How to run

This python program can be run from command line with 3 parameters:

python paytm.py "transaction data file" "return/cancel data file" "output file name"

The output of the program will be a file with a list of merchant IDs, along with the variables used for clustering, and a final column which denotes which cluster the merchant belongs to, 0..n, where n is the number of clusters requested.

A sub-sample of the merchants were used for the testing due to limitations of my computer.

Approach and methods

This problem, while not particularly difficult from a technical stand point, raised issues with the need to be a machine learning approach. Normally, creating a score for a record could be treated similar to a classification problem. If there had been historical ratings for this data, my approach would have been as simple as using an ANN to train, then using the output layer sigmoid function value to take a ranking. But because this data has no historical scores or labeling, an unsupervised learning approach needed to be taken. It would have been easy to create a weighted sum program or competitive based scoring, but that is not a machine learning approach, and would inherently inset designer bias into the weights used to rank and score.

The approach I took was to use a k-means cluster algorithm to group together like merchants, based on a number of different variables present in the data. From this clustering, the labeling of quality would need to be done by a human operator. This is because the purpose of machine learning is to make decisions and inference from whatever the data says, with as little intervention as possible. From just the raw variables and data given, it is possible to group together like merchants, but being able to say whether they are "good" or "bad" is a decision that will always have to be made by a human, since the metrics used to measure this quality is subjective. For example, goodness could be determined by the number of returns, or the number of canceled orders, or both. Or the weight that the number of late orders have could have different affects. Is a merchant who is always late but never has any returns better than a merchant who is never late but has a high number of returns?

A secondary step to this approach could be to take the results from this sample set, and use it to train a supervised learning algorithm, such as an ANN in order to score any new merchants. This could also be used to translate the 1-5 integer scale to a 0-100 rating, which would be more like a real merchant rating system.

Variable selection

Though the data files has a fairly high number of columns, many of them are not relevant to any kind of grouping or ranking. The first to be removed were any columns which related to the order or the product itself. So T1, T2, T4, order_id, order_item_id, product_id, item_created_at, and fulfillment_created_at were all dropped and not considered. Merchant information was aggregated at the ID level, and any differences in the product levels were not

considered. If a merchant is good with orders of socks, but very bad with orders of paper, then they are still the same merchant and must be taken as a whole. Variables to do with price were also removed. After some testing, price variables ended up dominating the grouping, where total sales tended to group together, no matter how many bad orders or lates the merchants had. The date and time variables were kept and used to determine if an order was late. An order was classified as late if the time it shipped was after the ship by date for the order. The time that the merchant created the fulfillment was not considered, as starting the shipment is not the same as actually sending it out.

In the end the variables which where used were the total number of orders a merchant had shipped, the total number of late orders a merchant had sent out, the number of cancelled orders, the number of returned orders, and the percentage of how many orders were late and how many were canceled or returned.

Ranking method

The output of the algorithm is the merchant IDs with the variable data and a cluster assignment. From this, I made the analysis of which cluster gets which rating, going from [Bad, Poor, Fair, Good, Excellent]. The merchants which were grouped together were found to have similar numbers of orders, as well as similar numbers of returns, lates, and cancellations. The percentages of bad and late orders played a roll as well. Merchants with very high numbers of bad orders were grouped together, and had a high likelihood of having a high number of bad orders. This is likely due to simple volume. From the grouping, it appears that the more orders you serve, and thus the more shipments you are needing to fulfill, the more likely you are to have more bad orders. This doesn't hold over entirely, as there are some smaller merchants which have bad orders and are also grouped out.