Association Rules Recommendations

Executive Summary:

Dillard's, a chain clothing retail store has locations across the country. They are aiming to bring in more profit by moving up to SKU's across the chain. In order to do use, they use utilize the data from their POS system to determine which SKU's are the best candidates for modifying their planograms. They can only make 20 moves due to budgetary reasons, and a move consists of moving one SKU to another position.

Our client has provided us with over 120 million instances of transactional data which should be more than enough for us to perform significant analysis using association rules. Namely, we want to examine which SKU's have the highest likelihood of being bought together and tend to even be dependent on each other.

To do the proper analysis, we have thoroughly explored the data and extract only what we are interested in order to run the association rules efficiently and effectively which they will then be able to copy a similar process in the future. Furthermore, we have provided a csv file for our client containing our results giving them the freedom to explore the results for themselves as well and justify our analysis or make their own judgements.

Problem Statement:

Our client, Dillard's, is primarily a large clothing retail store that is looking to change their planograms across their store in order to bring in more profits.

Methodology:

Initially we tried to load in all the data at once, but soon realized that there are simply too many transactions for any analysis to be done efficiently. Unfortunately, the computations and the memory required to do the analysis from the raw data alone is too taxing for the ordinary machine. Consequently, we decided to look at a specific subset of the data while dropping any features from the data tables that are not really relevant to the problem statement. For this, we decided to look at transactions from all stores in Colorado. This reduced the number of transactional data from roughly 120 million to 3.5 million which is still more than enough data. Furthermore, we selected the important features and left out the unnecessary ones. We use the following: 'sku', 'store', 'register', 'trannum', 'saledate', 'stype', 'quantity', and 'amt'.

Next, we noticed that each row in the data did not equate to a unique transaction. We had to figure out a way to combine unique transactions so that we can pull baskets of each transaction and the items (SKU's) bought or returned in the transaction. In order to do so, we needed to create our own unique identifiers for each instance in the data table. After looking at all the features, we kept 'store', 'register', 'trannum', 'saledate' for a reason as these are intuitively the most likely features that will separate one transaction from another. We combined these features into one and ordinal encode them so that every instance that has the same combination will have the same encoding allowing us to

separate transactions from one another. We are left with about 1.3 million unique transactions. As a side note, we also calculated that 7.8% of the transaction types were returns. We decided that it was small enough to keep in our analysis since they are still a purchase a customer made and want to keep it in consideration, and it seems as though the probability of returning is relatively small as they make up a small portion of the data.

Table 1: Example Instances of Modified Transaction Data

sku	store	stype	quantity	amt	trns
4	8309	P	1	40	529746
5	8609	P	1	50	851152
5	9709	R	1	50	1835294
131	8109	P	1	25	16208
131	8109	P	1	25	32335
155	9609	R	1	12	1673192
155	9609	P	1	12	1678128
164	8209	P	1	20	386819

While the transactions data table gives us the most useful information, it is not all telling. When looking for SKU's that are good candidates we want to ensure that these SKU's are profitable for the client and it is not guaranteed that they all are. We pulled the data table with information about the SKU's, their corresponding stores, and the cost and retail price. Then we inner joined our transactional data with our SKU cost data by the SKU and store features. Now we have singular data table that has information about each SKU and the amount a customer was charged for it as well as the cost to the client. From this, we can create a new feature, profit = amount charged – cost. We then subset the data even further by looking only at the SKU's that brought about an average profit over $1. We take an average since not all customers are charged the same price for an item due to factors like sales or promotions.

Table 2: Example Instances of Merged Data Tables

sku	store	cost	stype	quantity	amt	trns	profit
180	8109	11.25	P	1	25	231037	13.75
180	8109	11.25	R	1	25	233427	13.75
180	8109	11.25	P	1	25	303962	13.75
180	9409	11.25	P	1	25	1441340	13.75
327	9709	3.72	P	1	3.75	1863625	0.03
387	8109	38	P	1	79	205943	41
387	8109	38	P	1	79	209541	41
450	8109	2	P	1	3.99	196503	1.99

There are 147,277 unique SKU's in the data as well. Running the association rules algorithms requires unstacking the transactional data by one hot encoding each individual SKU so that we know if it's in the transaction or not. With 1.3 million transactions this takes up too much memory so we need

to cut down the number of SKU's we are looking at. Here we decided that if an SKU shows up in less than 0.04 of our transactions, then it's not an item popular enough for it to be worth a move for our client. This dramatically reduces our SKU's of interest down to 255 which is a lot more scalable. Now we can reduce our transactions of interest as well by doing another subset such that each transaction has at least 1 SKU of interest. This filtering process leaves us with 174,010 transactions of interest.

Finally, we merge all unique transactions and one hot encode the data so that it is in the format necessary for the association rules algorithms.

Table 3: Example Instances of Basket Data

SKU #	7915	19633	29633	39633	…
	0	0	1	0	…
	0	0	0	0	…
	0	0	0	0	…
	0	0	0	0	…

Association Rules Results and Analysis:

Running association rules on the transactions takes in a few arguments. We need to decide what our minimum support is and what our performance metric will be for us to consider a rule worth looking at. Given the amount of transactions we have, if we set our minimum support too high, we will end up filtering out too many rules which was indeed the case. Most combinations of SKU's are going to have really supports so we set our threshold at a conservative 0.01%. There are various performance metrics that we can look at such as support, confidence, or lift. All metrics are valuable indicators, but we chose to go with lift to set our minimum threshold which we set at 1. This is because a lift of 1 indicates that the items are essentially independent from one another.

Table 4: Example Instances of Rules Results

antecedents	consequents	antecedent support	consequent support	support	confidence	lift
[5199905]	[7049904]	0.00154579	0.001471087	0.000534	0.345724907	235.0133
[7049904]	[5199905]	0.001471087	0.00154579	0.000534	0.36328125	235.0133
[6409904]	[5379905]	0.001827366	0.001856098	0.000764	0.418238994	225.3324
[5379905]	[6409904]	0.001856098	0.001827366	0.000764	0.411764706	225.3324
[6989904]	[6949904, 5189905]	0.001907816	0.001344665	0.000523	0.274096386	203.8399
[6949904, 5189905]	[6989904]	0.001344665	0.001907816	0.000523	0.388888889	203.8399
[2726578, 3998011]	[803921, 3908011]	0.003235242	0.00024135	0.000138	0.042628774	176.6262
[803921, 3908011]	[2726578, 3998011]	0.00024135	0.003235242	0.000138	0.571428571	176.6262

Finally, we have the rules results. There are 1558 rules from our given parameters above and 179 unique SKU's in the rules results. We take the top 100 SKU's of best candidacy, let's call them

candidates, based on highest lift score. We then filter out the rules that do not contain any candidates in either the antecedent or consequent and we are left with 750 rules and our final results to give to our client. Again, with the results, the client can decide what metric or metrics they want to use to decide which 20 SKU's to move. While a lift of 1 indicates independence from the antecedent and consequent, a confidence of 1 indicates that the antecedent and consequent always occur together.

Rule with highest lift of 235: [5199905] -> [7049904]

Rule with highest confidence of 0.96: [803921, 3998011, 3908011] -> [2726578]

Rule with highest support of 0.0075: [3978011] -> [3524026]

It is important to note that looking at these metrics by themselves may not be the best idea. For example, the rule above with the highest support has a confidence of 0.166 and a lift of 3.12 which are not great. All metrics should be looked at in conjunction, but it seems as though the rules with high lifts tend to (not always) have decent confidence too at least with this dataset.

Recommendations:

Our final recommendation to the client is ultimately our deliverable which includes a csv file called 'best_rules.csv' containing the rules with only the top 100 SKU candidates using lift as the main metric. We also include 'rules.csv' which contain all the rule before we filter out just the candidate SKU's just in case the client would like to use a different metric and filter out 100 SKU's based on their metric. With just the SKU's our client can find out more information about the item such as brand using their database.

Our dataset is also only looking at Colorado due to computing limitations. If they have the capacity, they can follow a similar process to the entire dataset.