Machine Learning for Data Analysis

MSc in Data Analytics

CCT College Dublin

Association Rules – Market Basket Analysis

Week 10

Lecturer: Dr. Muhammad Iqbal*

Email: miqbal@cct.ie

Agenda

- Introduction
- Affinity Analysis
- Market Basket Analysis
- Support and Confidence
- Lift and Leverage
- Conviction
- Characteristics of Transaction Data
- The Apriori Principle
- FP Growth Algorithm

Introduction

- Suppose you are working as a manager in a super market and your boss comes to you and asks the following questions

Which products are purchased together most frequently?

How should the products be organized and positioned in the store?

How do we identify the best products to discount via coupons?

- You might respond with complete bewilderment, as those questions are very diverse and do not immediately answerable using a single algorithm and dataset.
- However, the answer to all those questions and many more is **Market Basket Analysis**.
- The general idea behind **Market Basket Analysis** is to identify and quantify which items, or groups of items, are purchased together frequently enough to drive insight into customer behaviour and product relationships.

Introduction

Affinity analysis is the study of attributes or characteristics that “go together.” Methods for affinity analysis, also known as market basket analysis, seek to uncover associations among these attributes.

- **Association Rules** take the form “If antecedent, then consequent,” along with a measure of the support and confidence associated with the rule. For example, a particular supermarket may find that of the 1000 customers shopping on a Thursday night, 200 bought diapers, and of the 200 who bought diapers, 50 bought beer. Thus, the association rule would be
- “If buy diapers, then buy beer,” with a support of 50/1000 = 5% and a confidence of 50/200 = 25%.
- **Examples of association tasks in business and research are the following**
  - Investigating the proportion of subscribers to your company's cell phone plan that respond positively to an offer of a service upgrade
  - Predicting degradation in telecommunications networks
  - Finding out which items in a supermarket are purchased together, and which items are never purchased together
  - Determining the proportion of cases in which a new drug will exhibit dangerous side effects

Market Basket Analysis: Association rules

- **Market basket analysis (MBA)** is a set of statistical affinity calculations that help the managers to better understand – and ultimately serve – their customers by highlighting purchasing patterns.
- Association rule mining is a technique that focuses upon observing frequently occurring patterns and associations from datasets found in the databases, such as relational and transactional databases.
- These rules do not say anything about the preferences of an individual, but they rely on the items within transactions to deduce a certain association.
- Every transaction is identified by a primary key (distinct ID) called, **transaction ID**. All these transactions are studied as a group and patterns are mined. **Association rules can be thought of as an if - then relationship.**
- We develop a rule: **if** an item **A** is being bought by the customer, **then** the chances of item **B** being picked by the customer too under the same transaction ID (along with item **A**) is found out.

Market Basket Analysis

Association rules

- There are two elements of these rules
- **Antecedent (if)**: This is an item/ group of items that are typically found in the item sets or datasets.
- **Consequent (then)**: This comes along as an item with an antecedent/ group of antecedents.
- We can see at the following rule

  {Bread, milk} ⇒ {Butter}

- The first part of this rule is called as **antecedent** and the second part (after the arrow) is called as **consequent**.
- It is able to convey that there is a chance of ***Butter*** being picked in a transaction if ***Bread*** and ***Milk*** are picked earlier. However, <u>the percentage chance for the consequent to be present in an itemset is not clear</u>.

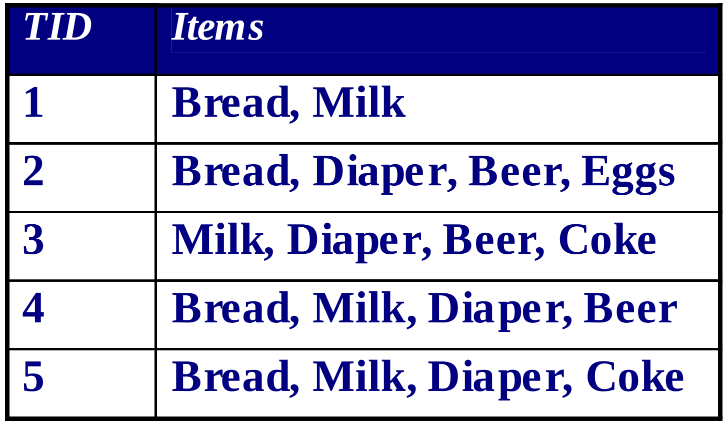
Support

- **Support** is simply the probability that a given item set appears in the data, which can be calculated by counting the number of transactions in which the item set appears and dividing that count by the total number of transactions.

- An item set can be a single item or a group of items. It is one of the primary metrics used to determine the believability and strength of association between groups of items.

- Note that since support is a probability, it will fall in the range [0, 1]. The formula takes the following form if the item set is two items, X and Y, and N is the total number of transactions

$$Support(X \Rightarrow Y) = Support(X,Y) = P(X,Y) = \frac{Frequency(X,Y)}{N}$$



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:

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

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

Confidence

- The confidence metric can be thought of in terms of conditional probability, as it is basically the probability that product **B** is purchased given the purchase of product **A**.

- Like support, confidence is a probability, so its range is [0, 1]. Using the same variable definitions as in the Support, the following is the formula for confidence

$$Confidence\ (X \Rightarrow Y) = P(Y|X) = \frac{Support(X,Y)}{P(X)} = \frac{\frac{Frequency(X,Y)}{N}}{\frac{Frequency(X)}{N}}$$

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 and Leverage

- When lift equals 1, the two products are independent and, hence, no conclusions can be made about product B when product A is purchased

$$Lift(X \Rightarrow Y) = \frac{Support\ (X,Y)}{Support\ (X) * Support\ (Y)} = \frac{P(X,Y)}{P(X) * P(Y)}$$

- **Leverage** calculates the difference between the two cases, so its range is [-1, 1].
- **Leverage** equaling zero can be interpreted the same way as lift equaling one.

$$Leverage(X \Rightarrow Y) = Support(X,Y) - (Support(X) * Support(Y)) = P(X,Y) - (P(X) * P(Y))$$

- **A value of lift other than 1 means that some dependency exists between the items.**

Conviction

- **Conviction** is the ratio of the expected frequency that X occurs without Y, given that X and Y are independent of the frequency of incorrect predictions.

- A value greater than 1 is ideal because that means the association between products or item sets X and Y is incorrect more often if the association between X and Y is random (in other words, X and Y are independent).

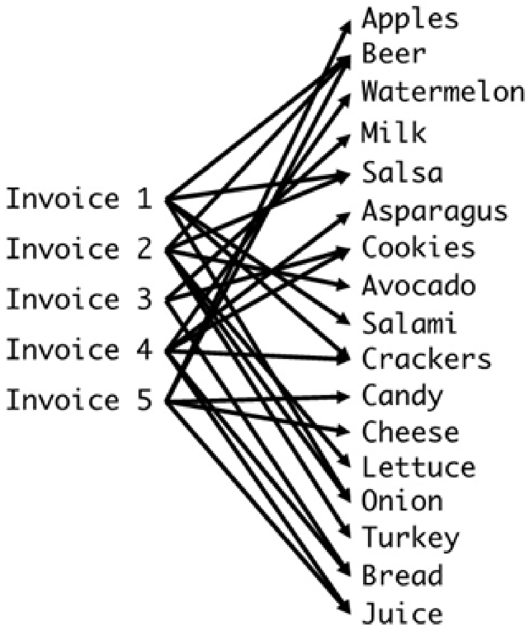
$$Conviction(X \Rightarrow Y) = \frac{1 - Support(Y)}{1 - Confidence\ (X \Rightarrow Y)}$$

- The data used in the market basket analysis is transaction data or any type of data that resembles transaction data.

- In its most basic form, transaction data has some sort of transaction identifier, such as an invoice or transaction number, and a list of products associated with said identifier.

- Transaction data includes pricing information, dates and times, and customer identifiers, among many other things.

- How is each product mapped to multiple invoices?



The Apriori Principle

- The **Apriori algorithm** is a data mining methodology for identifying and quantifying frequent item sets in transaction data and is the foundational component of association rule learning.
- Frequency is quantified as support, so the value input into the model is the minimum support acceptable for the analysis being done.
- **The model identifies all item sets whose support is greater than, or equal to, the minimum support provided to the model.**
- The minimum support hyperparameter is not a value that can be optimized via grid search because there is no evaluation metric for the Apriori algorithm. Instead, the minimum support parameter is set based on the data, the use case, and domain expertise.
- The main idea behind the Apriori algorithm is the Apriori principle: **any subset of a frequent item set must itself be frequent.**
- Another aspect worth mentioning is the corollary: **no superset of an infrequent item set can be frequent.**

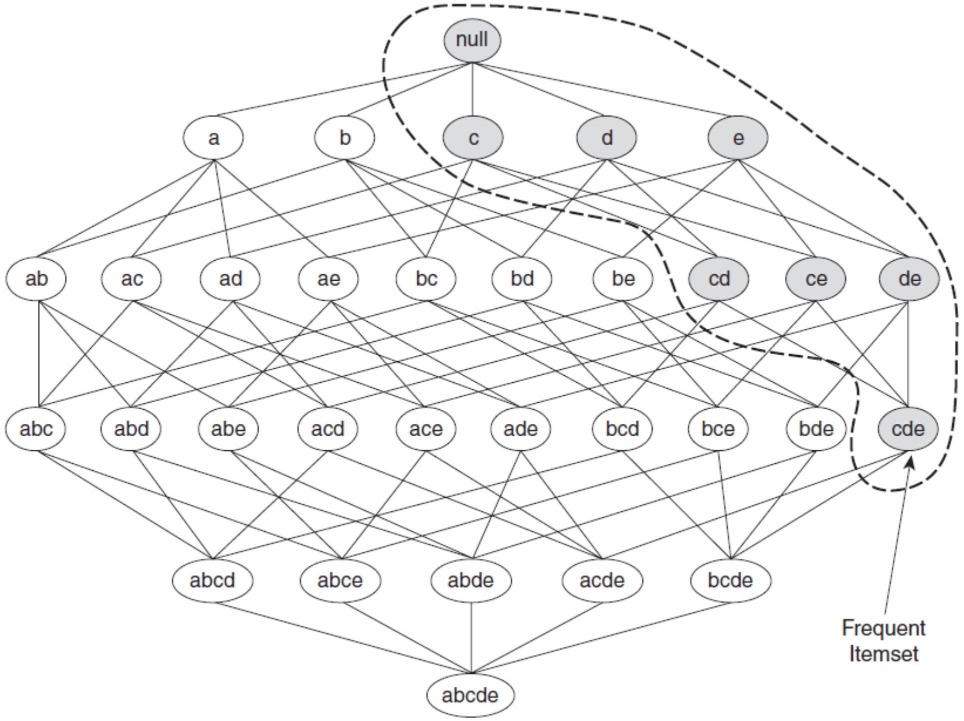
The Apriori Principle

- To illustrate the idea behind the Apriori principle, consider the itemset lattice as shown in Figure.

  Suppose **{c, d, e}** is a frequent itemset. Clearly, any transaction that contains {c, d, e} must also contain its subsets, {c, d}, {c, e}, {d, e}, {c}, {d}, and {e}. As a result, if {c, d, e} is frequent, then all subsets of {c, d, e} (i.e., the shaded item sets in this figure) must also be frequent.

- We can see that the support measure helps us to reduce the number of candidate item sets explored during frequent itemset generation.

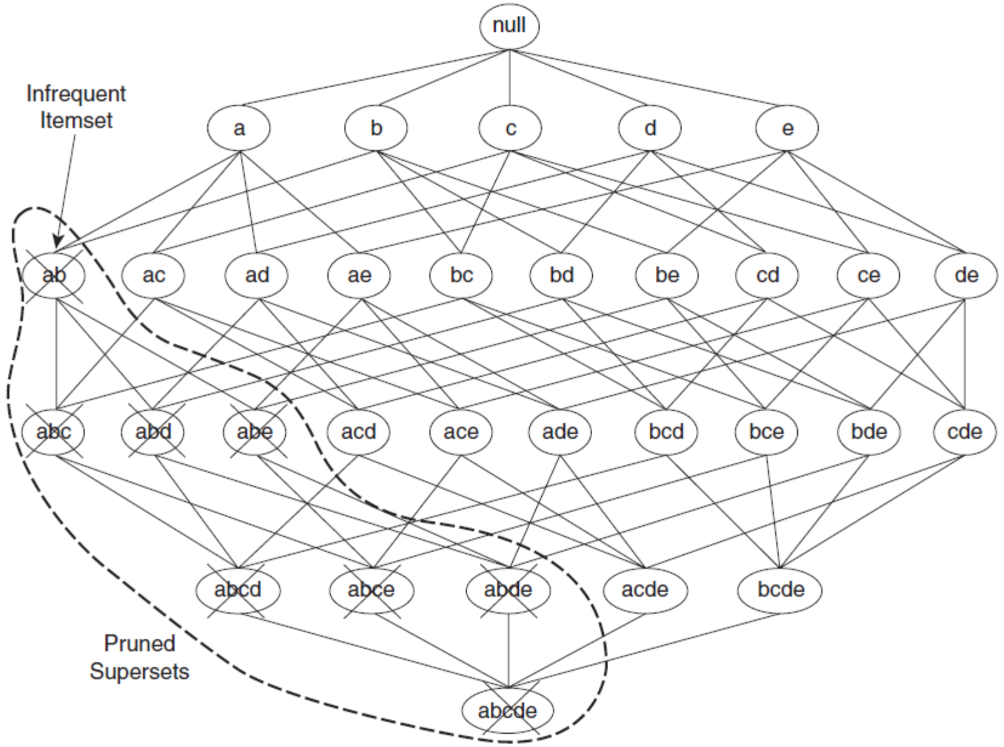
- The use of support for pruning candidate item sets is guided by the following principle. **If an itemset is frequent, then all of its subsets must also be frequent.**



An illustration of the *Apriori* principle. If $\{c, d, e\}$ is frequent, then all subsets of this itemset are frequent.

The Apriori Principle

- If an itemset such as **{a, b}** is infrequent, then all of its supersets must be infrequent too.
- The entire subgraph containing the supersets of **{a, b}** can be pruned immediately once **{a, b}** is found to be infrequent.
- This strategy of trimming the exponential search space based on the support measure is known as **support-based pruning**.
- Such a **pruning strategy** is made possible by a key property of the support measure, namely, that the support for an itemset never exceeds the support for its subsets.
- This property is also known as the **anti-monotone property** of the support measure.



An illustration of support-based pruning. If $\{a, b\}$ is infrequent, then all supersets of $\{a, b\}$ are infrequent.

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

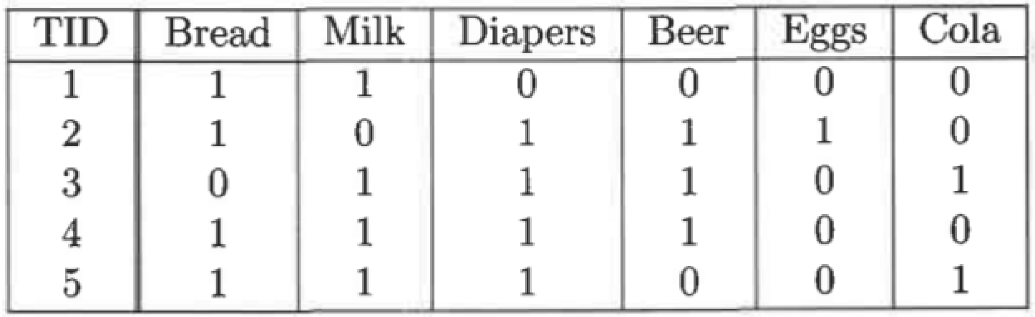
Itemset
{Bread,Milk}
{Bread, Beer }
{Bread,Diaper}
{Beer, Milk}
{Diaper, Milk}
{Beer,Diaper}

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

A binary 0/1 representation of market basket data.



TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

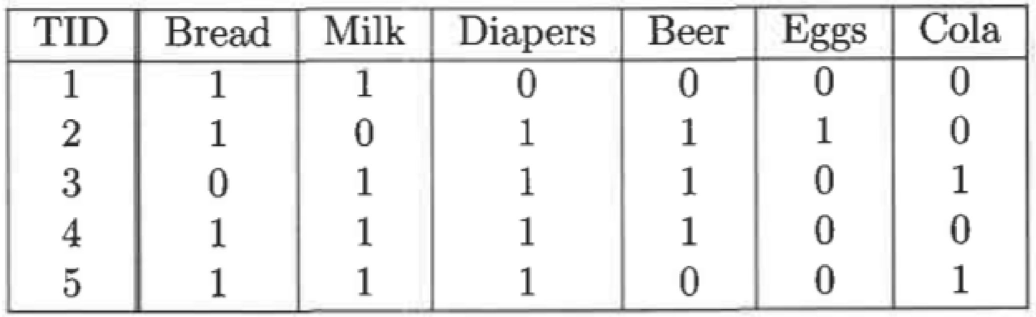
Itemset	Count
{Bread,Milk}	**3**
{Beer, Bread}	**2**
{Bread,Diaper}	**3**
{Beer,Milk}	**2**
{Diaper,Milk}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

A binary 0/1 representation of market basket data.



TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

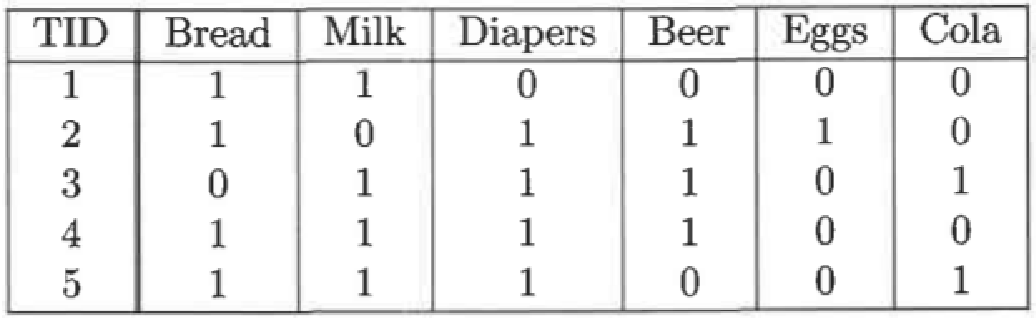
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset
{ Beer, Diaper, Milk}
{ Beer,Bread,Diaper}
{Bread, Diaper, Milk}
{ Beer, Bread, Milk}

A binary 0/1 representation of market basket data.



TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

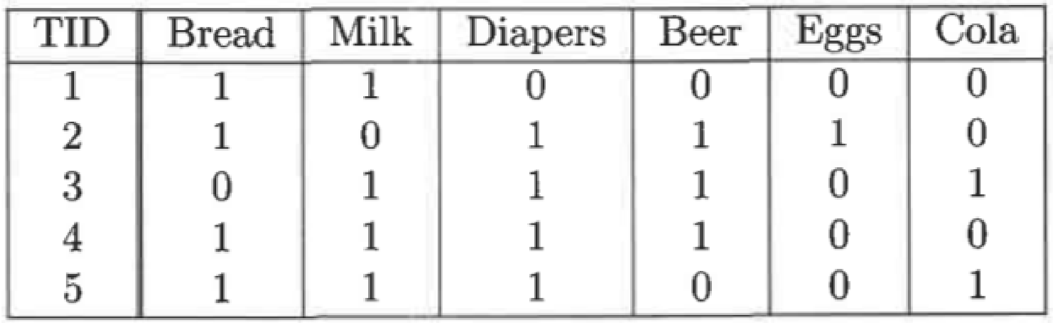
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{ Beer, Diaper, Milk}	**2**
{ Beer,Bread, Diaper}	**2**
{Bread, Diaper, Milk}	**2**
{Beer, Bread, Milk}	**1**

A binary 0/1 representation of market basket data.



TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Illustrating Apriori Principle

Item	Count
Bread	**4**
Coke	**2**
Milk	**4**
Beer	**3**
Diaper	**4**
Eggs	**1**

Items (1-itemsets)

Itemset	Count
{Bread,Milk}	**3**
{Bread,Beer}	**2**
{Bread,Diaper}	**3**
{Milk,Beer}	**2**
{Milk,Diaper}	**3**
{Beer,Diaper}	**3**

Pairs (2-itemsets)

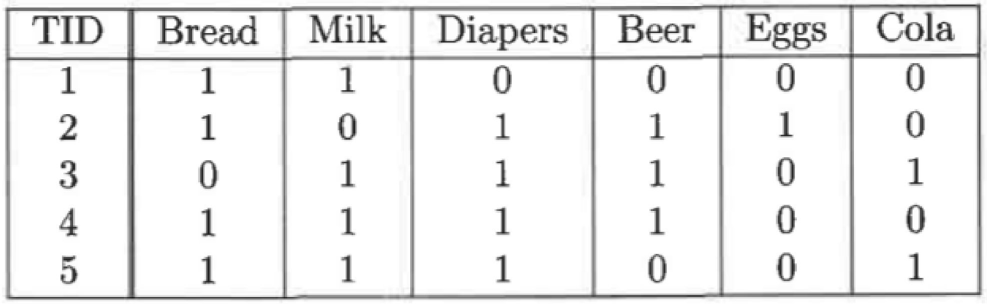
(No need to generate candidates involving Coke or Eggs)

Minimum Support = 3

Triplets (3-itemsets)

Itemset	Count
{Bread, Diaper, Milk}	**2**

A binary 0/1 representation of market basket data.



TID	Bread	Milk	Diapers	Beer	Eggs	Cola
1	1	1	0	0	0	0
2	1	0	1	1	1	0
3	0	1	1	1	0	1
4	1	1	1	1	0	0
5	1	1	1	0	0	1

Use of F_{k-1} x F_{k-1} method for candidate generation results in only one 3-itemset. This is eliminated after the support counting step.

Apriori Algorithm

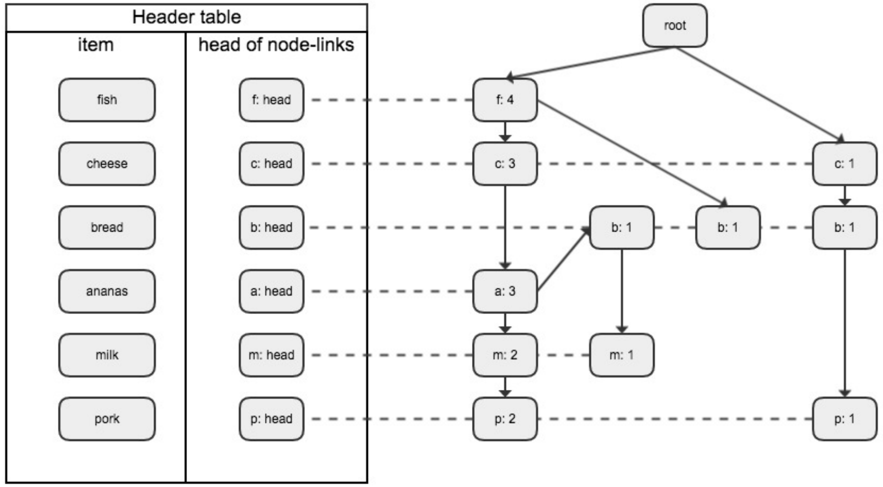
  - F_k: frequent k-itemsets
  - L_k: candidate k-itemsets

- **Algorithm**
  - Let k = 1
  - Generate F_1 = {frequent 1-itemsets}
  - Repeat until F_k is empty
    - **Candidate Generation**: Generate L_{k+1} from F_k
    - **Candidate Pruning**: Prune candidate item sets in L_{k+1} containing subsets of length k that are infrequent.
    - **Support Counting**: Count the support of each candidate in L_{k+1} by scanning the DB.
    - **Candidate Elimination**: Eliminate candidates in L_{k+1} that are infrequent, leaving only those that are frequent => F_{k+1}

FP-Growth Algorithm

- **Frequent Patterns (FP-growth)** is an improved version of the Apriori Algorithm which is widely used for frequent pattern mining (AKA Association Rule Mining).
- The method involved repeatedly searching the entire transaction database to create costly pattern candidates that were longer and longer, then evaluating their support.
- The fundamental concept behind FP-growth is to first carefully scan the transaction database D of interest, identify all of the frequently occurring patterns of length 1, and then construct an FP-tree (a unique type of tree structure) based on these patterns.
- After this step is completed, we perform recursive computations on the typically much smaller FP-tree rather than working with D.
- Since it builds trees recursively from the original tree's subtrees in order to find patterns, this stage of the algorithm is known as the FP-growth step.
- This process, which we will refer to as fragment pattern growth, is based on a divide-and-conquer tactic that significantly lowers the workload in each recursion step rather than requiring us to generate candidates.

FP-Growth Algorithm



Header table

item	head of node-links
fish	f: head
cheese	c: head
bread	b: head
ananas	a: head
milk	m: head
pork	p: head

The two primary drawbacks of the Apriori Algorithm are

1. At each step, candidate sets have to be built.
2. To build the candidate sets, the algorithm has to repeatedly scan the database.

- **Step 1:** Deduce the ordered frequent items. For items with the same frequency, the order is given by the alphabetical order.
- **Step 2:** Construct the **FP-tree** from the above data
- **Step 3:** From the **FP-tree** above, construct the **FP-conditional tree** for each item (or itemset).
- **Step 4:** Determine the frequent patterns.
- Let's take a look at an example of how to generate an **FP-tree**. Find all frequent item sets with **support ≥ 2**. First, find all items with support **count ≥ 2**.

FP-Growth Algorithm

- The given data is a hypothetical dataset of transactions with each letter representing an item. The frequency of each individual item is computed.
- Let the minimum support be 3. A **Frequent Pattern** set is built which will contain all the elements whose frequency is greater than or equal to the minimum support.
- These elements are stored in the descending order of their respective frequencies. After insertion of the relevant items, the set **L** looks like this
- **L = {K : 5, E : 4, M : 3, O : 3, Y : 3}**
- For each transaction, the respective Ordered-Item set is built. It is done by iterating the **Frequent Pattern set** and checking if the current item is contained in the transaction in question.
- If the current item is contained, the item is inserted in the Ordered-Item set for the current transaction. The following table is built for all the transactions.

Transaction ID	Items
T1	{E,K,M,N,O,Y}
T2	{D,E,K,N,O,Y}
T3	{A,E,K,M}
T4	{C,K,M,U,Y}
T5	{C,E,I,K,O,O}

Item	Frequency
A	1
C	2
D	1
E	4
I	1
K	5
M	3
N	2
O	3
U	1
Y	3

Transaction ID	Items	Ordered-Item Set
T1	{E,K,M,N,O,Y}	{K,E,M,O,Y}
T2	{D,E,K,N,O,Y}	{K,E,O,Y}
T3	{A,E,K,M}	{K,E,M}
T4	{C,K,M,U,Y}	{K,M,Y}
T5	{C,E,I,K,O,O}	{K,E,O}

FP-Growth Algorithm

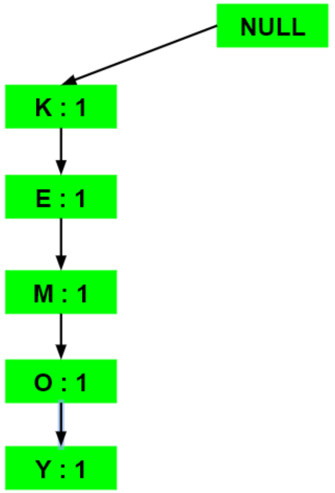
- Now, all the Ordered-Item sets are inserted into a Tree Data Structure.

a) Inserting the set {K, E, M, O, Y}

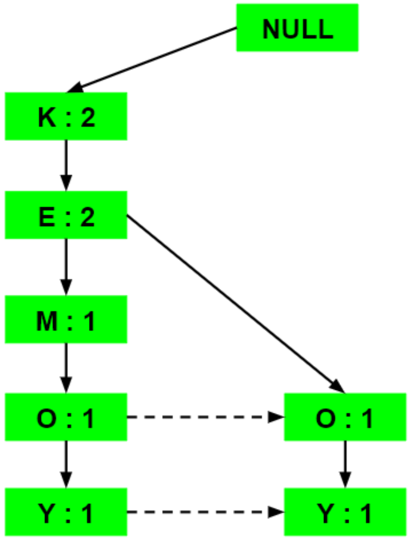
- All the items are simply linked one after the other in the order of occurrence in the set and initialize the support count for each item as 1.

b) Inserting the set {K, E, O, Y}

- Till the insertion of the elements K and E, simply the support count is increased by 1. On inserting O we can see that there is no direct link between E and O, therefore a new node for the item O is initialized with the support count as 1 and item E is linked to this new node. On inserting Y, we first initialize a new node for the item Y with support count as 1 and link the new node of O with the new node of Y.



(a)



(b)

FP-Growth Algorithm

- Now, all the Ordered-Item sets are inserted into a Tree Data Structure.

c) Inserting the set {K, E, M}

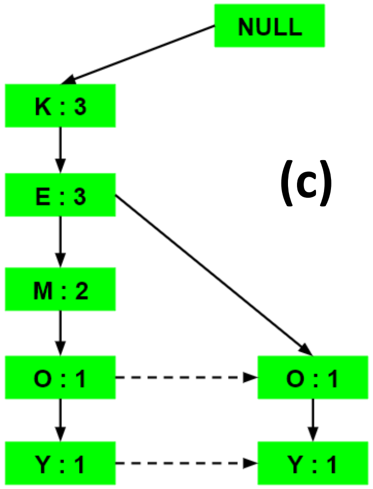
- Here simply the support count of each element is increased by 1.

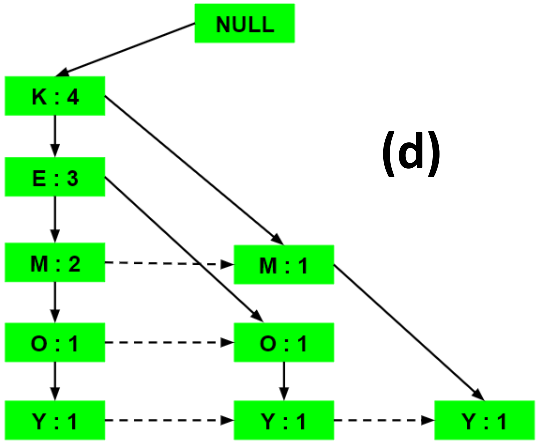
d) Inserting the set {K, M, Y}

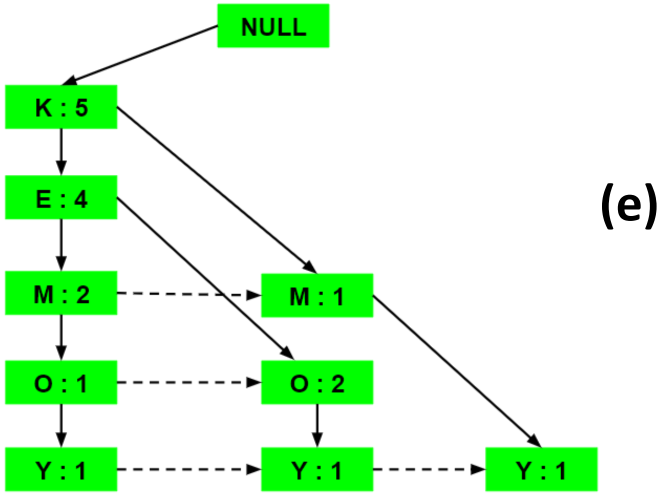
- Similar to step b), first the support count of K is increased, then new nodes for M and Y are initialized and linked accordingly.

e) Inserting the set {K, E, O}

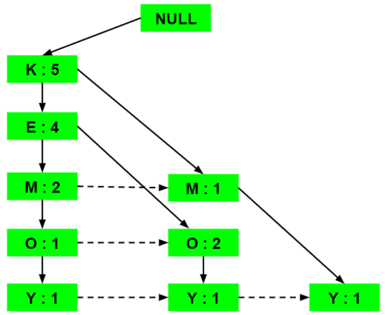
- We simply the support counts of the respective elements are increased. Note that the support count of the new node of item O is increased.



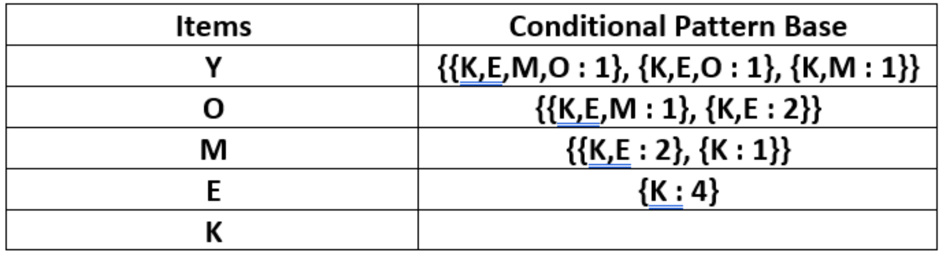




FP-Growth Algorithm

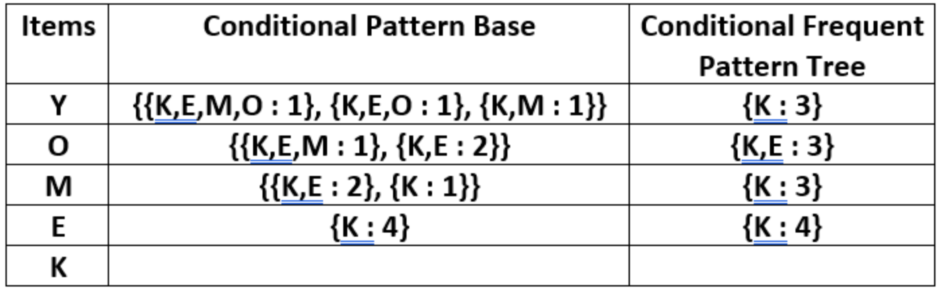


- For each item, the **Conditional Pattern Base** is computed which is path labels of all the paths which lead to any node of the given item in the frequent-pattern tree. Note that the items in the below table are arranged in the ascending order of their frequencies.



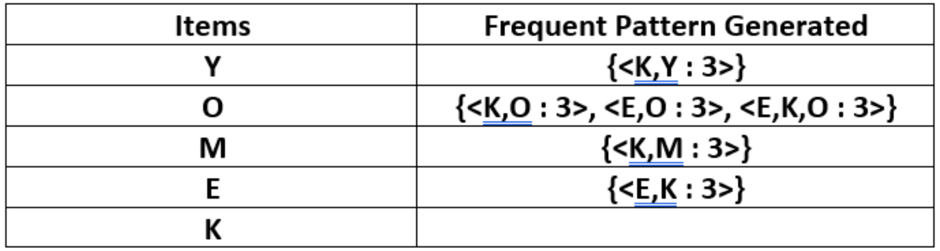
Items	Conditional Pattern Base
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}
O	{{K,E,M : 1}, {K,E : 2}}
M	{{K,E : 2}, {K : 1}}
E	{K : 4}
K	

- For each item, the **Conditional Frequent Pattern Tree** is built. It is done by taking the set of elements which is common in all the paths in the Conditional Pattern Base of that item and calculating it's support count by summing the support counts of all the paths in the Conditional Pattern Base.



Items	Conditional Pattern Base	Conditional Frequent Pattern Tree
Y	{{K,E,M,O : 1}, {K,E,O : 1}, {K,M : 1}}	{K : 3}
O	{{K,E,M : 1}, {K,E : 2}}	{K,E : 3}
M	{{K,E : 2}, {K : 1}}	{K : 3}
E	{K : 4}	{K : 4}
K		

- From the Conditional Frequent Pattern tree, the Frequent Pattern rules are generated by pairing the items of the **Conditional Frequent Pattern Tree set** to the corresponding item as given in the below table.



Items	Frequent Pattern Generated
Y	{<K,Y : 3>}
O	{<K,O : 3>, <E,O : 3>, <E,K,O : 3>}
M	{<K,M : 3>}
E	{<E,K : 3>}
K	

- For each row, two types of association rules can be inferred for example for the first row which contains the element, the rules **K -> Y** and **Y -> K** can be inferred. To determine the valid rule, the confidence of both the rules is calculated and the one with confidence greater than or equal to the minimum confidence value is retained.

Resources/ References

- The Unsupervised Learning Workshop by Christopher Kruger; Benjamin Johnston; Aaron JonesPublished by Packt Publishing, 2020.

- Hands-On Unsupervised Learning Using Python, by Ankur A. Patel, Published by O'Reilly Media, Inc., 2019.

- Mastering Machine Learning with Spark 2.x, Alex Tellez, Max Pumperla, Michal Malohlava, Packt Publishing, August 2017

- https://www.geeksforgeeks.org/ml-frequent-pattern-growth-algorithm