Open in Colab YouTube

Introduction to Association Rule Mining – Market Basket Analysis (MBA)

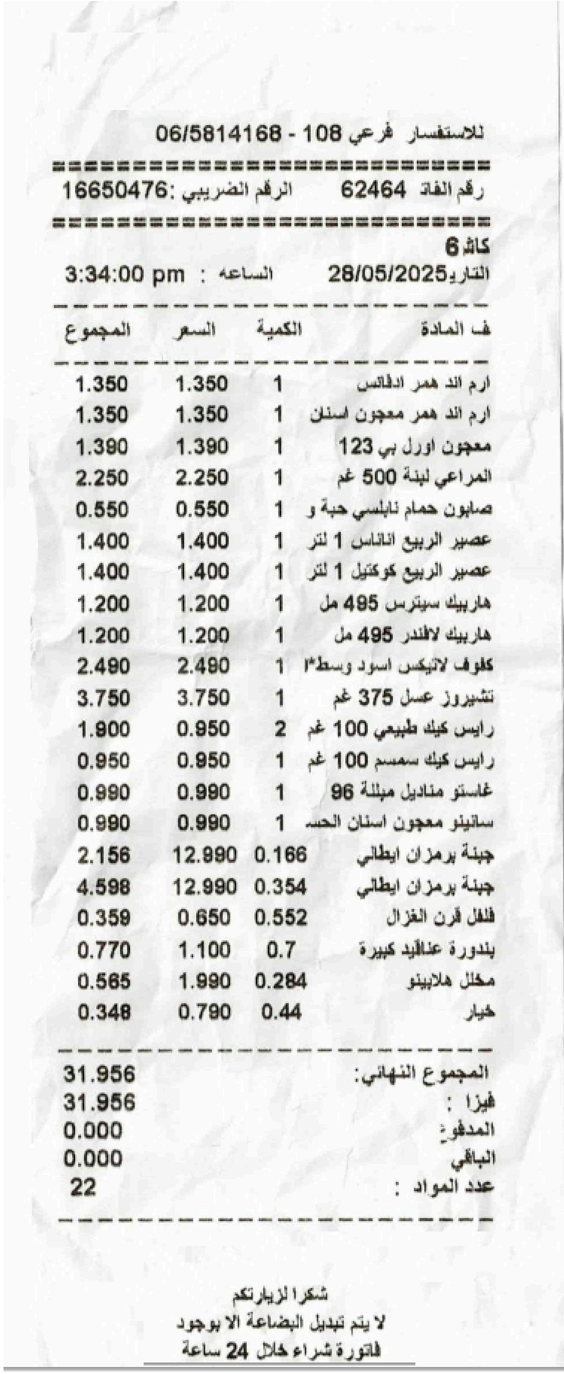
Association rule mining is a data mining technique used to uncover **if–then relationships** between items in large datasets, such as retail transaction logs. These relationships, known as **association rules**, capture patterns of items that frequently occur together (called **co-occurrences**).

In the context of **market basket analysis**, association rules help answer questions like: *"Which products are often bought together?"* For example, suppose we find that 75% of customers who buy cereal also buy milk. We can express this as the rule:

$$\{\text{cereal}\} \Rightarrow \{\text{milk}\}$$

This rule suggests that customers who purchase cereal often purchase milk as well. Such insights can guide **marketing and retail decisions**, including promotional strategies, product bundling, and shelf placement.

Market Basket Analysis
Are Bread Purchased When Banana And Egg Are Bought Together?
Where Should Apple Juice Be Placed In The Store To Maximize Their Sales?
Is Soda Typically Purchased With Bananas?
Does The Brand Of Soda Make A Difference?
How Are The Demographics Of The Neighborhood Affecting What Customers Are Buying?



للاستفسار فرعي 108 - 06/5814168

==============================

رقم الفاتورة 62464 الرقم الضريبي :16650476

==============================

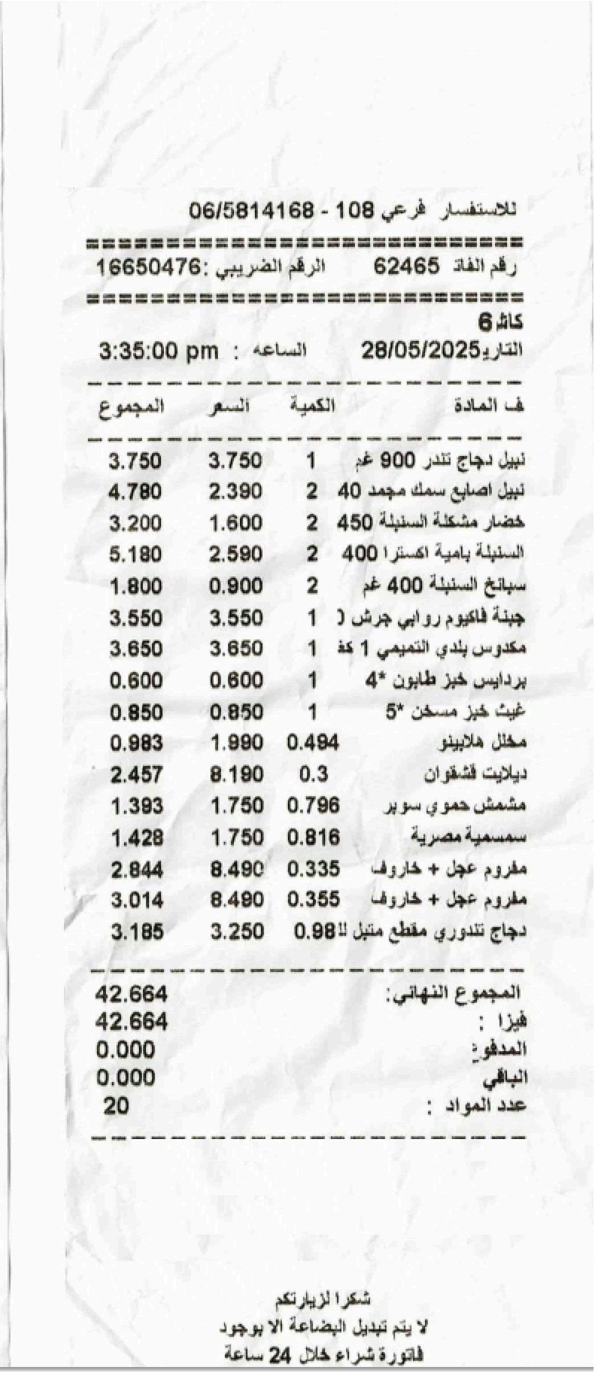
كاش 6

التاريخ28/05/2025 الساعه : 3:34:00 pm

ف المادة	الكمية	السعر	المجموع
ارم اند همر ادفانس	1	1.350	1.350
ارم اند همر معجون اسنان	1	1.350	1.350
معجون اورل بي 123	1	1.390	1.390
المراعي لبنة 500 غم	1	2.250	2.250
صابون حمام نابلسي حبة و	1	0.550	0.550
عصير الربيع اناناس 1 لتر	1	1.400	1.400
عصير الربيع كوكتيل 1 لتر	1	1.400	1.400
هاربيك سيترس 495 مل	1	1.200	1.200
هاربيك لافندر 495 مل	1	1.200	1.200
كفوف لاتيكس اسود وسط*٣	1	2.490	2.490
تشيروز عسل 375 غم	1	3.750	3.750
رايس كيك طبيعي 100 غم	2	0.950	1.900
رايس كيك سمسم 100 غم	1	0.950	0.950
غاستو مناديل مبللة 96	1	0.990	0.990
سانيتو معجون اسنان الحس	1	0.990	0.990
جبنة برمزان ايطالي	0.166	12.990	2.156
جبنة برمزان ايطالي	0.354	12.990	4.598
فلفل قرن الغزال	0.552	0.650	0.359
بندورة عناقيد كبيرة	0.7	1.100	0.770
مخلل هلابينو	0.284	1.990	0.565
خيار	0.44	0.790	0.348

المجموع النهائي:	31.956
فيزا :	31.956
المدفوع	0.000
الباقي	0.000
عدد المواد :	22

شكرا لزيارتكم
لا يتم تبديل البضاعة الا بوجود
فاتورة شراء خلال 24 ساعة



للاستفسار فرعي 108 - 06/5814168

==============================

رقم الفاتورة 62465 الرقم الضريبي :16650476

==============================

كاش 6

التاريخ28/05/2025 الساعه : 3:35:00 pm

ف المادة	الكمية	السعر	المجموع
نبيل دجاج تندر 900 غم	1	3.750	3.750
نبيل اصابع سمك مجمد 40	2	2.390	4.780
خضار مشكلة السنبلة 450	2	1.600	3.200
السنبلة بامية اكسترا 400	2	2.590	5.180
سبانخ السنبلة 400 غم	2	0.900	1.800
جبنة فاكيوم روابي جرش (	1	3.550	3.550
مكدوس بلدي التميمي 1 كغ	1	3.650	3.650
برادايس خبز طابون *4	1	0.600	0.600
غيث خبز مسخن *5	1	0.850	0.850
مخلل هلابينو	0.494	1.990	0.983
ديلايت قشقوان	0.3	8.190	2.457
مشمش حموي سوبر	0.796	1.750	1.393
سمسمية مصرية	0.816	1.750	1.428
مفروم عجل + خاروف	0.335	8.490	2.844
مفروم عجل + خاروف	0.355	8.490	3.014
دجاج تندوري مقطع متبل لا	0.98	3.250	3.185

المجموع النهائي:	42.664
فيزا :	42.664
المدفوع	0.000
الباقي	0.000
عدد المواد :	20

شكرا لزيارتكم
لا يتم تبديل البضاعة الا بوجود
فاتورة شراء خلال 24 ساعة

Video Lecture
محاضرة مسجلة

Association Rule Use Cases and Domains

#	Domain	Use Case	Description (what + why it’s useful)
1	Healthcare	Disease diagnosis	Identifying associations between symptoms and diseases to support faster, more accurate diagnosis **Example:** {fever, cough} → {flu}
2	Healthcare	Drug interactions	Discovering meds often prescribed together or risky combos to improve treatment safety **Example:** {drug A, drug B} → {adverse reaction}
3	Web Usage Mining	Website optimization	Analyzing navigation paths to remove friction and increase conversions **Example:** {homepage, product page} → {checkout}
4	Web Usage Mining	Recommendation systems	Finding frequently co-accessed content to power “you may also like” recommendations **Example:** {clicked “smartphone”} → {clicked “smartphone accessories”}
5	Education	Student behavior analysis	Finding patterns in course enrollment to improve course offerings and planning **Example:** {math101, cs101} → {stat101}
6	Education	Learning paths	Identifying typical learning sequences to design better curricula and study plans **Example:** {topic A, topic B} → {topic C}
7	Telecommunications	Customer churn analysis	Detecting usage patterns linked to cancellations to target retention campaigns **Example:** {low data usage, few calls} → {churn}
8	Telecommunications	Service bundling	Finding services often bought together to create attractive bundles and offers **Example:** {broadband, mobile} → {TV subscription}
9	Banking & Finance	Fraud detection	Uncovering transaction patterns linked to fraud to trigger alerts and investigations **Example:** {high transaction frequency, odd hours} → {fraud}
10	Banking & Finance	Loan approvals	Identifying traits of successful loans to refine credit policies and scoring models **Example:** {high income, good credit score} → {loan approved}
11	Manufacturing	Fault detection	Linking machine conditions to failures for early warning and preventive maintenance **Example:** {high temperature, low pressure} → {equipment failure}
12	Manufacturing	Supply chain optimization	Discovering material usage patterns to optimize inventory and production planning

#	Domain	Use Case	Description (what + why it’s useful)
			Example: {material A, material B} → {product C}
13	Retail (beyond basket)	Shelf placement	Grouping products often bought together to improve store layout and increase sales **Example:** {item A, item B} → {co-located on shelf}
14	Retail (beyond basket)	Customer segmentation	Finding purchase behavior patterns to tailor promotions to specific customer segments **Example:** {frequent discount purchases} → {low brand loyalty}
15	Social Media Analysis	Trending topics	Finding hashtag co-occurrences to understand themes and plan relevant content **Example:** {#climatechange, #sustainability} → {#renewableenergy}
16	Social Media Analysis	User behavior patterns	Analyzing engagement sequences to optimize content for likes, comments, and shares **Example:** {likes post, comments on post} → {shares post}
17	Energy Sector	Usage patterns	Linking context (time, devices) to consumption peaks to improve demand forecasting **Example:** {high A/C usage, weekend} → {peak energy consumption}
18	Energy Sector	Smart grid analysis	Detecting conditions before failures to improve grid reliability and maintenance **Example:** {low voltage, high demand} → {power outage}
19	Transportation & Logistics	Traffic analysis	Discovering conditions that cause congestion to improve planning and routing **Example:** {morning rush hour, bad weather} → {traffic jam}
20	Transportation & Logistics	Route optimization	Finding route combinations tied to faster delivery to optimize logistics operations **Example:** {route A, route B} → {delivered faster}
21	E-commerce	User preferences	Identifying viewing/buying patterns to personalize recommendations and UX **Example:** {view item A} → {view similar item B}
22	E-commerce	Cross-selling	Finding items frequently bought together to suggest add-ons and increase basket value **Example:** {bought laptop} → {bought laptop bag}

#	Domain	Use Case	Description (what + why it's useful)
23	Sports Analytics	Performance metrics	Linking player actions to outcomes to guide tactics and training **Example:** {high possession, accurate passes} → {win}
24	Sports Analytics	Injury prevention	Identifying conditions before injuries to adjust training load and reduce risk **Example:** {high training load, lack of rest} → {injury risk}

Approaches to Association Rule Mining

To discover meaningful patterns such as items frequently bought together, we need methods that can search through large transactional datasets efficiently. A straightforward exhaustive search is theoretically possible but becomes computationally explosive as the number of items grows. Practical association rule mining therefore relies on specialized algorithms designed to avoid this combinatorial blow-up.

- **Brute force:**
  - Enumerate all itemsets and all possible rules
  - Compute support and confidence for each
  - Grows exponentially; infeasible for real datasets
- **Efficient algorithms:**
  - **Apriori** – level-wise search with pruning
  - **FP-Growth** – compresses data into FP-tree, avoids candidate generation
  - **Eclat** – uses vertical data format and intersections
  - All reduce the search space dramatically compared to brute force

The Apriori Algorithm

Apriori is a classic algorithm for **finding frequent itemsets** and **deriving association rules** in transactional databases (e.g., shopping baskets, web logs). It is based on the idea that **every subset of a frequent itemset must also be frequent**.

Key points:

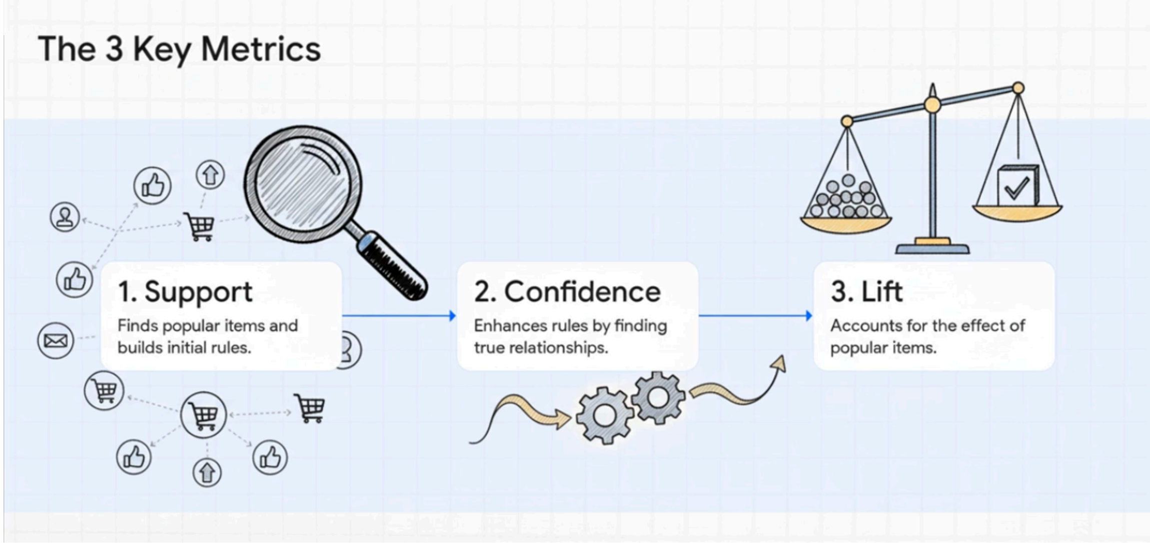
- Proposed by **Agrawal & Srikant (1994)**.
- Works on **transaction data** (e.g., sets of items bought together).
- First finds **frequent single items**, then **grows** them into larger itemsets as long as they remain frequent.

- Uses a **bottom-up, level-wise** search with **candidate generation** and testing.
- Stops when **no more frequent extensions** can be found.
- The resulting frequent itemsets are used to build **association rules** (e.g., for **market basket analysis**).

Creating Association Rules using The Apriori Algorithm

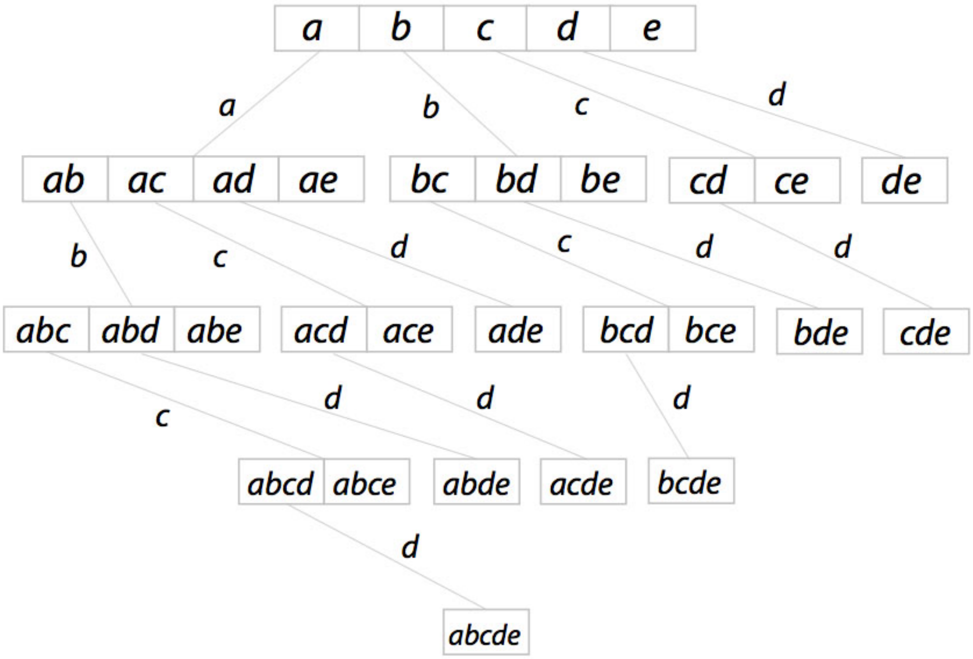
To create association rules, we use 3 metrics:

1. Support: finds the popular items and builds initial rules.
2. Confidence: Enhance initial rules by finding true relationships.
3. Lift: Enhance rules by acomodating for popular items effect (x -> y where x is popular e.g. bread in a bakery)


The 3 Key Metrics
1. Support
Finds popular items and builds initial rules.
2. Confidence
Enhances rules by finding true relationships.
3. Lift
Accounts for the effect of popular items.

The Apriori Algorithm in a NutShell

The Setup: Imagine you have a database of transactions, and you want to discover which items are frequently bought together. You have 5 items: a, b, c, d, and e.



Level 1 - The Beginning: The algorithm starts at the top, scanning the database to count each individual item: {a}, {b}, {c}, {d}, {e}.

Let's say you set a minimum support threshold of 40%. Suppose item {e} is infrequent—it gets pruned.

Level 2 - Building Pairs: The algorithm generates all possible pairs from the surviving items: {ab}, {ac}, {ad}, {bc}, {bd}, {cd}.

Notice: {ae}, {be}, {ce}, {de} are **not generated** because {e} was already eliminated. This saves computational effort!

The algorithm scans the database again. Say {be} and {ce} would have been checked if {e} survived, but they weren't even considered. All other pairs pass the threshold.

Level 3 - Building Triplets: Now here's where it gets interesting. The algorithm generates all 3-itemsets from frequent 2-itemsets: {abc}, **{abd}**, {abe}, {acd}, {ace}, {ade}, {bcd}, {bce}, {bde}, {cde}.

Wait—{abe}, {bce}, {bde} contain {e}, which was infrequent at Level 1, so they're **not generated**. The actual candidates are: {abc}, **{abd}**, {acd}, {ade}, {bcd}, {cde}.

After scanning, suppose {abd} and {bcd} are infrequent—they get pruned.

Level 4 - Narrowing Further: To generate a 4-itemset, all its 3-subsets must be frequent. For example:

- {abcd} needs {abc}, {abd}, {acd}, {bcd} all to be frequent
- But {abd} and {bcd} were infrequent, so **{abcd} cannot be generated**

This is the power of Apriori: entire branches get cut off.

The Result: The algorithm efficiently navigates downward, pruning branches where any subset failed the support threshold, dramatically reducing the search space.

Apriori in Detail

1. The Support Measure

Support measures **how often** an itemset appears in the dataset.
It is a probability-like measure between 0 and 1.

Definition

$$\text{Support}(\text{X}) = \frac{\text{Number of transactions containing X}}{\text{Total transactions}}$$

Range

- 0 ≤ Support(X) ≤ 1
  - 0 → X never appears
  - 1 → X appears in every transaction

Interpretation

- **High support** → X is common, good candidate for rules
- **Low support** → X is rare, often less useful (unless domain-specific)

Thresholds in practice

- Set a **minimum support** to filter out infrequent itemsets.
- Typical starting points:
  - **5–10%** for large retail datasets
  - **20–30%** for smaller datasets or only very popular combinations
- Too high → you miss interesting but less frequent patterns
- Too low → you keep many unimportant, noisy patterns

Example

Let’s assume that we have a small dataset with 12 transactions:

Transaction ID	Item 1	Item 2	Item 3	Item 4	Item 5
1	Milk	Egg	Bread	Butter	
2	Milk	Butter	Egg	Ketchup	
3	Bread	Butter	Ketchup		
4	Milk	Bread	Butter		
5	Bread	Butter	Cookies		
6	Milk	Bread	Butter	Cookies	
7	Milk	Cookies			
8	Milk	Bread	Butter		
9	Bread	Butter	Egg	Cookies	
10	Milk	Butter	Bread		
11	Milk	Bread	Butter		
12	Milk	Bread	Cookies	Ketchup	

First, we set out the **Minimum Support** value to:

- 50% (focus on items present in at least half of the transactions), this value can be set to 5% or 10% in real-life situations with large datasets.

Step 1: Dataset Transformation Convert the dataset into a binary format:

Transaction	Milk	Egg	Bread	Butter	Ketchup	Cookies
1	1	1	1	1	0	0
2	1	1	0	1	1	0
3	0	0	1	1	1	0
4	1	0	1	1	0	0
5	0	0	1	1	0	1
6	1	0	1	1	0	1
7	1	0	0	0	0	1
8	1	0	1	1	0	0
9	0	1	1	1	0	1
10	1	0	1	1	0	0
11	1	0	1	1	0	0
12	1	0	1	0	1	1

Step 2: Identify Frequent 1-Itemsets

The algorithm starts by calculating the support for each single item.

Item	Support Count	Support (%)	Frequent?
Milk	9	75%	Yes
Egg	3	25%	No
Bread	10	83%	Yes
Butter	10	83%	Yes
Ketchup	3	25%	No
Cookies	5	42%	No

Frequent 1-itemsets: `{Milk}`, `{Bread}`, `{Butter}`.

Step 3: Generate 2-Itemsets

Now, combine frequent 1-itemsets into 2-itemsets and calculate their support.

Itemset	Support Count	Support (%)	Frequent?
{Milk, Bread}	7	58%	Yes
{Milk, Butter}	7	58%	Yes
{Bread, Butter}	9	75%	Yes

Frequent 2-itemsets: `{Milk, Bread}`, `{Milk, Butter}`, `{Bread, Butter}`.

Step 4: Generate 3-Itemsets

Combine frequent 2-itemsets into 3-itemsets and calculate their support.

Itemset	Support Count	Support (%)	Frequent?
{Milk, Bread, Butter}	6	50%	Yes

Frequent 3-itemset: `{Milk, Bread, Butter}`.

Step 5: No Larger Itemsets Can Be Created

Since there are no frequent 4-itemsets (support would drop below 50%), we stop here.

Step 6: Generate Initial Association Rules

Now, use the frequent itemsets to generate association rules, we will focus on the largest item set, we can create rules from smaller ones if needed.

Rules from `{Milk, Bread, Butter}`:

1. **Rule 1**:

$$\text{Milk, Bread} \rightarrow \text{Butter}$$

$$Support = P(\text{Milk, Bread, Butter}) = 6/12 = 50\%$$

1. **Rule 2**:

$$\text{Milk, Butter} \rightarrow \text{Bread}$$

$$Support = P(\text{Milk, Bread, Butter}) = 6/12 = 50\%$$

2. **Rule 3**:

$$\text{Bread, Butter} \rightarrow \text{Milk}$$

$$Support = P(\text{Milk, Bread, Butter}) = 6/12 = 50\%$$

Why Support Is Not Enough (SmartBuy Example)

Support is **symmetric**, it does not give us the direction of the relation, it tells us **how often they appear together**, but not **which direction** is more useful.

For example, assume SmartBuy has **1,000 transactions** in a certain period, and we observe:

- 200 transactions include a **Phone**
- 800 transactions include a **Cover**
- 120 transactions include **both** a Phone and a Cover

So the **support** of the pair is:

$\text{support}(\text{Phone} \cap \text{Cover}) = \frac{120}{1000} = 12$

This 12% is the same for both:

- Phone → Cover
- Cover → Phone

Support alone just says: *"Phones and covers occur together in 12% of all SmartBuy transactions."* It does **not** tell us which rule is actually useful.

Direction 1: Phone → Cover (Makes Sense)

$\text{confidence}(\text{Phone} \rightarrow \text{Cover}) = \frac{\text{support}(\text{Phone}\cap\text{Cover})}{\text{support}(\text{Phone})} = \frac{120/1000}{200/1000} = \frac{120}{200} = 60$

Interpretation:

> In **60%** of transactions where a customer buys a **phone**, they also buy a **cover**.

This is a **strong rule** and very reasonable: If someone is buying a phone, suggesting a cover is a good cross-sell.

Direction 2: Cover → Phone (Much Weaker)

$$\text{confidence}(\text{Cover} \to \text{Phone}) = \frac{\text{support}(\text{Phone} \cap \text{Cover})}{\text{support}(\text{Cover})} = \frac{120/1000}{800/1000} = \frac{120}{800} = 15$$

Interpretation:

> In only **15%** of transactions where a customer buys a **cover**, they also buy a **phone**.

This is **quite low**. Most people buying covers already **have** a phone, so suggesting a new phone when they buy a cover is usually **not useful**.

Takeaway

- Same support for the pair: $\text{support}(\text{Phone} \cap \text{Cover}) = 12$
- Very different confidences:
  - Phone → Cover: **60%** (strong, useful rule)
  - Cover → Phone: **15%** (weak, not very useful)

So even though **support is the same**, only

$$\text{Phone} \to \text{Cover}$$

really makes sense as a business rule; the reverse direction does not.

2. TheConfidence Measure

To choose the more useful rule, we use **confidence**.

Definition

Confidence measures how often (Y) appears when (X) appears (how reliable the rule is):

$$\text{conf}(X \to Y) = \frac{\text{support}(X \cap Y)}{\text{support}(X)}.$$

- **Range:** (0 $\leq \text{conf}(X \to Y) \leq$ 1)
  - 0 → the rule never holds
  - 1 → the rule always holds

Interpretation

- **High confidence (~1):** when (X) happens, (Y) almost always happens (strong rule).
- **Low confidence (~0):** when (X) happens, (Y) rarely happens (weak rule).

In practice, we often set a **minimum confidence threshold** (e.g., 70–80%) and keep only rules above that value.

....

Step 7: Validate the Rules using Confidence

First, let us set the **Minimum Confidence** value to: 70%.

Rules from `{Milk, Bread, Butter}`:

1. **Rule 1**:

$$\text{Milk, Bread} \to \text{Butter}$$

$$Confidence = P(\text{Butter}|\text{Milk, Bread}) = \frac{P(\text{Milk, Bread, Butter})}{P(\text{Milk, Bread})} = \frac{6}{7} = 85\%$$

1. **Rule 2**:

$$\text{Milk, Butter} \to \text{Bread}$$

$$Confidence = P(\text{Bread}|\text{Milk, Butter}) = \frac{P(\text{Milk, Bread, Butter})}{P(\text{Milk, Butter})} = \frac{6}{7} = 85\%$$

1. **Rule 3**:

$$\text{Bread, Butter} \to \text{Milk}$$

$$Confidence = P(\text{Milk}|\text{Bread, Butter}) = \frac{P(\text{Milk, Bread, Butter})}{P(\text{Bread, Butter})} = \frac{6}{9} \approx 66\%$$

3. The Lift Measure

Confidence alone can be **misleading** when the consequent (Y) is very frequent: a rule (X → Y) may have high confidence just because (Y) is common.

Given the same data:

- (P(\text{bread}) = 0.80)
- (P(\text{cake}) = 0.20)
- (P(\text{cake} \cap \text{bread}) = 0.16)

Rule

$\text{bread} \to \text{cake}$

Confidence

$$\text{conf}(\text{bread} \to \text{cake}) = P(\text{cake} \mid \text{bread}) = \frac{P(\text{cake} \cap \text{bread})}{P(\text{bread})} = \frac{0.16}{0.80} = 0.20$$

Interpretation: 20% of customers who buy bread also buy cake.

Why confidence alone is misleading

Bread is extremely common:

$$P(\text{cake}) = 0.20$$

So a confidence of 20% is **exactly what we would expect by chance**, even if buying bread has no effect on buying cake.

Lift

$$\text{Lift}(\text{bread} \to \text{cake}) = \frac{P(\text{cake}|\text{bread})}{P(\text{cake})} = \frac{0.20}{0.20} = 1$$

interpretation

> **Do we buy cake more often when we buy bread than we do on average?**

Answer: **No.** Lift = 1 means buying bread does **not** increase or decrease the likelihood of buying cake.

Intuition summary

Metric	Meaning in this example
Support(cake)	% of all customers who buy cake
Confidence	% of bread buyers who also buy cake
Lift	Comparison of those two probabilities

Because:

$$P(\text{cake} \mid \text{bread}) = P(\text{cake})$$

bread and cake are **statistically independent**.

مقياس الرفع (Lift)

مقياس الرفع (Lift) عمليًا يقيس ما إذا كان **احتمال شراء الكيك عند شراء الخبز** أكبر من **احتمال شراء الكيك بشكل عام**.

أو بصيغة أدق:

> يقيس مقياس الرفع ما إذا كان شراء الخبز يزيد من احتمال شراء الكيك مقارنةً بالمعدل العام لشراء الكيك.

....

> **Interpreting Lift**

Lift measures how strongly two items or itemsets are associated:

- $(\text{lift} > 1)$: items occur together **more often than expected** (positive association).
- $(\text{lift} = 1)$: items occur together **as often as expected** under independence (no association).
- $(\text{lift} < 1)$: items occur together **less often than expected** (negative association).

So lift tells us whether items co-occur more (or less) frequently **than expected under independence**, making it a useful complement to confidence.

....

Step 8: Validate Rules using the Lift Measure

First, we set the **Lift > 1** for actionable rules

1. **Rule 1**:

$$\text{Milk, Bread} \rightarrow \text{Butter}$$

$$\text{Lift} = \frac{P(\text{Butter}|\text{Milk, Bread})}{P(\text{Butter})} = \frac{0.85}{0.83} \approx 1.02$$

2. **Rule 2**:

$$\text{Milk, Butter} \rightarrow \text{Bread}$$

$$\text{Lift} = \frac{P(\text{Bread}|\text{Milk, Butter})}{P(\text{Bread})} = \frac{0.85}{0.83} \approx 1.02$$

3. **Rule 3**:

$$\text{Bread, Butter} \rightarrow \text{Milk}$$

$$\text{Lift} = \frac{P(\text{Milk}|\text{Bread, Butter})}{P(\text{Milk})} = \frac{0.66}{0.75} \approx .88$$

Final Results

Frequent Itemsets:

- **1-itemsets**: `{Milk}`, `{Bread}`, `{Butter}`
- **2-itemsets**: `{Milk, Bread}`, `{Milk, Butter}`, `{Bread, Butter}`
- **3-itemset**: `{Milk, Bread, Butter}`

Rules:

Rule	Support	Confidence	Lift	Actionable?
$\text{Milk, Bread} \rightarrow \text{Butter}$	50%	85%	1.02	Yes
$\text{Milk, Butter} \rightarrow \text{Bread}$	50%	85%	1.02	Yes
$\text{Bread, Butter} \rightarrow \text{Milk}$	50%	66%	.88	No

Interpretation and Actionable Insights

Rule	Explaniation
$\text{Milk, Bread} \rightarrow \text{Butter}$	Customers buying Milk and Bread are highly likely (85%) to also buy Butter. Consider bundling these items.
$\text{Milk, Butter} \rightarrow \text{Bread}$	Strong association; placing these items together could increase sales.
$\text{Bread, Butter} \rightarrow \text{Milk}$	Suggests that Milk is a complementary product to Bread and Butter, Weak lift thought

Python Example

We will implement the **Apriori algorithm** using the `mlxtend` library for frequent itemset mining and rule generation.

a. Install Required Libraries Make sure you have the required libraries installed:

```
pip install pandas mlxtend
```

b. Load the Dataset We will represent the dataset as a **binary transaction matrix** where each row is a transaction, and each column is an item (1 = purchased, 0 = not purchased).

In [1]:

```python
import pandas as pd

# Define the dataset
data = {
    "Milk":    [1, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1],
    "Egg":     [1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0],
    "Bread":   [1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1],
    "Butter":  [1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0],
    "Ketchup": [0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1],
    "Cookies": [0, 0, 0, 0, 1, 1, 1, 0, 1, 0, 0, 1],
}

# Create a DataFrame
df = pd.DataFrame(data)
print("Transaction Dataset:")
df
```

```
Transaction Dataset:
```

Out[1]:

	Milk	Egg	Bread	Butter	Ketchup	Cookies
0	1	1	1	1	0	0
1	1	1	0	1	1	0
2	0	0	1	1	1	0
3	1	0	1	1	0	0
4	0	0	1	1	0	1
5	1	0	1	1	0	1
6	1	0	0	0	0	1
7	1	0	1	1	0	0
8	0	1	1	1	0	1
9	1	0	1	1	0	0
10	1	0	1	1	0	0
11	1	0	1	0	1	1

c. Apply the Apriori Algorithm We will use the `mlxtend` library to find frequent itemsets and generate association rules.

In [3]:

```
! pip install mlxtend
```

```
Collecting mlxtend
  Downloading mlxtend-0.23.4-py3-none-any.whl.metadata (7.3 kB)
Requirement already satisfied: scipy>=1.2.1 in /home/me/myenv/lib/python3.12/site-
packages (from mlxtend) (1.13.1)
Requirement already satisfied: numpy>=1.16.2 in /home/me/myenv/lib/python3.12/site
-packages (from mlxtend) (1.26.4)
Requirement already satisfied: pandas>=0.24.2 in /home/me/myenv/lib/python3.12/sit
e-packages (from mlxtend) (2.2.3)
Requirement already satisfied: scikit-learn>=1.3.1 in /home/me/myenv/lib/python3.1
2/site-packages (from mlxtend) (1.5.2)
Requirement already satisfied: matplotlib>=3.0.0 in /home/me/myenv/lib/python3.12/
site-packages (from mlxtend) (3.9.2)
Requirement already satisfied: joblib>=0.13.2 in /home/me/myenv/lib/python3.12/sit
e-packages (from mlxtend) (1.4.2)
Requirement already satisfied: contourpy>=1.0.1 in /home/me/myenv/lib/python3.12/s
ite-packages (from matplotlib>=3.0.0->mlxtend) (1.3.1)
Requirement already satisfied: cycler>=0.10 in /home/me/myenv/lib/python3.12/site-
packages (from matplotlib>=3.0.0->mlxtend) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /home/me/myenv/lib/python3.12/
site-packages (from matplotlib>=3.0.0->mlxtend) (4.54.1)
Requirement already satisfied: kiwisolver>=1.3.1 in /home/me/myenv/lib/python3.12/
site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.7)
Requirement already satisfied: packaging>=20.0 in /home/me/myenv/lib/python3.12/si
te-packages (from matplotlib>=3.0.0->mlxtend) (24.2)
Requirement already satisfied: pillow>=8 in /home/me/myenv/lib/python3.12/site-pac
kages (from matplotlib>=3.0.0->mlxtend) (11.0.0)
Requirement already satisfied: pyparsing>=2.3.1 in /home/me/myenv/lib/python3.12/s
ite-packages (from matplotlib>=3.0.0->mlxtend) (3.2.0)
Requirement already satisfied: python-dateutil>=2.7 in /home/me/myenv/lib/python3.
12/site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /home/me/myenv/lib/python3.12/site-
packages (from pandas>=0.24.2->mlxtend) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /home/me/myenv/lib/python3.12/sit
e-packages (from pandas>=0.24.2->mlxtend) (2024.2)
Requirement already satisfied: threadpoolctl>=3.1.0 in /home/me/myenv/lib/python3.
12/site-packages (from scikit-learn>=1.3.1->mlxtend) (3.5.0)
Requirement already satisfied: six>=1.5 in /home/me/myenv/lib/python3.12/site-pack
ages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)
Downloading mlxtend-0.23.4-py3-none-any.whl (1.4 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.4/1.4 MB 5.5 MB/s eta 0:00:00[36m0:0
0:01m eta 0:00:01
Installing collected packages: mlxtend
Successfully installed mlxtend-0.23.4
```

In [4]:

```python
from mlxtend.frequent_patterns import apriori, association_rules

# Step 1: Generate frequent itemsets with Apriori
frequent_itemsets = apriori(df, min_support=0.5, use_colnames=True)

# Display frequent itemsets
print("\nFrequent Itemsets:")
print(frequent_itemsets)
```

```
Frequent Itemsets:
    support               itemsets
0  0.750000                 (Milk)
1  0.833333                (Bread)
2  0.833333               (Butter)
3  0.583333          (Bread, Milk)
4  0.583333         (Milk, Butter)
5  0.750000        (Bread, Butter)
6  0.500000  (Bread, Milk, Butter)
```

```
/home/me/myenv/lib/python3.12/site-packages/mlxtend/frequent_patterns/fpcommon.py:
161: DeprecationWarning: DataFrames with non-bool types result in worse computatio
nalperformance and their support might be discontinued in the future.Please use a
DataFrame with bool type
  warnings.warn(
```

d. Generate Association Rules We generate rules based on **confidence** and calculate **lift**.

Generate rules based on confidence

In [ ]:

```
# Step 2: Generate association rules based on confidence
rules = association_rules(frequent_itemsets, num_itemsets=len(df), metric="confide

# Display the rules
print("\nAssociation Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
Association Rules:
      antecedents consequents   support  confidence      lift
0         (Bread)      (Milk)  0.583333    0.700000  0.933333
1          (Milk)     (Bread)  0.583333    0.777778  0.933333
2        (Butter)      (Milk)  0.583333    0.700000  0.933333
3          (Milk)    (Butter)  0.583333    0.777778  0.933333
4        (Butter)     (Bread)  0.750000    0.900000  1.080000
5         (Bread)    (Butter)  0.750000    0.900000  1.080000
6   (Milk, Bread)    (Butter)  0.500000    0.857143  1.028571
7  (Butter, Milk)     (Bread)  0.500000    0.857143  1.028571
```

We can also generate rules based on lift

In [ ]:

```
# Step 2: Generate association rules based on lift
rules = association_rules(frequent_itemsets, num_itemsets=len(df), metric="lift",

# Display the rules
print("\nAssociation Rules:")
print(rules[['antecedents', 'consequents', 'support', 'confidence', 'lift']])
```

```
Association Rules:
      antecedents     consequents  support  confidence      lift
0        (Butter)         (Bread)     0.75    0.900000  1.080000
1         (Bread)        (Butter)     0.75    0.900000  1.080000
2   (Milk, Bread)        (Butter)     0.50    0.857143  1.028571
3  (Butter, Milk)         (Bread)     0.50    0.857143  1.028571
4         (Bread)  (Butter, Milk)     0.50    0.600000  1.028571
5        (Butter)   (Milk, Bread)     0.50    0.600000  1.028571
```

e. Filter and Interpret Rules You can filter rules for high lift or specific itemsets.

In [ ]:

```
# Filter rules with Lift > 1
filtered_rules = rules[rules['lift'] > 1]
print("\nFiltered Rules with Lift > 1:")
print(filtered_rules[['antecedents', 'consequents', 'support', 'confidence', 'lift
```

```
Filtered Rules with Lift > 1:
      antecedents      consequents  support  confidence      lift
0        (Butter)          (Bread)     0.75    0.900000  1.080000
1         (Bread)         (Butter)     0.75    0.900000  1.080000
2   (Milk, Bread)         (Butter)     0.50    0.857143  1.028571
3  (Butter, Milk)          (Bread)     0.50    0.857143  1.028571
4         (Bread)   (Butter, Milk)     0.50    0.600000  1.028571
5        (Butter)    (Milk, Bread)     0.50    0.600000  1.028571
```

Interpretation

1. $\text{Milk, Bread} \rightarrow \text{Butter}$
   - Support = 58%, Confidence = 87.5%, Lift = 1.05.
   - Suggest bundling these items for promotions.
2. $\text{Milk, Butter} \rightarrow \text{Bread}$
   - Similar metrics as above, showing strong relationships.
3. $\text{Bread, Butter} \rightarrow \text{Milk}$
   - High confidence but slightly weaker lift, still actionable.

Two-Items Rules

If desired, we can focus on association rules where the antecedent (X) contains just **one item** (i.e., 1-item antecedent, 2 items in the rule overall).

Consider the rule:

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

Suppose we have **12 transactions**, and we observe:

- **Milk** appears in **9** transactions.
- **Milk and Bread together** appear in **7** transactions.

Then:

- **Support** of the rule is the fraction of all transactions that contain both Milk and Bread: $\text{supp}(\text{Milk} \rightarrow \text{Bread}) = \frac{7}{12} \approx 58$

- **Confidence** of the rule is the fraction of Milk-transactions that also contain Bread: $\text{conf}(\text{Milk} \rightarrow \text{Bread}) = \frac{7}{9} \approx 78$

So we can write the rule as:

$\text{Milk} \rightarrow \text{Bread}$ {S = 58%, C = 78%}.

2-Item Rules vs 3-Item Rules

2-Item Rules (e.g., $\text{Milk} \rightarrow \text{Bread}$)

- **Simplicity**: Easy to read, explain, and act on.
- **Higher support and confidence (typically)**: Fewer items need to co-occur, so these rules tend to appear more often in the data.
- **Broad applicability**: Capture general trends such as $\text{Milk} \rightarrow \text{Bread}$, which can inform store layout (placing items nearby) or simple promotions.

3-Item Rules (e.g., $\text{Milk}, \text{Bread} \rightarrow \text{Butter}$)

- **Deeper insights**: Reveal more specific patterns, like $\text{Milk}, \text{Bread} \rightarrow \text{Butter}$ that might not be visible in 2-item rules.
- **Lower support (usually)**: All three items must appear together, so these combinations are naturally less frequent.
- **Targeted application**: Well-suited for niche marketing, personalized recommendations, or special bundle offers.

Which Is Better?

- Use **2-item rules** when you want:
  - General trends,
  - Simple explanations,
  - Broad business strategies (e.g., product placement, popular combos).
- Use **3-item rules** when you want:
  - More specific patterns,
  - Finer-grained understanding of customer behavior,
  - Targeted campaigns or tailored bundles.

There is no universally “better” rule size—the choice depends on your **data characteristics** (how dense/sparse the transactions are) and your **business objectives** (broad policies vs. targeted actions).