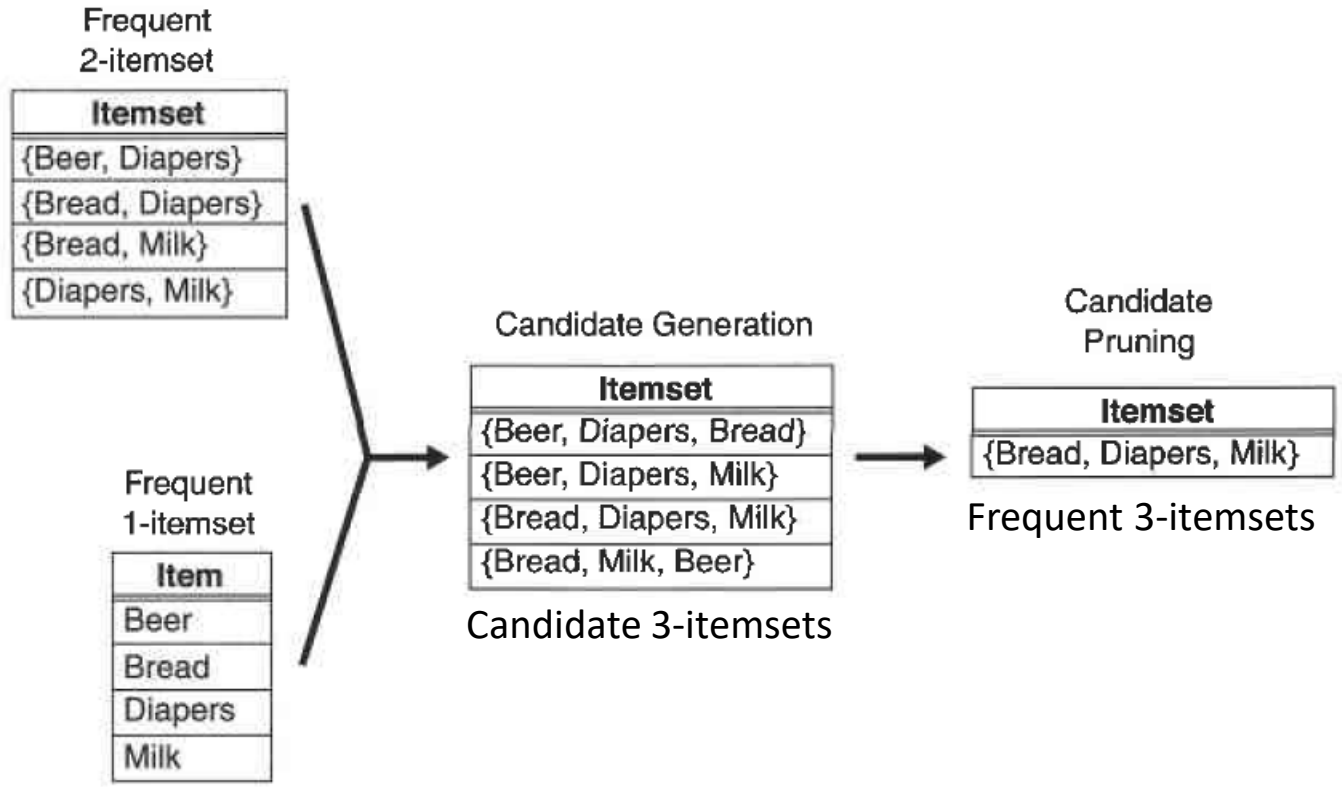
Brute-Force Method:

- The number of candidate itemsets generated at level k is equal to ${}^{d}C_{k}$, where d is the total number of items.
- The brute-force method considers every k-itemset as a potential candidate and then applies the candidate pruning step to remove any unnecessary candidates.
- Example: there are ${}^{6}C_{3}$ = 2O candidate 3-itemsets that can be formed using the six items given in the previous example.

Candidate itemsets generation (cont.)

F_{k-1} x F_1 Method:

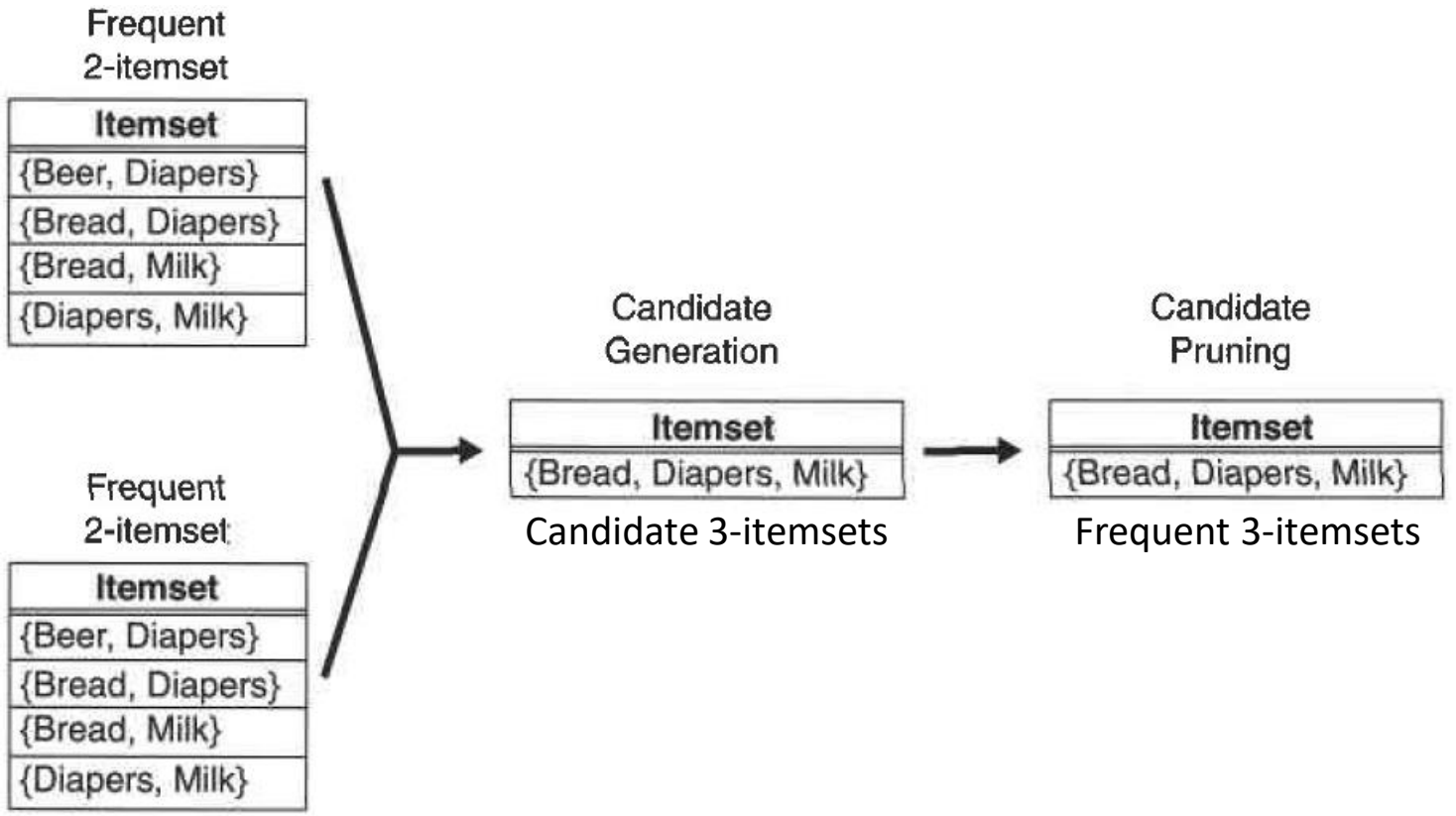
- Every candidate k-itemsets is composed of a frequent (k - 1)-itemset and a frequent 1-itemset. Items must be lexicographically ordered.



Candidate itemsets generation (cont.)

F_{k-1} x F_{k-1} Method:

- Merge two frequent (k-1)-itemsets if their first (k-2) items are identical, items must be lexicographically ordered.



Candidate itemsets generation (cont.)

F_{k-1} x F_{k-1} Method (Another Example):

- Consider K=4.

  F_3 = {ABC, ABD, ABE, ACD, BCD, BDE}

  $C_4 = F_3$ x F_3

  Merge(ABC, ABD) = ABCD

  Merge(ABC, ABE) = ABCE

  Merge(ABD, ABE) = ABDE

  Do not merge(ABD, ACD) because they share only prefix of length 1 instead of length 2

  Candidate 4-itemsets C_4 = {ABCD, ABCE, ABDE}

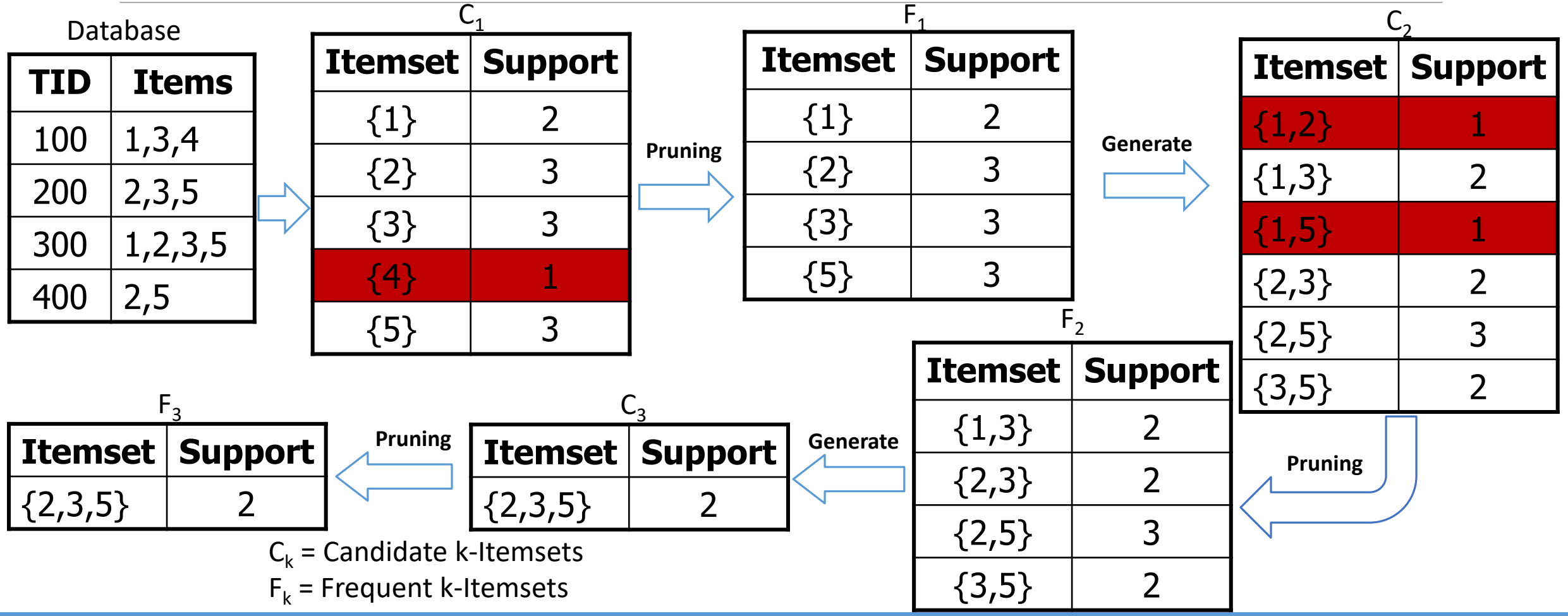
Source

Apriori algorithm : generate frequent itemsets

Method:

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are identified
  - Generate length (k+1) candidate itemsets from length k frequent itemsets, i.e., $C_{k+1}=F_k \times F_k$
  - Prune candidate itemsets containing subsets of length k that are infrequent
  - Count the support of each candidate by scanning the DB
  - Eliminate candidates that are infrequent, leaving only those that are frequent

Apriori Algorithm : Generate Frequent Itemsets – An Example



Rule generation from frequent itemsets

- From the previous example, we find the frequent itemsets using the Apriori algorithm in F_1, F_2, and F_3 :

  {1}, {2}, {3}, {5}, {1,3}, {1,5}, {2,3}, {2,5}, {3,5} and {2,3,5}

- Rule generation (*minconf=0.8*):

  $\{1\} \rightarrow \{3\}$, c=2/2=1
  $\{2\} \rightarrow \{3\}$, c=2/3=0.67
  $\{2\} \rightarrow \{5\}$, c=3/3=1
  $\{3\} \rightarrow \{5\}$, c=2/3=0.67

  $\{3\} \rightarrow \{1\}$, c=2/3=0.67
  $\{3\} \rightarrow \{2\}$, c=2/3=0.67
  $\{5\} \rightarrow \{2\}$, c=3/3=1
  $\{5\} \rightarrow \{3\}$, c=2/3=0.67

  $\{2,3\} \rightarrow \{5\}$, c=2/2=1
  $\{2,5\} \rightarrow \{3\}$, c=2/3=0.67
  $\{3,5\} \rightarrow \{2\}$, c=2/2=1
  $\{2\} \rightarrow \{3,5\}$, c=2/3=0.67
  $\{3\} \rightarrow \{2,5\}$, c=2/3=0.67
  $\{5\} \rightarrow \{2,3\}$, c=2/3=0.67

- As the given minimum confidence is 0.8, so the five rules: $\{1\} \rightarrow \{3\}$, $\{2\} \rightarrow \{5\}$, $\{5\} \rightarrow \{2\}$, $\{2,3\} \rightarrow \{5\}$, $\{3,5\} \rightarrow \{2\}$ can be considered as the strong association rules for the given problem.

Apriori Algorithm: Another Example

- Suppose we have the following dataset that has various transactions, and from this dataset, we need to find the frequent itemsets and generate the association rules using the Apriori algorithm

TID	ITEMSETS
T1	A, B
T2	B, D
T3	B, C
T4	A, B, D
T5	A, C
T6	B, C
T7	A, C
T8	A, B, C, E
T9	A, B, C

Solution: JavaTpoint

N.B. Consider all frequent itemsets to find association rules

Given: Minimum Support= 2, Minimum Confidence= 50%

Advantages & Disadvantages of Apriori Algorithm

Advantages of Apriori Algorithm

- This is easy to understand algorithm
- The join and prune steps of the algorithm can be easily implemented on large datasets.

Disadvantages of Apriori Algorithm

- The apriori algorithm works slow compared to other algorithms.
- The overall performance can be reduced as it scans the database for multiple times.
- The time complexity and space complexity of the apriori algorithm is $O(2^D)$, which is very high. Here D represents the horizontal width present in the database.

Association Rule Mining in Python

- Given a dataset, find the frequent itemsets and generate the association rules using the Apriori algorithm in Python

TID	*Items*
1	Bread, Diaper, Eggs
2	Milk, Diaper, Coke
3	Bread, Milk, Diaper, Coke
4	Milk, Coke

Association Rule Mining in Python (cont.)

```
# We will need the following Python library to install
# Command: pip install mlxtend
# load the libraries
import pandas as pd
# load dataset
dataset = [
    ["Bread", "Diaper", "Eggs"],
    ["Milk", "Diaper", "Coke"],
    ["Bread", "Milk", "Diaper", "Coke"],
    ["Milk", "Coke"],
]
```

TID	*Items*
1	Bread, Diaper, Eggs
2	Milk, Diaper, Coke
3	Bread, Milk, Diaper, Coke
4	Milk, Coke

Association Rule Mining in Python (cont.)

```
# Convert list (dataset) to dataframe with boolean values
from mlxtend.preprocessing import TransactionEncoder
te = TransactionEncoder()
te_array = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_array, columns=te.columns_)
print(df)
```

TID	*Items*
1	Bread, Diaper, Eggs
2	Milk, Diaper, Coke
3	Bread, Milk, Diaper, Coke
4	Milk, Coke

	Bread	Coke	Diaper	Eggs	Milk
0	True	False	True	True	False
1	False	True	True	False	True
2	True	True	True	False	True
3	False	True	False	False	True

Association Rule Mining in Python (cont.)

```
# Find frequently occurring itemsets using Apriori Algorithm

from mlxtend.frequent_patterns import apriori

frequent_itemsets_ap = apriori(df, min_support=0.5, use_colnames=True)

print(frequent_itemsets_ap)
```

	support	itemsets
0	0.50	(Bread)
1	0.75	(Coke)
2	0.75	(Diaper)
3	0.75	(Milk)
4	0.50	(Diaper, Bread)
5	0.50	(Diaper, Coke)
6	0.75	(Milk, Coke)
7	0.50	(Diaper, Milk)
8	0.50	(Diaper, Milk, Coke)

Association Rule Mining in Python (cont.)

```
# Mine the Association Rules

from mlxtend.frequent_patterns import association_rules

rules_ap = association_rules(frequent_itemsets_ap, metric="confidence", min_threshold=0.8)

print(rules_ap)
```

	antecedents	consequents	...	leverage	conviction
0	(Bread)	(Diaper)	...	0.1250	inf
1	(Milk)	(Coke)	...	0.1875	inf
2	(Coke)	(Milk)	...	0.1875	inf
3	(Diaper, Milk)	(Coke)	...	0.1250	inf
4	(Diaper, Coke)	(Milk)	...	0.1250	inf

Some Learning Materials

Lecture Notes for Chapter 6, Introduction to Data Mining by Tan, Steinbach, Kumar

A Comprehensive Guide on Market Basket Analysis

JavaTpoint: Association Rule Learning, Apriori Algorithm

Association Rule Mining in Python