Unsupervised Learning (Association)

Penyusun Modul: Chairul Aulia

Editor: Citra Chairunnisa

Association Rule Learning

Association rule learning is a type of unsupervised learning technique that **checks for the dependency** of one data item on another data item and maps accordingly so that it can be more profitable. It tries to find some **interesting relations** or **associations among the variables** of dataset. It is based on different rules to discover the interesting relations between variables in the database.

Association Rule Learning

Famous example, i.e. Market Basket Analysis is a technique used by the various big retailer to discover the associations between items.
For example, if a customer buys bread, he most likely can also buy butter, eggs, or milk, so these products are stored within a shelf or mostly nearby.
Association rule learning works on the concept of **If and Else Statement**, such as if A then B.

Common Ways to Measure Association

Measure 1: Support. Support indicates how frequently an item or item-set appears in the dataset or transaction.

Measure 2: Confidence. Confidence indicates how frequently a rule is found to be true.

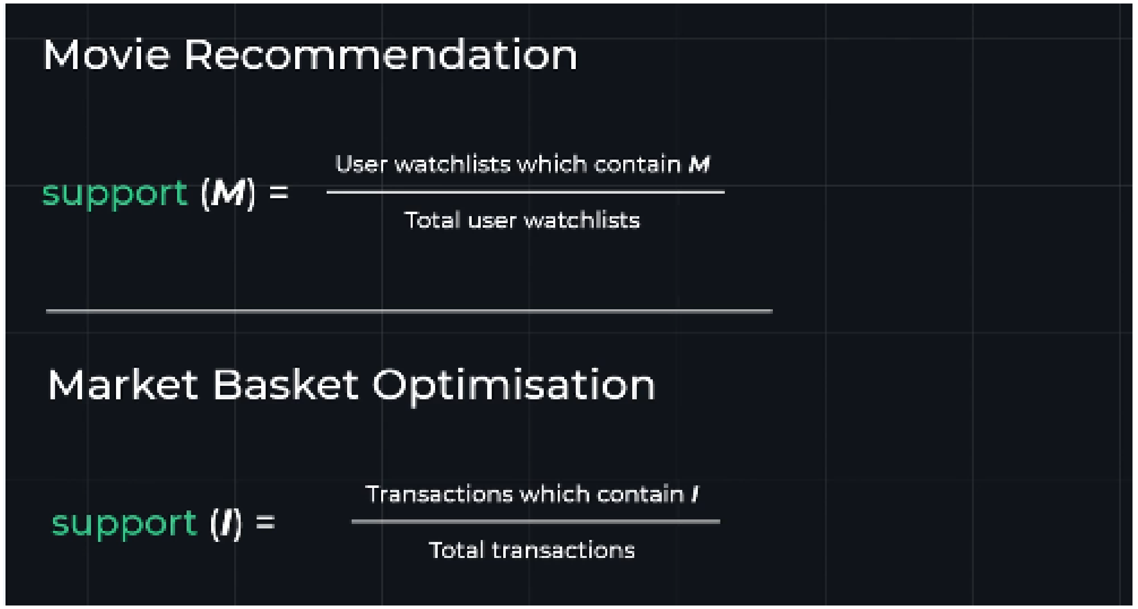
Measure 3: Lift. Lift (A →B) indicates the rise in probability of occurrence of B when A has already occurred.

Support

Support is **how frequently an item appears** in the dataset. It is defined as the fraction of the transaction N that contains the itemset X. If there are X datasets, then for transactions N, it can be written as:

$$support\ (X) = \frac{freq(X)}{N}$$

By computing support, we know whether the rule is important or not.



Movie Recommendation

$$\text{support } (M) = \frac{\text{User watchlists which contain } M}{\text{Total user watchlists}}$$

Market Basket Optimisation

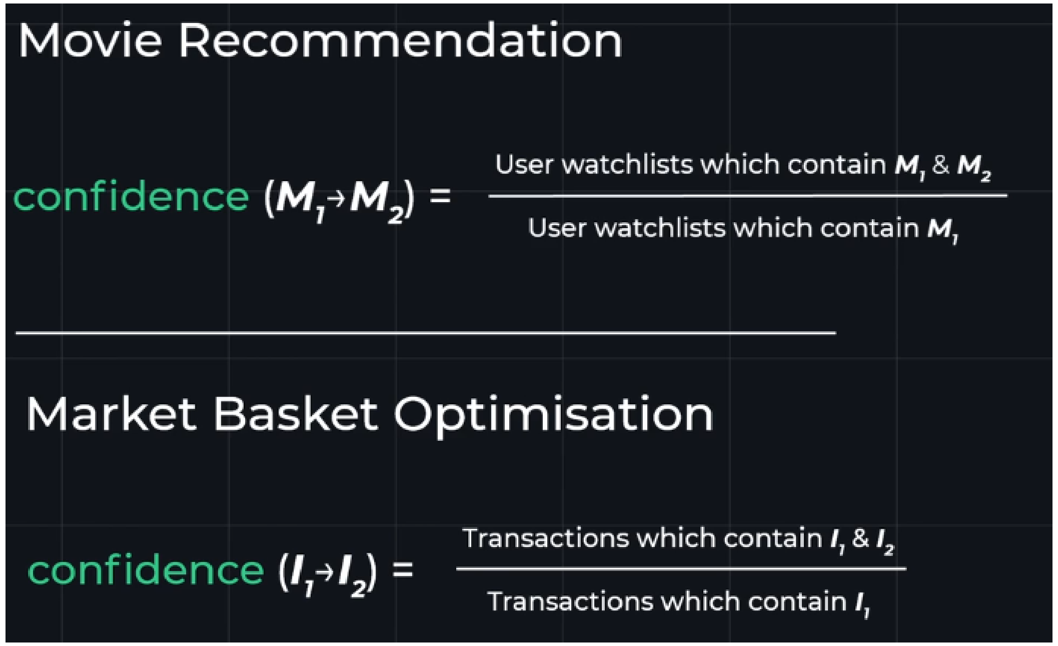
$$\text{support } (I) = \frac{\text{Transactions which contain } I}{\text{Total transactions}}$$

Confidence

Confidence indicates **how often the rule has been found to be true**. Or how often the items X and Y occur together in the dataset when the occurrence of X is already given. It is the ratio of the transaction that contains X and Y to the number of records that contain X.

$$confidence\ (X) = \frac{freq(X, Y)}{freq(X)}$$

One drawback of the confidence measure is that it might misrepresent the importance of an association. This is because it only accounts for how frequent X is, but not Y. If Y is also very frequent in general, there will be a higher chance that a transaction containing X will also contain Y, thus inflating the confidence measure. To account for the base frequency of both constituent items, we use a third measure called lift.



Movie Recommendation

$$\text{confidence}\ (M_1 \rightarrow M_2) = \frac{\text{User watchlists which contain } M_1 \text{ \& } M_2}{\text{User watchlists which contain } M_1}$$

Market Basket Optimisation

$$\text{confidence}\ (I_1 \rightarrow I_2) = \frac{\text{Transactions which contain } I_1 \text{ \& } I_2}{\text{Transactions which contain } I_1}$$

Lift

Lift gives **the correlation between X and Y in the rule X=>Y**. Correlation shows how one item-set X effects the item-set Y

$lift\ (X) = \frac{support(X,Y)}{support(X)\ support(Y)}$ or in another form $lift\ (X) = \frac{confidence(X,Y)}{support(Y)}$

For a rule: {X} → {Y} (or {antecedent} → {consequent})

- If lift = 1, then the probability of occurrence of X and Y is independent of each other.
- If lift < 1, then it means that having X reduces the probability of having Y
- If lift > 1, then it proves for high association between X and Y. It also helps us know that to what extent the occurrences are dependent on each other.

Example

Let's understand those 3 measures with an example

Transaction 1	🍎 🍺 🍚 🍗
Transaction 2	🍎 🍺 🍚
Transaction 3	🍎 🍺
Transaction 4	🍎 🍐
Transaction 5	🍼 🍺 🍚 🍗
Transaction 6	🍼 🍺 🍚
Transaction 7	🍼 🍺
Transaction 8	🍼 🍐

$$\text{Support}\ \{🍎\} = \frac{4}{8}$$

The support of {apple} is 4 out of 8, or 50%. Itemsets can also contain multiple items. For instance, the support of {apple, beer, rice} is 2 out of 8, or 25%

$$\text{Confidence}\ \{🍎 \rightarrow 🍺\} = \frac{\text{Support}\ \{🍎, 🍺\}}{\text{Support}\ \{🍎\}}$$

The confidence of {apple -> beer} is 3 out of 4, or 75%

$$\text{Lift}\ \{🍎 \rightarrow 🍺\} = \frac{\text{Support}\ \{🍎, 🍺\}}{\text{Support}\ \{🍎\} \times \text{Support}\ \{🍺\}}$$

The lift of {apple -> beer} is 3/8 out of (4/8 times 6/8), or 1. Remember the lift of 1 implies no association between items. It makes sense since beer is bought in several transactions, with or without apple!

Types of Association Rule Algorithms

- Apriori
- Eclat
- F-P Growth Algorithm

We'll just focus on the first 2

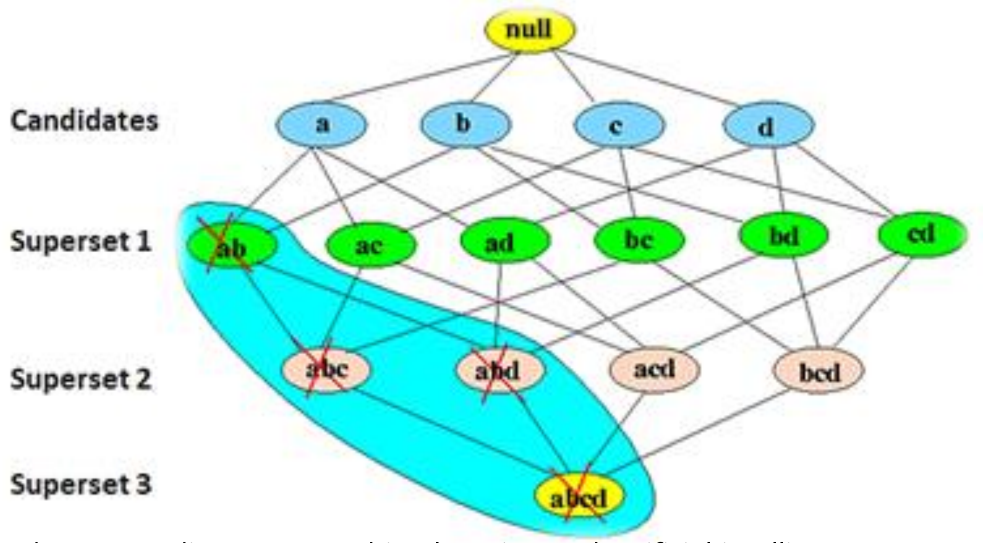
Apriori Algorithm

For large sets of data, there can be hundreds of items in hundreds of thousands transactions. The Apriori algorithm tries to extract rules for each possible combination of items. This process can be extremely slow due to the number of combinations. To speed up the process, we need to perform the following steps:

1. Set a minimum value for support and confidence (we are only interested in finding rules for the items that have certain default existence and have a minimum value for co-occurrence with other items)
2. Extract all the subsets having higher value of support than minimum threshold.
3. Select all the rules from the subsets with confidence value higher than minimum threshold.
4. Order the rules by descending order of Lift.

Apriori Algorithm

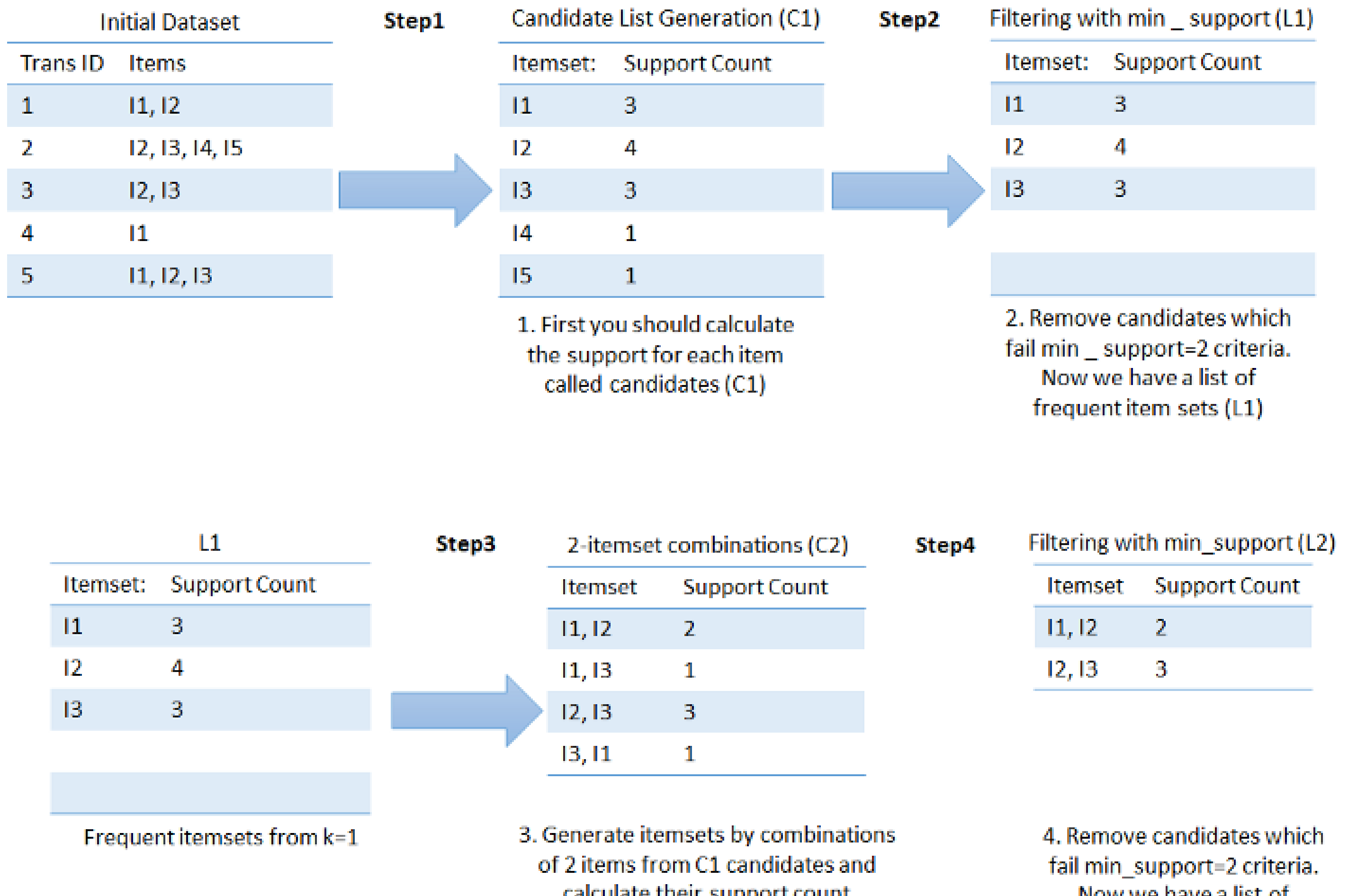
Apriori algorithm states: “Any subset of a frequent item set must also be frequent“. Conversely, if an itemset is infrequent then all of its supersets must be infrequent to. In other words, no super set of an infrequent item set must be generated or tested!



https://medium.com/machine-learning-and-artificial-intelligence/3-2-association-rule-mining-using-apriori-and-eclat-algorithms-b834fe2cf3da

If item-set {a,b} is infrequent (support<min_support) then we do not need to take into account all its super-sets (abc, abd, abcd). You can see how this principle reduces the number of total supersets for calculating the support

Apriori Algorithm



Initial Dataset

Trans ID	Items
1	I1, I2
2	I2, I3, I4, I5
3	I2, I3
4	I1
5	I1, I2, I3

Step1

Candidate List Generation (C1)

Itemset:	Support Count
I1	3
I2	4
I3	3
I4	1
I5	1

1. First you should calculate the support for each item called candidates (C1)

Step2

Filtering with min _ support (L1)

Itemset:	Support Count
I1	3
I2	4
I3	3

2. Remove candidates which fail min _ support=2 criteria. Now we have a list of frequent item sets (L1)

L1

Itemset:	Support Count
I1	3
I2	4
I3	3

Frequent itemsets from k=1

Step3

2-itemset combinations (C2)

Itemset	Support Count
I1, I2	2
I1, I3	1
I2, I3	3
I3, I1	1

3. Generate itemsets by combinations of 2 items from C1 candidates and calculate their support count

Step4

Filtering with min_support (L2)

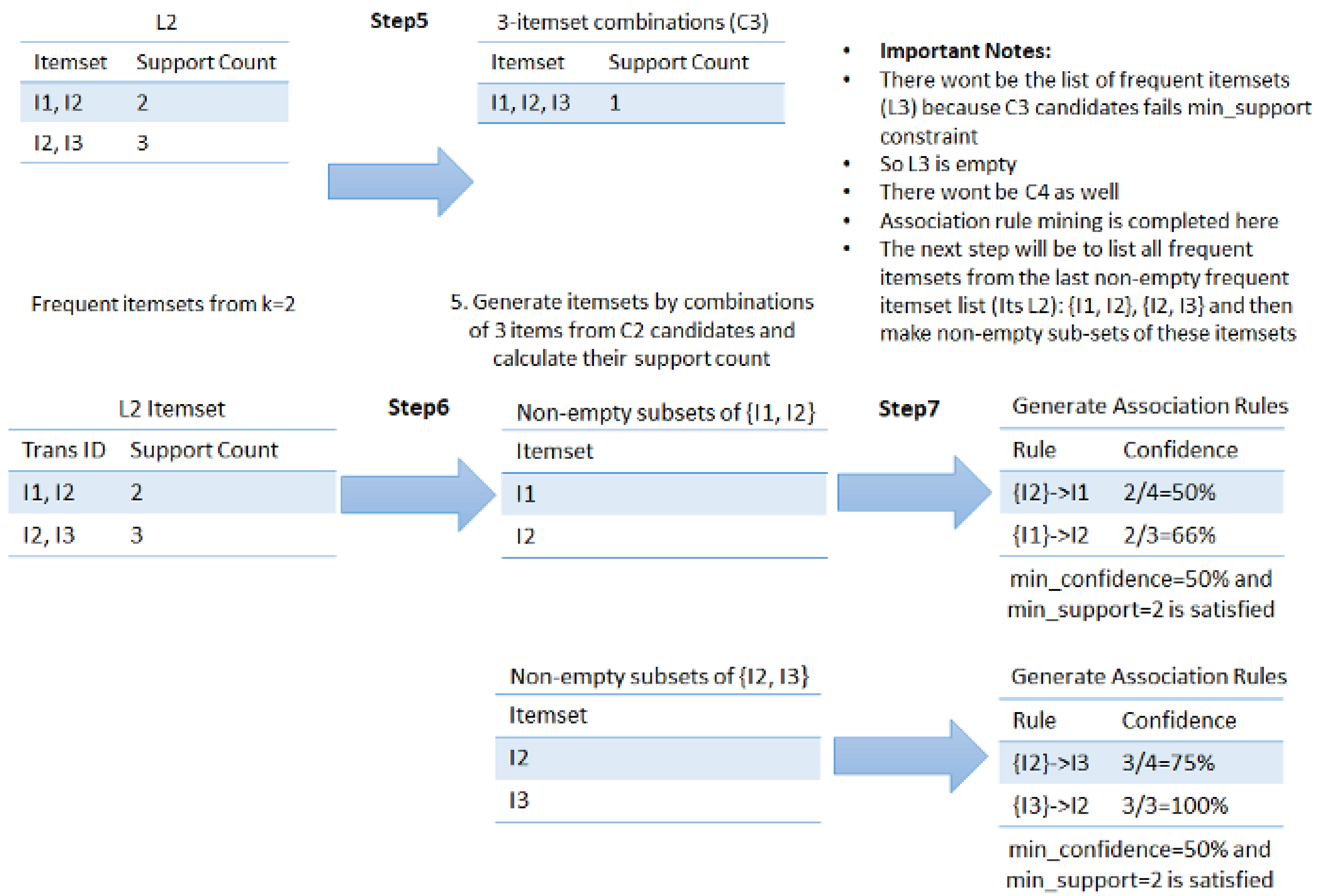
Itemset	Support Count
I1, I2	2
I2, I3	3

4. Remove candidates which fail min_support=2 criteria. Now we have a list of

1. Generate itemsets of k=1 and prune candidates whose support<min_support

2. Generate k=2 itemsets from previous frequent itemsets and prune candidates whose support<min_sup

Apriori Algorithm



L2

Itemset	Support Count
I1, I2	2
I2, I3	3

Frequent itemsets from k=2

Step5

3-itemset combinations (C3)

Itemset	Support Count
I1, I2, I3	1

5. Generate itemsets by combinations of 3 items from C2 candidates and calculate their support count

- **Important Notes:**
- There wont be the list of frequent itemsets (L3) because C3 candidates fails min_support constraint
- So L3 is empty
- There wont be C4 as well
- Association rule mining is completed here
- The next step will be to list all frequent itemsets from the last non-empty frequent itemset list (Its L2): {I1, I2}, {I2, I3} and then make non-empty sub-sets of these itemsets

L2 Itemset

Trans ID	Support Count
I1, I2	2
I2, I3	3

Step6

Non-empty subsets of {I1, I2}

Itemset
I1
I2

Step7

Generate Association Rules

Rule	Confidence
{I2}->I1	2/4=50%
{I1}->I2	2/3=66%

min_confidence=50% and min_support=2 is satisfied

Non-empty subsets of {I2, I3}

Itemset
I2
I3

Generate Association Rules

Rule	Confidence
{I2}->I3	3/4=75%
{I3}->I2	3/3=100%

min_confidence=50% and min_support=2 is satisfied

3. Generate k=3 frequent itemsets from k=2 frequent itemsets

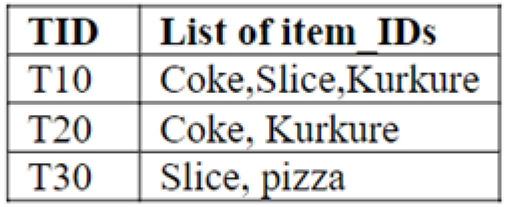
4. No new frequent item-sets can be generated. Creating rules. Finally, we got 4 strong rules:
   {I2}->I1
   {I1}->I2
   {I2}->I3
   {I3}->I2

Eclat Algorithm

The most commonly used layout is the horizontal data layout. That is, each transaction has a transaction identifier (TID) and a list of items occurring in that transaction, i.e., {TID:itemset}.

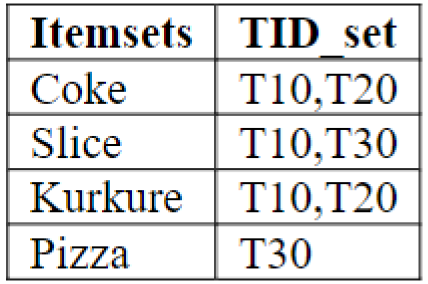
Another commonly used layout is the vertical data layout n which the database consists of a set of items, each followed by the set of transaction identifiers containing the item, i.e., {item:TID_set}.

Apriori algorithm uses horizontal format while Eclat can be used only for vertical format data sets



TID	List of item_IDs
T10	Coke,Slice,Kurkure
T20	Coke, Kurkure
T30	Slice, pizza

Horizontal Layout



Itemsets	TID_set
Coke	T10,T20
Slice	T10,T30
Kurkure	T10,T20
Pizza	T30

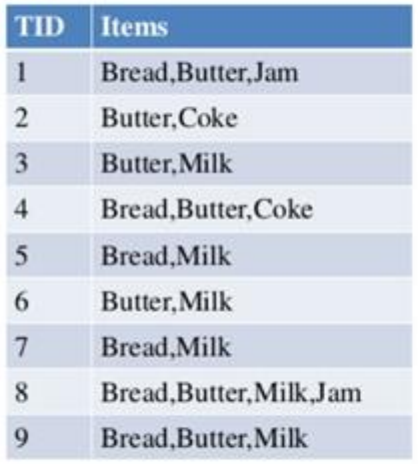
Vertical Layout

Eclat Algorithm

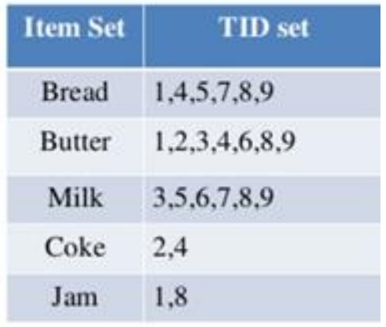
Let’s see an example of how Eclat works on vertical database:

1. Vertical data format. It is important to note that an item should not be appear more than once in the same transaction and also the items are assumed to be sorted by lexicographical order in a transaction.

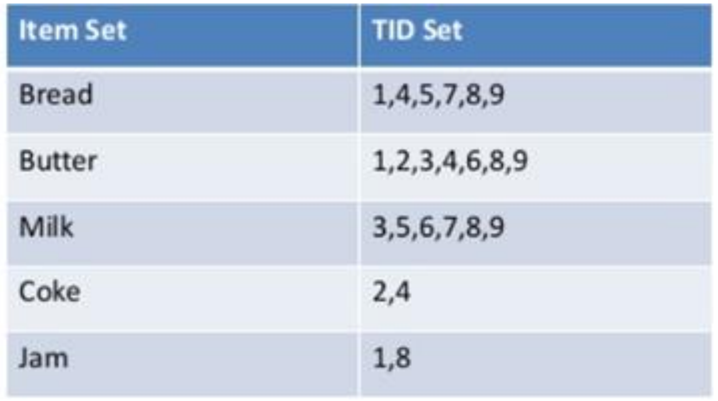
2. Here is the first call of function and all single items or data are used along with their respective tid_sets. Each frequent itemset is marked with its corresponding support value. The support of an itemset is given by number of times the itemset appears in the transaction database. Let’s say min_support=2



TID	Items
1	Bread,Butter,Jam
2	Butter,Coke
3	Butter,Milk
4	Bread,Butter,Coke
5	Bread,Milk
6	Butter,Milk
7	Bread,Milk
8	Bread,Butter,Milk,Jam
9	Bread,Butter,Milk



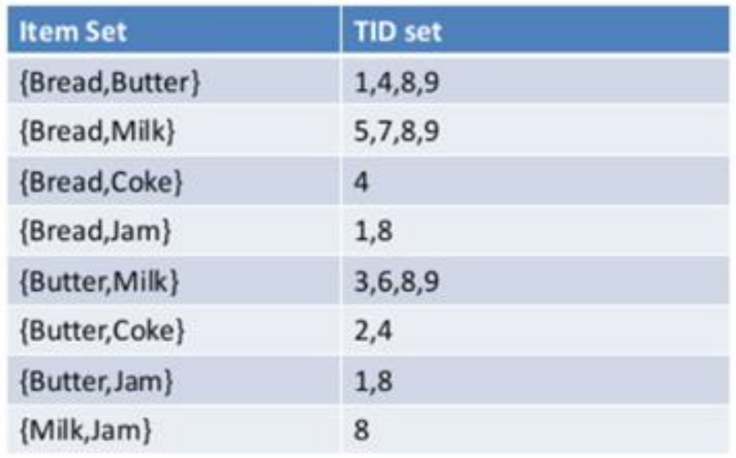
Item Set	TID set
Bread	1,4,5,7,8,9
Butter	1,2,3,4,6,8,9
Milk	3,5,6,7,8,9
Coke	2,4
Jam	1,8



Item Set	TID Set
Bread	1,4,5,7,8,9
Butter	1,2,3,4,6,8,9
Milk	3,5,6,7,8,9
Coke	2,4
Jam	1,8

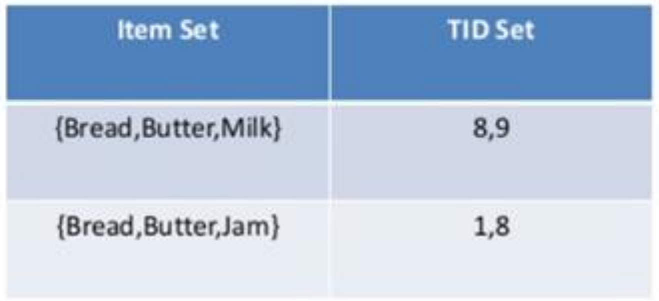
Eclat Algorithm

3. Here is recursive call of function (now k=2) and itemsets are generated. Let’s say min_support=2. So all itemsets whose support<min_support will be filtered out.



Item Set	TID set
{Bread,Butter}	1,4,8,9
{Bread,Milk}	5,7,8,9
{Bread,Coke}	4
{Bread,Jam}	1,8
{Butter,Milk}	3,6,8,9
{Butter,Coke}	2,4
{Butter,Jam}	1,8
{Milk,Jam}	8

4. Here is recursive call of function (now k=3) and frequent itemsets are generated.



Item Set	TID Set
{Bread,Butter,Milk}	8,9
{Bread,Butter,Jam}	1,8

This process repeats, with k incremented by 1 each time, until no frequent items or no candidate itemsets can be found.

The end result of Eclat algorithm is frequent item-sets with their support. We are not creating the rules with calculating confidence. We assume that frequent itemset is important and support is enough information to generate recommendations.

Advantages of Eclat

1. Since the Eclat algorithm uses a Depth-First Search approach, it consumes **less memory** than the Apriori algorithm
2. The Eclat algorithm does not involve in the repeated scanning of the data in order to calculate the individual support values
3. Eclat algorithm scans the currently generated dataset unlike Apriori which scans the original dataset

Apriori vs Eclat

The Eclat algorithm is naturally faster compared to the Apriori algorithm while the size of data set is small or medium, in case of large dataset there is a chance that Apriori performs faster, because intermediate Tidsets which are created in Eclat algorithm consumes more space in memory than Apriori and when we lave a large dataset intermediate results of vertical tid lists become too large for memory, thus affecting the algorithm scalability. So, **Eclat algorithm is better suited for small and medium datasets** where as **Apriori algorithm is used for large datasets.**

Some Use Cases for Association Rules

- **Medicine.** Doctors can use association rules to help diagnose patients. There are many variables to consider when making a diagnosis, as many diseases share symptoms. By using association rules and machine learning-fueled data analysis, doctors can determine the conditional probability of a given illness by comparing symptom relationships in the data from past cases.
- **Retail.** Retailers can collect data about purchasing patterns, recording purchase data as item barcodes are scanned by point-of-sale systems. Machine learning models can look for co-occurrence in this data to determine which products are most likely to be purchased together. The retailer can then adjust marketing and sales strategy to take advantage of this information.

Some Use Cases for Association Rules

- **User experience (UX) design.** Developers can collect data on how consumers use a website they create. They can then use associations in the data to optimize the website user interface -- by analyzing where users tend to click and what maximizes the chance that they engage with a call to action, for example.
- **Entertainment.** Services like Netflix and Spotify can use association rules to fuel their content recommendation engines. Machine learning models analyze past user behavior data for frequent patterns, develop association rules and use those rules to recommend content that a user is likely to engage with, or organize content in a way that is likely to put the most interesting content for a given user first.

Exercise

Let’s implement Apriori and Eclat algorithm using apyori and pyECLAT libraries

Use `Market_Basket_Optimisation` dataset for Apriori and the following list for Eclat

```
transactions = [
    ['beer', 'wine', 'cheese'],
    ['beer', 'potato chips'],
    ['eggs', 'flower', 'butter', 'cheese'],
    ['eggs', 'flower', 'butter', 'beer', 'potato chips'],
    ['wine', 'cheese'],
    ['potato chips'],
    ['eggs', 'flower', 'butter', 'wine', 'cheese'],
    ['eggs', 'flower', 'butter', 'beer', 'potato chips'],
    ['wine', 'beer'],
    ['beer', 'potato chips'],
    ['butter', 'eggs'],
    ['beer', 'potato chips'],
    ['flower', 'eggs'],
    ['beer', 'potato chips'],
    ['eggs', 'flower', 'butter', 'wine', 'cheese'],
    ['beer', 'wine', 'potato chips', 'cheese'],
    ['wine', 'cheese'],
    ['beer', 'potato chips'],
    ['wine', 'cheese'],
    ['beer', 'potato chips']
]
```