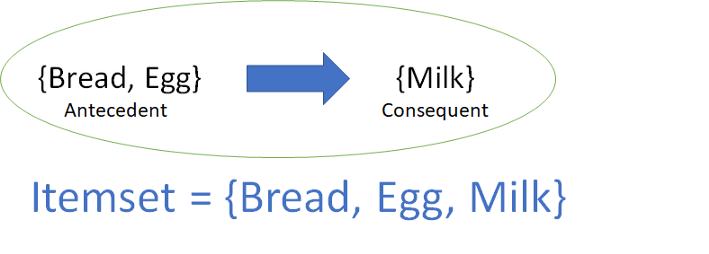
# Association Rule

Association Rules is one of the very important concepts of machine learning being used in market basket analysis. In a store, all vegetables are placed in the same aisle, all dairy items are placed together and cosmetics form another set of such groups. Investing time and resources on deliberate product placements like this not only reduces a customer’s shopping time, but also reminds the customer of what relevant items (s)he might be interested in buying, thus helping stores cross-sell in the process. Association rules help uncover all such relationships between items from huge databases. One important thing to note is-

Rules do not extract an individual’s preference, rather find relationships between set of elements of every distinct transaction. This is what makes them different from collaborative filtering.

Let us now see what an association rule exactly looks like. It consists of an antecedent and a consequent, both of which are a list of items. Note that implication here is co-occurrence and not causality. For a given rule, item set is the list of all the items in the antecedent and the consequent.

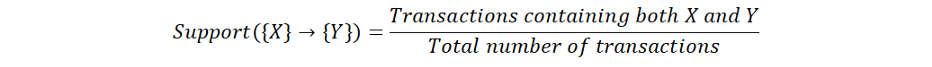


## **Thresholds used for Relations**

Various metrics are in place to help us understand the strength of association between these two.

### Support

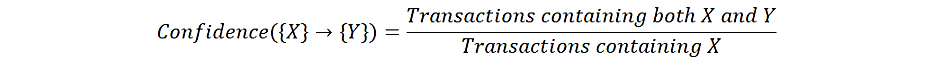
This measure gives an idea of how frequent an itemset is in all the transactions. Consider itemset1 = {bread} and itemset2 = {shampoo}. There will be far more transactions containing bread than those containing shampoo. Therefore, as you rightly guessed, itemset1 will generally have a higher support than itemset2. Now consider itemset1 = {bread, butter} and itemset2 = {bread, shampoo}. Many transactions will have both bread and butter on the cart but bread and shampoo. Not so much. Therefore, in this case, itemset1 will generally have a higher support than itemset2. Mathematically, support is the fraction of the total number of transactions in which the itemset occurs.



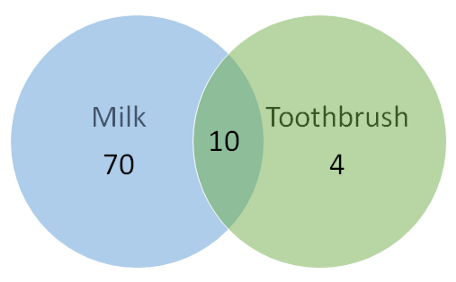
Value of support helps us identify the rules worth considering for further analysis. For example, one might want to consider only the itemsets which occur at least 50 times out of a total of 10,000 transactions i.e. support = 0.005. If an itemset happens to have a very low support, we do not have enough information on the relationship between its items and hence no conclusions can be drawn from such a rule.

### Confidence

This measure defines the likeliness of occurrence of consequent on the cart given that the cart already has the antecedents. That is to answer the question — of all the transactions containing say, {Captain Crunch}, how many also had {Milk} on them? We can say by common knowledge that {Captain Crunch} → {Milk} should be a high confidence rule. Technically, confidence is the conditional probability of occurrence of consequent given the antecedent.



It does not matter what you have in the antecedent for such a frequent consequent. The confidence for an association rule having a very frequent consequent will always be high.



Confidence for {Toothbrush} → {Milk} will be 10/(70) = 0.7

Looks like a high confidence value. But we know intuitively that these two products have a weak association and there is something misleading about this high confidence value. Lift is introduced to overcome this challenge. Considering just the value of confidence limits our capability to make any business inference.

### Lift

Lift controls for the support (frequency) of consequent while calculating the conditional probability of occurrence of {Y} given {X}. Lift is a very literal term given to this measure. Think of it as the \*lift\* that {X} provides to our confidence for having {Y} on the cart. To rephrase, lift is the rise in probability of having {Y} on the cart with the knowledge of {X} being present over the probability of having {Y} on the cart without any knowledge about presence of {X}. Mathematically,

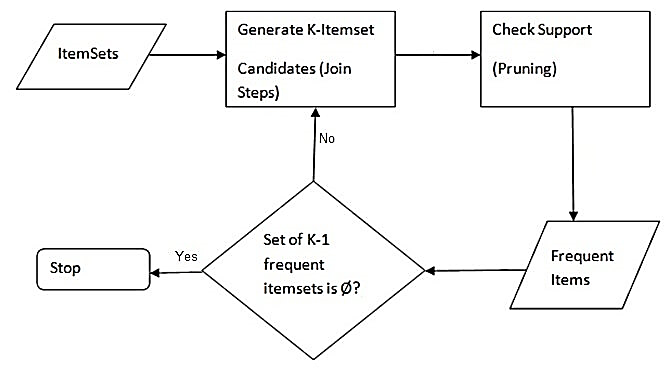
Confidence (X|Y) = Support (XY) / Support (X)

## Apriori algorithm

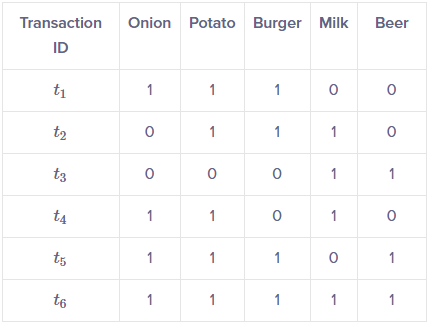
The entire algorithm can be divided into two steps:

* Step 1: Apply minimum support to find all the frequent sets with k items in a database.
* Step 2: Use the self-join rule to find the frequent sets with k+1 items with the help of frequent k-itemsets. Repeat this process from k=1 to the point when we are unable to apply the self-join rule.

This approach of extending a frequent itemset one at a time is called the “bottom up” approach.

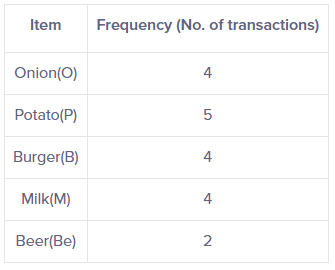


**Example**



Let us now look at the intuitive explanation of the algorithm with the help of the example we used above. Before beginning the process, let us set the support threshold to 50%, i.e. only those items are significant for which support is more than 50%.

**Step 1**: Create a frequency table of all the items that occur in all the transactions. For our case:

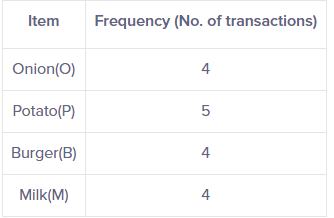


Now



(50/100)\*6 = 3

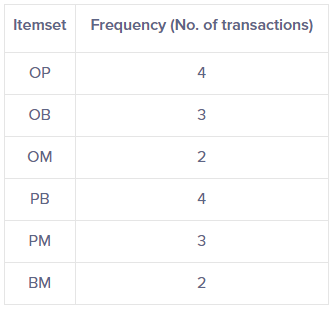
**Step 2**: We know that only those elements are significant for which the support is greater than or equal to the threshold support. Here, support threshold is 50%, hence only those items are significant which occur in more than three transactions and such items are Onion(O), Potato(P), Burger(B), and Milk(M). Therefore, we are left with:



The table above represents the single items that are purchased by the customers frequently.

**Step 3**: The next step is to make all the possible pairs of the significant items keeping in mind that the order doesn’t matter, i.e., AB is same as BA. To do this, take the first item and pair it with all the others such as OP, OB, OM. Similarly, consider the second item and pair it with preceding items, i.e., PB, PM. We are only considering the preceding items because PO (same as OP) already exists. So, all the pairs in our example are OP, OB, OM, PB, PM, BM.

**Step 4**: We will now count the occurrences of each pair in all the transactions.



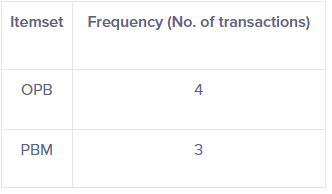
**Step 5**: Again only those itemsets are significant which cross the support threshold, and those are OP, OB, PB, and PM.

**Step 6**: Now let’s say we would like to look for a set of three items that are purchased together. We will use the itemsets found in step 5 and create a set of 3 items.

To create a set of 3 items another rule, called self-join is required. It says that from the item pairs OP, OB, PB and PM we look for two pairs with the identical first letter and so we get

* **O**P and **O**B, this gives OPB
* **P**B and **P**M, this gives PBM

Next, we find the frequency for these two itemsets.



Applying the threshold rule again, we find that OPB is the only significant itemset. Therefore, the set of 3 items that was purchased most frequently is OPB.

Let us look at our previous example to get an efficient association rule. We found that OPB was the frequent itemset. So for this problem, step 1 is already done. So, let’ see step 2. All the possible rules using OPB are:

OP⟶B, OB⟶P, PB⟶O, B⟶ OP, P⟶OB, O⟶PB

Now Compare Confidence with min confidence which we mention earlier which 80% so skip Itemset with less than 80% confidence.