# Association Analysis with the Apriori Algorithm

*David Palumbo and Manav Majithia*
*CSE 469: Introduction to Data Mining, Fall 2020*
*State University of New York at Buffalo*

**Introduction**:
We are using transaction data from a grocery store. With this we decided the best data mining problem to approach would be association mining. The goal is to find relationships between frequent items, and this can have many different applications in a real world setting. For example, if a market finds that two items are frequently bought together they can take actions to increase this correlation, such as moving the items physically closer together in the store. This information can also help them develop sales and coupon combinations in the future.

**Formulation**:
To do the association mining we decided to use the apriori algorithm that was the subject of the second class assignment. For this we needed to pre process the data set a little. We removed the column and row headers so that the dataset was just a csv where each row represented one transaction. We also changed the load data to remove the blank entries (line padding) in each row so that it was the same format that we used for Assignment 2. We also had to input a min support, which was difficult to come up with. We mention more about this in the challenges section.

As for the output, the most information was gained with the verbose set to true, where it would output all of the frequent datasets and their support.

**Datasets**:
The dataset was downloaded from this link: https://www.kaggle.com/irfanasrullah/groceries. It contained 9835 transactions with 169 unique items. This is relatively small for a real world example, but it was difficult to find any larger transaction data. As for the preprocessing of the data this was mentioned in the previous section.

**Algorithm**:
We applied the apriori algorithm for association mining. This finds frequent data subsets of size k, and then uses these sets to find frequent data subsets of size (k+1). Other than changing it slightly to preprocess the csv file, it is the same code that we submitted for Assignment 2.

**Experiments**:
Frequent data subsets for (min_sup = 0.02, which is a little less than 200 transactions) can be found in Appendix A.

Of these, the subsets of cardinality greater than 1 are the ones that would convey the information we are looking for. Even then, looking back, there seems to be a major flaw in using

the apriori algorithm for finding item correlations. While it can find frequent sets, it doesn't ensure that these sets show correlation. That is for the following reason:

Assume we have the hypothetical problem with a min_sup = 0.1. Now assume we have an item A with support 0.5 and item B with support 0.5 as well. If A and B are independent items (no correlation) we will still expect a set with A & B will have a support around 0.25 (since P(A) * P(B) = P(A & B) for independent variables). This would be greater than our min support and so itemset A & B would show up in our output even though there is no correlation between the items, simply because they are both items that are bought substantially more than our min support.

To avoid this in the future we could postprocess our data to remove any sets where (support(A) * support(B) + const > support(A & B)). This would essentially remove any sets from the output that show signs of being independent variables (within some constant) since they don't give very much meaningful information to the client. The hard part of this evaluation would be to determine this constant, and would presumably require some testing to find a sweet spot.

Interestingly, in a similar way, one could invert this formula to (support(A) * support(B) - const < support(A & B)). Along with some minor adjustments to the algorithm, this could return any item sets that have negative correlation (i.e. items that are bought together less than would be assumed by random chance). This could presumably be just as important information to a store as positive correlation since they could take efforts to increase correlation and get consumers to buy products from more sections of the store in one shopping trip.

As for the actual evacuation of the apriori algorithm that we used; I selected a random sample of 10 2-item frequent subsets and their corresponding 1-item subsets. With this I did the calculations talked about above, and while all of the support(A & B) > support(A) * support(B) for these samples, they were by varying amounts. This means that some of them presumably showed correlation, while others may not have. This is where one would have to look to try and come up with a proper constant for post processing. It is hard to find any figure to represent the accuracy unless we have this constant to determine exactly what proportion of our output depicts item correlation.

**Challenges**:
The main challenge was choosing a proper min_sup. The dataset is large and has many items, so there are few items that are able to reach a large support. But if the support is lowered too much the output can be altered by outliers. It can also result in really long runtimes for large datasets, though ours was not that large. For our test we decided that 0.02 was a decent spot. It was a bit low and presumably had some bad data but we decided that this was better than choosing a min_sup that was too high resulting in losing some of the meaningful data potentially.

**Appendix A:**

{UHT-milk}:  sup = 0.033
{beef}:  sup = 0.052
{berries}:  sup = 0.033
{beverages}:  sup = 0.026
{bottled beer}:  sup = 0.081
{bottled water}:  sup = 0.111
{brown bread}:  sup = 0.065
{butter}:  sup = 0.055
{butter milk}:  sup = 0.028
{candy}:  sup = 0.03
{canned beer}:  sup = 0.078
{cat food}:  sup = 0.023
{chewing gum}:  sup = 0.021
{chicken}:  sup = 0.043
{chocolate}:  sup = 0.05
{citrus fruit}:  sup = 0.083
{coffee}:  sup = 0.058
{cream cheese}:  sup = 0.04
{curd}:  sup = 0.053
{dessert}:  sup = 0.037
{domestic eggs}:  sup = 0.063
{frankfurter}:  sup = 0.059
{frozen meals}:  sup = 0.028
{frozen vegetables}:  sup = 0.048
{fruit/vegetable juice}:  sup = 0.072
{grapes}:  sup = 0.022
{ham}:  sup = 0.026
{hamburger meat}:  sup = 0.033
{hard cheese}:  sup = 0.025
{hygiene articles}:  sup = 0.033
{ice cream}:  sup = 0.025
{long life bakery product}:  sup = 0.037
{margarine}:  sup = 0.059
{meat}:  sup = 0.026
{misc. beverages}:  sup = 0.028
{napkins}:  sup = 0.052
{newspapers}:  sup = 0.08
{oil}:  sup = 0.028
{onions}:  sup = 0.031
{other vegetables}:  sup = 0.193
{pastry}:  sup = 0.089
{pip fruit}:  sup = 0.076

{pork}:  sup = 0.058
{rolls/buns}:  sup = 0.184
{root vegetables}:  sup = 0.109
{salty snack}:  sup = 0.038
{sausage}:  sup = 0.094
{shopping bags}:  sup = 0.099
{sliced cheese}:  sup = 0.025
{soda}:  sup = 0.174
{specialty bar}:  sup = 0.027
{specialty chocolate}:  sup = 0.03
{sugar}:  sup = 0.034
{tropical fruit}:  sup = 0.105
{waffles}:  sup = 0.038
{whipped/sour cream}:  sup = 0.072
{white bread}:  sup = 0.042
{whole milk}:  sup = 0.256
{yogurt}:  sup = 0.14
{whole milk, beef}:  sup = 0.021
{whole milk, bottled beer}:  sup = 0.02
{other vegetables, bottled water}:  sup = 0.025
{bottled water, rolls/buns}:  sup = 0.024
{soda, bottled water}:  sup = 0.029
{whole milk, bottled water}:  sup = 0.034
{yogurt, bottled water}:  sup = 0.023
{whole milk, brown bread}:  sup = 0.025
{other vegetables, butter}:  sup = 0.02
{whole milk, butter}:  sup = 0.028
{other vegetables, citrus fruit}:  sup = 0.029
{whole milk, citrus fruit}:  sup = 0.031
{yogurt, citrus fruit}:  sup = 0.022
{whole milk, curd}:  sup = 0.026
{other vegetables, domestic eggs}:  sup = 0.022
{whole milk, domestic eggs}:  sup = 0.03
{whole milk, frankfurter}:  sup = 0.021
{whole milk, frozen vegetables}:  sup = 0.02
{other vegetables, fruit/vegetable juice}:  sup = 0.021
{whole milk, fruit/vegetable juice}:  sup = 0.027
{margarine, whole milk}:  sup = 0.024
{whole milk, newspapers}:  sup = 0.027
{other vegetables, pastry}:  sup = 0.023
{other vegetables, pip fruit}:  sup = 0.026
{other vegetables, pork}:  sup = 0.022
{other vegetables, rolls/buns}:  sup = 0.043
{other vegetables, root vegetables}:  sup = 0.047

{other vegetables, sausage}:  sup = 0.027
{other vegetables, shopping bags}:  sup = 0.023
{other vegetables, soda}:  sup = 0.033
{other vegetables, tropical fruit}:  sup = 0.036
{other vegetables, whipped/sour cream}:  sup = 0.029
{other vegetables, whole milk}:  sup = 0.075
{other vegetables, yogurt}:  sup = 0.043
{pastry, rolls/buns}:  sup = 0.021
{soda, pastry}:  sup = 0.021
{whole milk, pastry}:  sup = 0.033
{pip fruit, tropical fruit}:  sup = 0.02
{whole milk, pip fruit}:  sup = 0.03
{whole milk, pork}:  sup = 0.022
{root vegetables, rolls/buns}:  sup = 0.024
{sausage, rolls/buns}:  sup = 0.031
{soda, rolls/buns}:  sup = 0.038
{tropical fruit, rolls/buns}:  sup = 0.025
{whole milk, rolls/buns}:  sup = 0.057
{yogurt, rolls/buns}:  sup = 0.034
{tropical fruit, root vegetables}:  sup = 0.021
{whole milk, root vegetables}:  sup = 0.049
{yogurt, root vegetables}:  sup = 0.026
{soda, sausage}:  sup = 0.024
{whole milk, sausage}:  sup = 0.03
{shopping bags, soda}:  sup = 0.025
{whole milk, shopping bags}:  sup = 0.025
{tropical fruit, soda}:  sup = 0.021
{whole milk, soda}:  sup = 0.04
{yogurt, soda}:  sup = 0.027
{whole milk, tropical fruit}:  sup = 0.042
{yogurt, tropical fruit}:  sup = 0.029
{whole milk, whipped/sour cream}:  sup = 0.032
{yogurt, whipped/sour cream}:  sup = 0.021
{whole milk, yogurt}:  sup = 0.056
{other vegetables, root vegetables, whole milk}:  sup = 0.023
{other vegetables, yogurt, whole milk}:  sup = 0.022