Making Dillard's Great Again: An Associative Analysis

IEMS 308 - Data Science and Analytics, Homework 2: Association Rules

Saif Bhatti

February 17, 2020

1 Executive Summary

Dillard's are a large department store, with a desire to optimise their item layout in order to improve revenue. By optimising this layout in some strategic manner, Dillard's can drive spontaneous purchases of items that are frequently purchased together. Point of Sales data (transactions) can be analysed to find the relevant item associations.

- In order to determine the positioning of a 'stock keeping unit' (SKU) in the store, it is key to find items with strong associations. Association rules analysis is an analytic technique serving to identify such relations.
- First, initial data exploration was performed on the transactions to determine key attributes that would be most relevant to uncover associative relations between SKUs. In order to provide specific advice (and also to mitigate memory and runtime issues pertaining to the size of data (initial dataset was over 11 GB)), the data was subset to only contain transaction data pertaining to a Dillard's outlet in Oklahoma City, Oklahoma. However, by changing the store ID, this same analysis may be applied to any Dillard's store.
- The results were...

2 Problem Statement

- The Point of Sales (POS) data contains transaction data from store locations across the US. Given Dillard's desire to rearrange the floors of their stores (change the planograms), I will provide a list of 100 SKUs that are the best candidates to modify their planograms.
- It is useful to consider the visual of what the SKUs represent. In the planogram representation of a commercial medication aisle below, each item is an SKU. The red circle overlaid is indicative of decision to stock relevant items together based on frequency of purchase. This exercise (in some level of abstraction) requires finding 100 such red circles through the use of *associative rule analysis*.

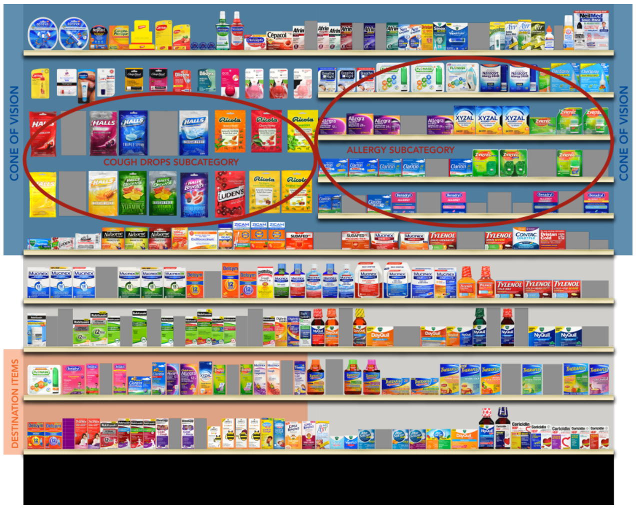


Figure 1: An Example Planogram

3 Data Methodology

The steps taken are itemised as below.

3.1 Exploratory Data Analysis

- Run **Exploratory Data Analysis** to find data subsets that provide maximal value to associative rule analysis without using the entire dataset. This results in removing many columns and subset down to a single Dillard's outlet.
  - The data is stored as several csv files, which are connected as per the relational database schema laid out below.

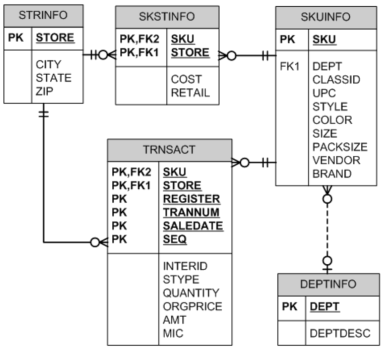
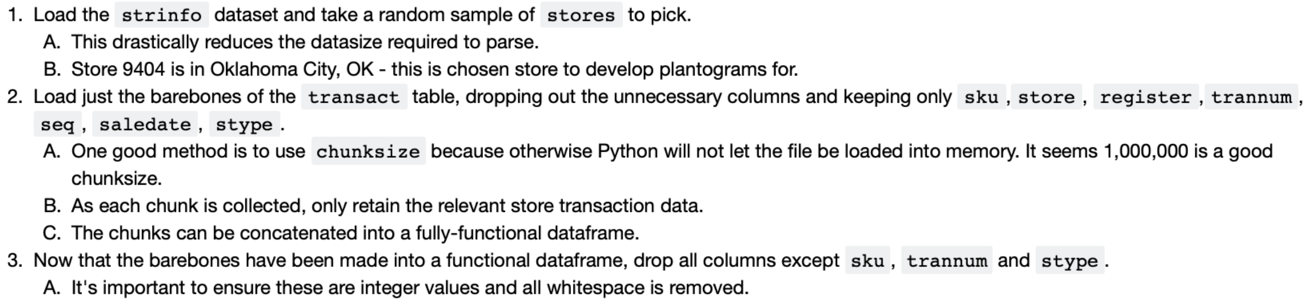


Figure 2: Dillard Relational Database Schema

- The original data .zip file downloads at over 1 GB, and then unzips to be over 11 GB of data. This size as well as complex schema made it difficult to load successfully into the sqlite3 database as a SQL database. It seemed that the data was not complying with the given schema, and to resolve issues would require finding issues - this requires parsing the data - and returns in an infinite loop. Of course, there is an alternative option: chunk processing.
- Instead, I used chunk processing on the standard pandas.read_csv functionally to parse the data without violating memory constraints. Even so, the resulting dataset is so large that we need to in some "clever way select a subset of data".
  - The following steps were applied to the data:



1. Load the `strinfo` dataset and take a random sample of `stores` to pick.
   A. This drastically reduces the datasize required to parse.
   B. Store 9404 is in Oklahoma City, OK - this is chosen store to develop plantograms for.
2. Load just the barebones of the `transact` table, dropping out the unnecessary columns and keeping only `sku` , `store` , `register` , `trannum` , `seq` , `saledate` , `stype` .
   A. One good method is to use `chunksize` because otherwise Python will not let the file be loaded into memory. It seems 1,000,000 is a good chunksize.
   B. As each chunk is collected, only retain the relevant store transaction data.
   C. The chunks can be concatenated into a fully-functional dataframe.
3. Now that the barebones have been made into a functional dataframe, drop all columns except `sku` , `trannum` and `stype` .
   A. It's important to ensure these are integer values and all whitespace is removed.

Figure 3: Data Preprocessing

3.2 Pre-processing

- **Pre-process** the data by (1) removing duplicate records; (2) intra-transaction SKUs are represented as just 1 instance instead of many; (3) SKUs that were represented less than 5 instances across transaction table were removed due to lack of prevalence; (4) all remaining SKUs were one-hot encoded so that they appeared as binary indicators per transaction row to signify if a particular SKU was purchased in that particular transaction.

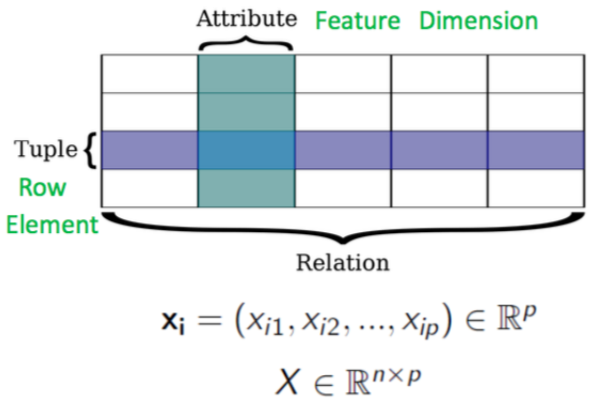

Attribute
Feature
Dimension
Tuple
Row
Element
Relation
$\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{ip}) \in \mathbb{R}^p$
$X \in \mathbb{R}^{n \times p}$

Figure 4: Final Data Format prior to Apriori Algorithm

3.3 Associative Rule Analysis

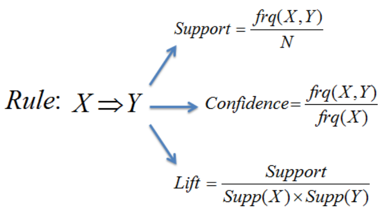
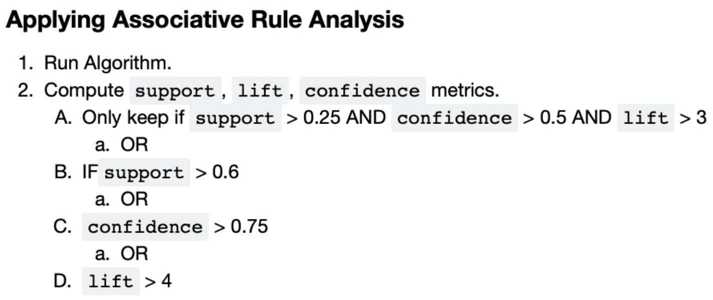

$Rule\text{:}\ X \Rightarrow Y$
$Support = \frac{frq(X,Y)}{N}$
$Confidence = \frac{frq(X,Y)}{frq(X)}$
$Lift = \frac{Support}{Supp(X) \times Supp(Y)}$

Figure 5: Abstract Metrics for Associative Rule Analysis

- The abstract metrics (and their mathematical forms) are shown above. Specific to this dataset, the Apriori algorithm computes the following characteristics of each item:
  - **Support**: relative frequency of item occurrence. An association rule is better the higher its support.
  - **Confidence**: probability that a basket contains the consequent given that it contains the antecedent. An association rule is better the higher its confidence.
  - **Lift**: confidence of a rule divided by the probability that the consequent is in a transaction. High lift - just like high confidence and high support - implies that the associative rule has significant meaning.



Applying Associative Rule Analysis

1. Run Algorithm.
2. Compute `support` , `lift` , `confidence` metrics.
   A. Only keep if `support` > 0.25 AND `confidence` > 0.5 AND `lift` > 3
      a. OR
   B. IF `support` > 0.6
      a. OR
   C. `confidence` > 0.75
      a. OR
   D. `lift` > 4

Figure 6: Association Rule Process

- For each of these metrics, if a particular SKU does not meet the minimum threshold set (in order of computation), all subsequent considerations (computations) of that SKU are thrown out. However, if that SKU preforms poorly on just one of these metrics and higher on the others, it remains in consideration.
- This process then iterates with a growing basket size until all possible sets of SKUs are exhausted (either eliminated or pass the thresholds set).
- In order to return a list of 100 SKUs considered valuable to move, the algorithm was iterated using varying threshold values until 100 unique SKUs could be produced as antecedents.

4 Results & Conclusions

Make Recommended Moves:

- Given the analysis, it is clear that Dillard's should reevaluate its planograms, and consider SKUs around. The association rules created had high confidence and lift values, indicating that grouping some item combinations will result in greater revenue from related spontaneous purchasing.

Pareto Principle in Action:

- At outset, Dillard's laid out the constraint that they can make *at most* 20 SKU moves across the entire chain due to fiscal considerations. The original hypothetical planogram layout was presented as being highly sub-optimal, and this analysis has confirmed these suspicions.

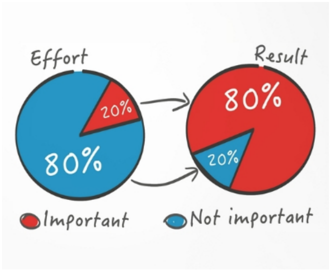
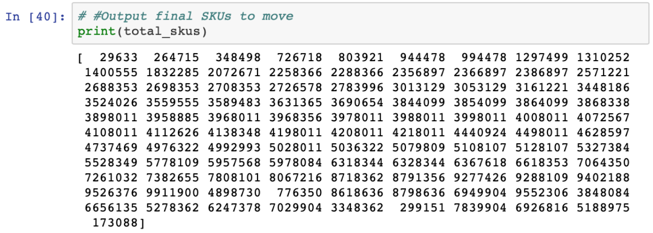


Figure 7: Visualising Progress

- The rules created involve many of the same SKUs, organised in various patterns. A large number of the rules with shorter antecedents and consequents are subsets of rules with larger antecedents and consequents. Since the algorithm returned the required 100 unique rules despite requiring a low minimum support value, this hints that Dillard's would reap a significant amount of the benefit of associative rule analysis within the constraints of only 20 moves.



```
In [40]: # #Output final SKUs to move
         print(total_skus)

[  29633  264715  348498  726718  803921  944478  994478 1297499 1310252
 1400555 1832285 2072671 2258366 2288366 2356897 2366897 2386897 2571221
 2688353 2698353 2708353 2726578 2783996 3013129 3053129 3161221 3448186
 3524026 3559555 3589483 3631365 3690654 3844099 3854099 3864099 3868338
 3898011 3958885 3968011 3968356 3978011 3988011 3998011 4008011 4072567
 4108011 4112626 4138348 4198011 4208011 4218011 4440924 4498011 4628597
 4737469 4976322 4992993 5028011 5036322 5079809 5108107 5128107 5327384
 5528349 5778109 5957568 5978084 6318344 6328344 6367618 6618353 7064350
 7261032 7382655 7808101 8067216 8718362 8791356 9277426 9288109 9402188
 9526376 9911900 4898730  776350 8618636 8798636 6949904 9552306 3848084
 6656135 5278362 6247378 7029904 3348362  299151 7839904 6926816 5188975
  173088]
```

Figure 8: Final SKUs to move

5 Next Steps

- The Apriori algorithm may be effective at generating associations within sets, it does not account for fiscal considerations. It is possible that profit margins on sets of items with lower metric performance {support, confidence, lift} is significantly higher than the profit margin on sets with much higher metric performance. Adding these to considerations will be a helpful step in gauging and comparing efficacy of SKU moves.

- Given the compute power, time and memory limitations - this analysis featured only one store. However, the rules generated may not generalise well across stores.

-∞. **Warning**: *It seems that nobody actually uses the Apriori algorithm for real associative learning applications; the reasons are as follows. For large data,* ***candidate itemsets will always be very large*** *- even 2-item, 3-item sets with support-level pruning end up being very large (clearly a lot bigger than the data itself). If the dataset has a lot of frequently purchased items with low support values, the resulting itemsets will be massive. In addition, the Apriori algorithm also scans the database multiple times to construct its itemsets, which makes it both* ***computationally expensive and slow****.*

6 Appendix

1. Data can be found at this hyperlink: Dillard's POS data.

2. Source code can be found at this hyperlink: Github-saif1457