Apriori Algorithm - Ben Hellman

**INTRODUCTION**

Association Rule Learning is the process of mining data for associations between subsets of items. This means finding rules that allow us to deduce “if this set of items is in this set we can reasonably expect this other set of items to be in this set”. This is very abstract so it makes sense to use a concrete example.

The standard example for this is analyzing transactions at a supermarket. A supermarket has a dataset of all of its sales and it can run an Association Rule Learning algorithm to find what things are commonly bought together. For example, it might find that if someone buys toothbrushes they are highly likely to buy floss so it is a good idea to place them next to each other in the store. Also note that these techniques don’t just apply to supermarkets but many data sets can be treated in this way. For example: apps might find content that is rated highly by a group of people and so if somebody watches one video the other video should be recommended.

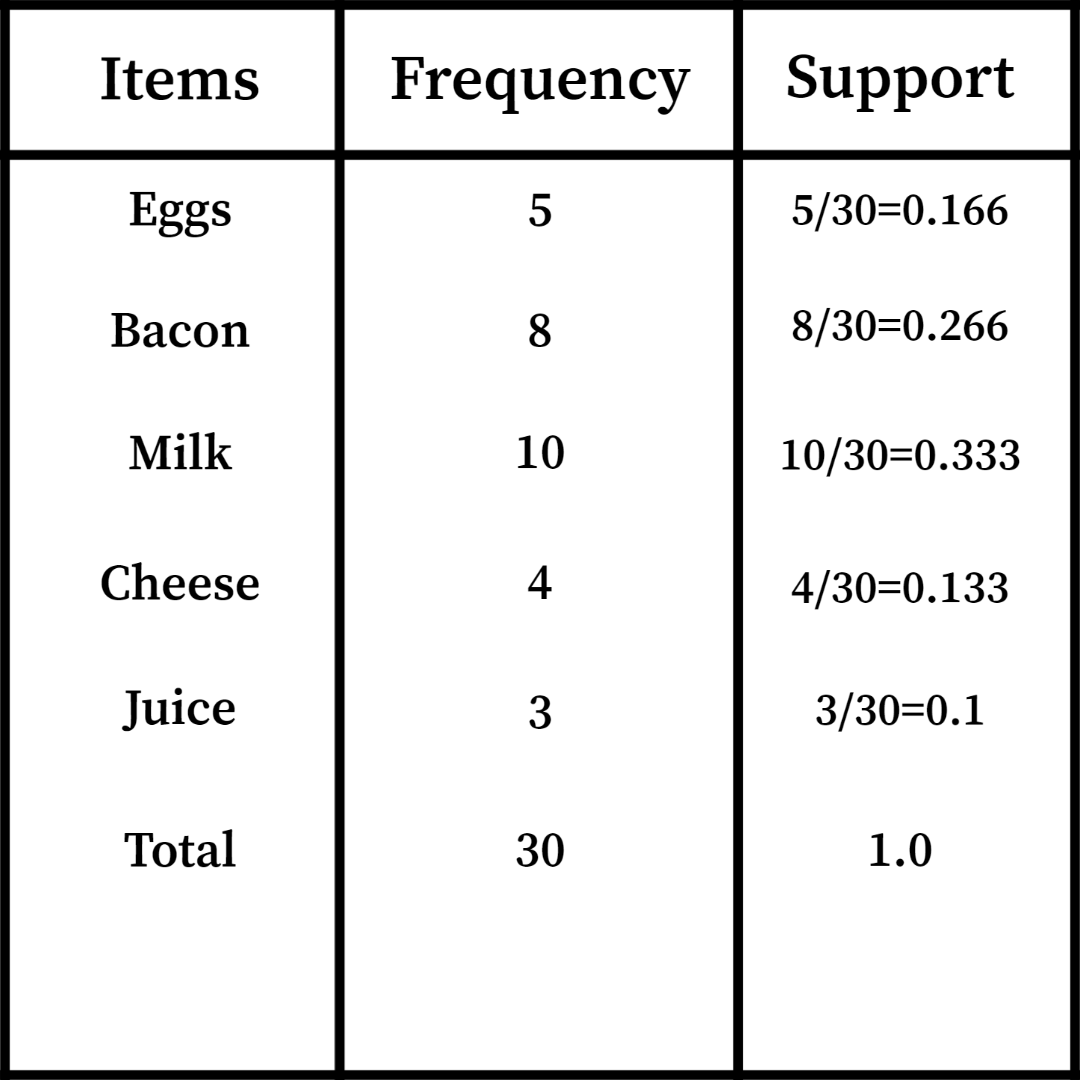
Now we can define some measures that can be used to find associations. For these examples let: set of items I = i1, i2, ..., im set of transactions T = t1, t2, ..., tn, where each transaction is a set of items Our goal is to find rules of the form X → Y , where X and Y are subsets of I, that are considered to be strongly associated within the transactions. Meaning if we find the elements of set X in a transaction we can reasonably expect the set of items in Y to be there too. Going to our supermarket example, the set of items would be everything the grocery store sells and the set of transactions would be receipts from customers buying items.

**BASIC MEASURES**

* Support: This measures how frequently these two items are bought in general it is calculated as the fraction of transactions that have X and Y. To put it simply, support is the number of transactions containing items X and Y divided by the total number of transactions. Going back to the grocery store example, if a store owner wanted to calculate the support of milk, if they had 30 total receipts and 10 of those receipts had milk, then Support(Milk) = 10/30. A result closer to one indicates that the items appear together in the database frequently. In probability notation:

P(X ∈ t ∧ Y ∈ t)

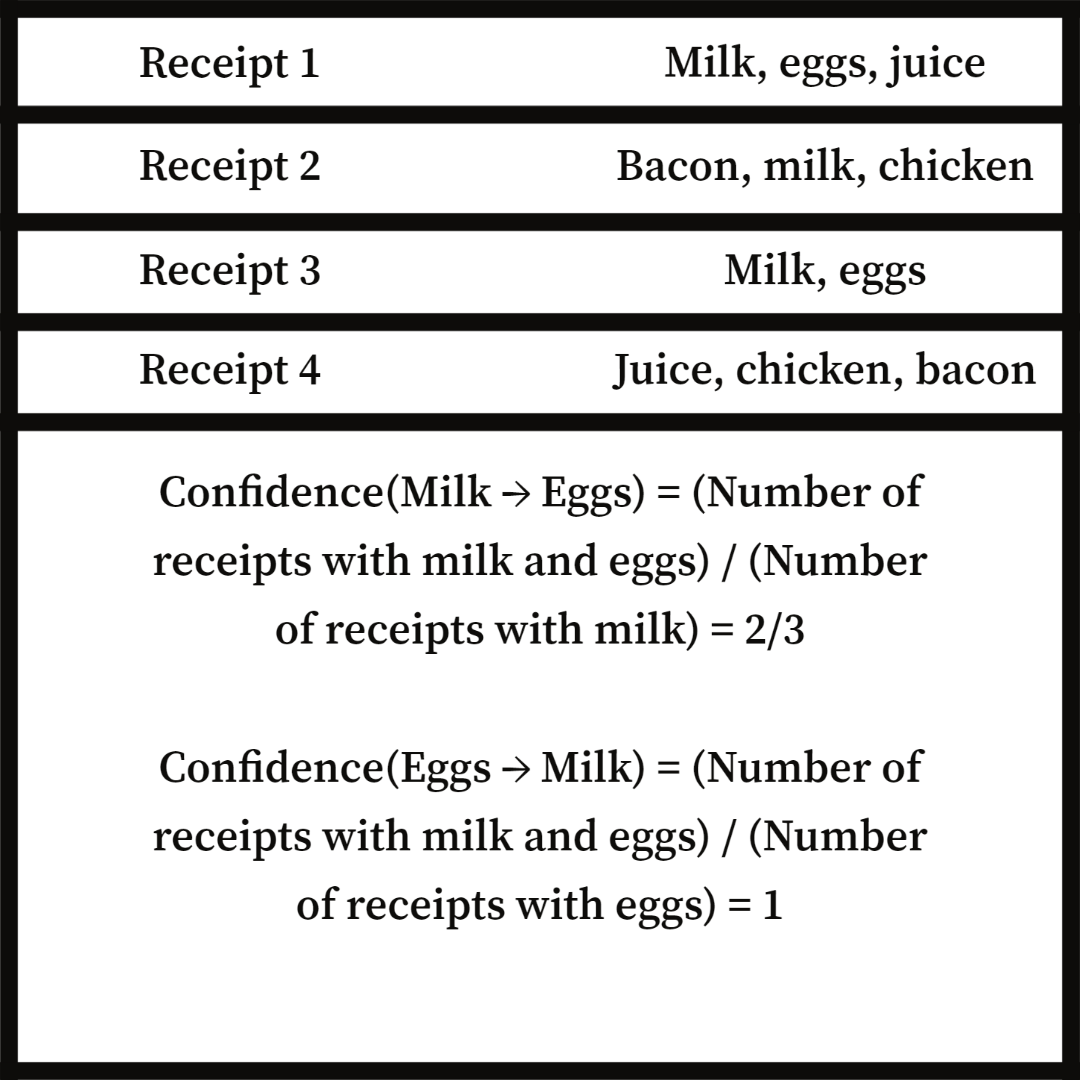
Support(X → Y ) = |t ∈ T | X ⊆ t and Y ⊆ t| / |T|



* Confidence: This measures how confident we can be that if X appears then Y appears. It’s calculated as the fraction of transactions containing X that also contain Y. Note that this is not commutative. This can be simply looked at as the number of transactions that contain X and Y divided by the number of transactions with X. A confidence value close to one can be interpreted as the presence of X is strongly associated with the presence of Y. Below is an example of calculating confidence within the context of a grocery store. In probability notation:

P(Y ∈ t|X ∈ t)

Confidence(X → Y ) = |t ∈ T | X ⊆ t and Y ⊆ t| / |t ∈ T | X ⊆ t|



* Lift: When developing an association X → Y it is helpful to know if X appearing actually makes Y more likely to appear. This is what lift does; it is calculated as the ratio of Confidence to Support. Lift is commutative.

– if lift > 1 that means X appearing makes Y more likely to appear

– if lift = 1 that means it has no affect

– if lift < 1 it means X appearing makes Y less likely to appear.

Lift(X → Y ) = Confidence(X → Y ) / Support(Y )

* Conviction: Measures the ratio between the probability an element B does not appear in general and the probability B does not appear if A is present. Unlike Lift Conviction is directed meaning Conviction from A to B is not equal to Conviction from B to A. Just like lift:

– if conviction > 1 that means X appearing makes Y more likely to appear

– if conviction = 1 that means it has no affect

– if conviction < 1 it means X appearing makes Y less likely to appear.

Conviction(A → B) = (1 − Support(B)) / (1 − Confidence(A → B))

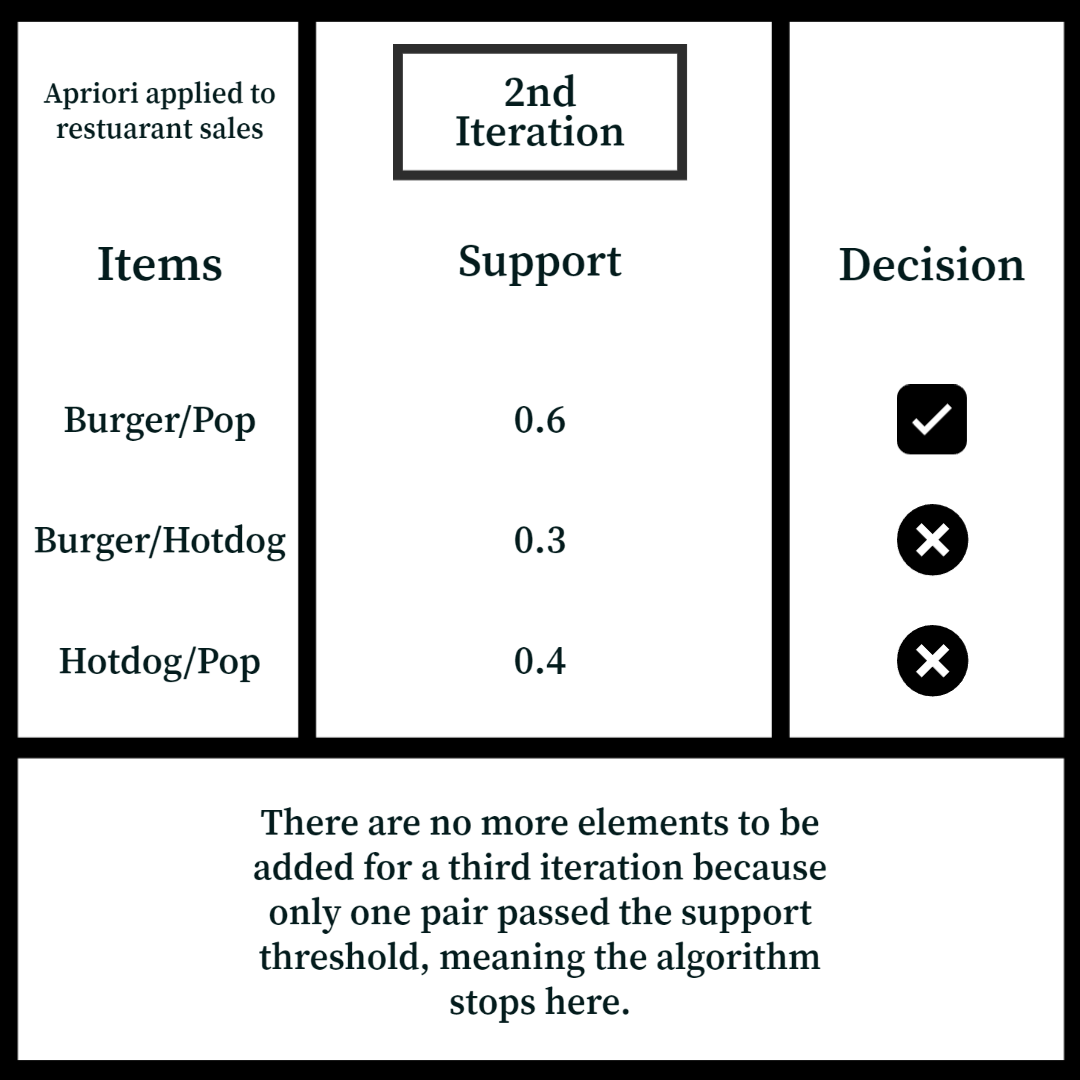
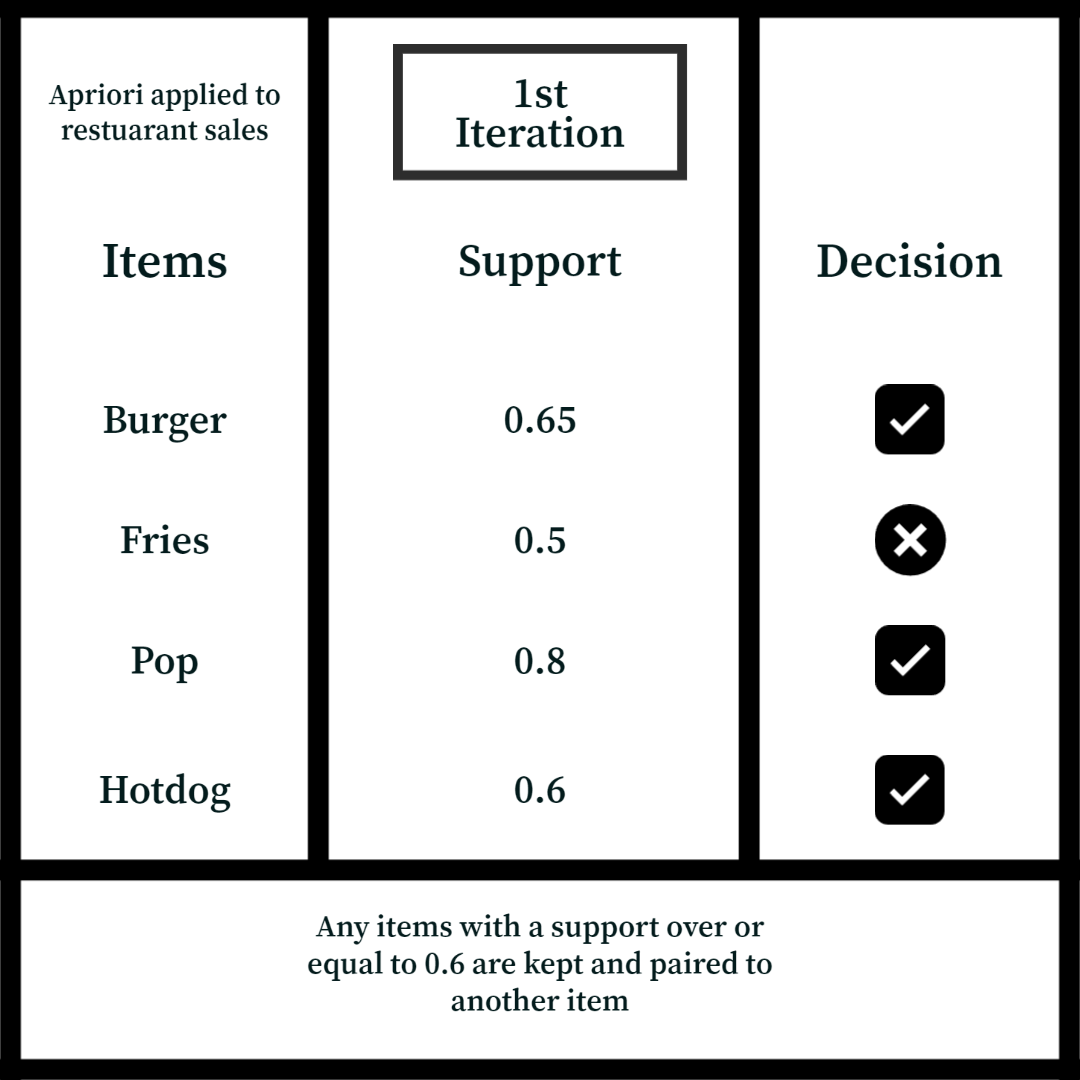
Within all of these measures, we assume that items within the dataset are independent of one another and that the transactions are independent of one another. You can interpret item independence as meaning that none of our items at the grocery store are on sale together. Meaning that buying milk doesn’t make eggs any cheaper. And transaction independence could be interpreted as each transaction belongs to a separate person, or basically that each transaction is a separate occurrence.

These four values allow us to measure associations in a large data set, but these individual measures can’t give us any data without a way to apply them. A store owner can’t apply conviction to every possible combination of items to find correlation. This is where algorithms come in, specifically we will talk about Apriori algorithm which uses support to find correlations in datasets.

**APRIORI ALGORITHM**

Apriori algorithm is an unsupervised algorithm, meaning it is given a dataset and simply operates on it with no idea what the output should be as opposed to a supervised algorithm which must be given human labeled data to operate. It is used for finding commonly occurring sets of items in a dataset, as described in the supermarket example from the previous section. On a high level, the way Apriori works is it first identifies individual items that appear frequently in the dataset, and gradually builds bigger itemsets that appear frequently by adding more items. The algorithm sets a minimum support threshold, with which it decides if items from the previous iteration are continued on to the current iteration. It does this until it reaches a plateau, where there are no more new frequent itemsets that can be found. The resulting itemsets at the end of the algorithm can be used to draw important conclusions. In the first iteration, Apriori will look at the individual items and see if they pass a set support threshold. If they do, then they will move on to the second iteration. In the second iteration, every possible pair of items that passed the first iteration is checked against the support threshold and this continues until no more items can be added. The reason that this works is because adding an extra element to calculate support can only decrease the support value. If the support of milk is 0.5, then the support of milk and eggs is less than or equal to 0.5 because support calculates the amount with both items in the transaction divided by total items. And the amount of transactions with milk and eggs is at most the amount of transactions with milk.

An example of how the algorithm operates on restaurant sales with a support threshold of 0.6 is shown on the below:



**PRACTICE QUESTIONS**

For the following questions, if needed, use this grocery store example dataset:

I = {milk, butter, eggs, bread, chips, apples, lettuce, tomatoes}

T = {{milk, bread, apples}, {eggs, bread}, {eggs, chips, apples, tomatoes}, {milk, eggs, apples}, {butter, lettuce}, {milk, bread, lettuce}

1. Calculate Support(Milk → Bread ):
2. Calculate Confidence(Milk → Bread):
3. What does the difference between these values represent?
4. Calculate Conviction(Apples → Chips) and does this result tell us anything about Apples and Chips being correlated in the data:
5. Use Apriori Algorithm with a support threshold of 0.3 and write down the results, was this a good support threshold to use