Final Year Dissertation

Market Basket Analysis with Graph Theory

April 19, 2021

Sahil M. Pattni

Bachelor of Science with Honours in Computer Science

Supervised by Dr. Neamat El Gayar

Declaration

I, Sahil Manojkumar Pattni, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Date: April 19, 2021

Signed: Sahil Manojkumar Pattni

Abstract

In this digital age, data is being generated and collected at an unprecedented rate, with data analytics employed by corporations and small businesses alike to produce actionable insights, reduce costs, optimize operations and increase revenue. Association rules allow us to identify relationships between products - some of which may be non-intuitive - that can provide insights into customer spending habits and product perception.

Several algorithms exist for deriving association rules from transactional datasets, and many of these produce a vast number of rules, sometimes too many for the end-user to extract those that may ultimately be beneficial. To that end, we introduce a novel algorithm: the CGRG algorithm, that generates a limited number of high value association rules from a clustered minimum spanning tree, with each rule being of a distinct class: either a bi-cluster rule or an intra-cluster rule. The rules generated were compared against those generated by the Apriori Algorithm, and it was found that the CGRG algorithm managed to capture a majority of the high value rules from the ruleset generated by the Apriori Algorithm, a result of generating rules that can be classified only as intra-cluster or bi-cluster.

Contents

Chapter 1

Introduction

1.1 Motivation

For businesses such as groceries, hypermarkets, and retail outlets that deal with the trade of heterogeneous physical assets, operations such as inventory management and product placement play an instrumental role in determining the business' financial success. These involve asking questions such as:

- Which products should be placed at the entrance of the store? Which should be placed closer to the exit?
- Which products will benefit the most by being placed at eye-level?
- Which products should be placed next to each other to maximize the purchase volume?

One way to find optimal solutions to such queries is to employ the use of Association Rule Mining (also known as Market Basket Analysis). This set of techniques assess frequent itemsets (e.g. from sales data) and generate association rules between products. A prime example of the utility of association rules is the urban legend of *"Beers and Pampers"*, where a company allegedly studied their point-of-sale data and found a strong association between the purchase of diapers and a particular brand of beer during a certain time. With *diapers* as the antecedent and *beer* as the consequent, this rule can be written as:

$$\{Diapers\} \rightarrow \{Beer\}$$

This is an example of a single-element rule, where both the antecedent and consequent are sets that contain only one element each. Several algorithms exist for association rule mining, most prominently the Apriori Algorithm **(Agrawal and Srikant 1994)** and FP-Growth **(Han et al. 2000)**, however these algorithms tend to generate an overwhelming amount of rules, rendering it inconvenient for the end-user to extract actionable information from the ruleset.

1.2 Aims

The aim of this project is to improve upon the methodology proposed by **(Valle et al. 2018)**, where the authors derived association rules from a minimum spanning tree. We plan on improving upon this by introducing a methodology to derive two distinctive types of rules (bi-cluster and intra-cluster) from a minimum spanning tree that has been segmented by a clustering algorithm.

1.3 Objectives

The research objectives for this paper are as described below:

1. Acquire a suitable dataset upon which the study can be conducted.
2. Construct an affinity graph from the chosen dataset.
3. Explore and evaluate MST extraction algorithms.
4. Explore methods for clustering topological structures such as graphs.
5. Extract the minimum spanning tree from the affinity graph.
6. Segment the minimum spanning tree into distinct clusters.
7. Develop and implement a methodology for extracting association rules from a minimum spanning tree.
8. Evaluate the generated association rules against the rules generated by the Apriori Algorithm.

Chapter 2

Background

2.1 Transaction and Association Representation

2.1.1 Binary Purchase Vectors

Let $I = I_1, I_2, \ldots, I_m$ be a set of binary attributes (i.e. items), and let T be a database of transactions. As defined in **(Agrawal, Imieliński, et al. 1993)**, a binary purchase vector is a transaction t represented as a vector of length m, where:

$$t[k] = \begin{cases} 1 & \text{if } I_k \text{ purchased in } t \\ 0 & \text{otherwise} \end{cases}$$

For example, consider a grocer who only sells five items: milk, eggs, bread, apples and oranges. Consider the following transactions:

Transaction t_1: Customer purchases bread and oranges.

Transcation t_2: Customer purchases eggs, bread and apples.

Transaction t_3: Customer purchases milk and oranges.

The binary purchase vectors for these transactions would be:

	milk	eggs	bread	apples	oranges
t_1	0	0	1	0	1
t_2	0	1	1	1	0
t_3	1	0	0	0	1

Table 2.1: Binary Purchase Vectors

2.1.2 Product Association

The association between products pairs across all transactions can be ascertained from the binary purchase vectors using the Pearson's Correlation Coefficient **(Pearson 1895)**. Since the correlations are of two

binary variables, the Pearson's Correlation Coefficient is equivalent to the Phi-Coefficient ϕ **(Ernest C and El-Sanhurry 1991)**, where for n observations:

$$\phi = \sqrt{\frac{\chi^2}{n}}$$

Applying this correlation formula to the set of binary purchase vectors in Table 2.1, we get:

	milk	eggs	bread	apples	oranges
milk	1.0	-0.5	-1.0	-0.5	0.5
eggs	-0.5	1.0	0.5	1.0	-1.0
bread	-1.0	0.5	1.0	0.5	-0.5
apples	-0.5	1.0	0.5	1.0	-1.0
oranges	0.5	-1.0	-0.5	-1.0	1.0

Table 2.2: Correlation Matrix

Note that the correlation matrix is diagonally symmetrical. This is true of all correlation matrices where the items on the x-axis and y-axis are the same and in the same order. This correlation matrix can then be represented as a graph, where the nodes are the products, and the edges are the correlation values.

2.2 Graph Theory

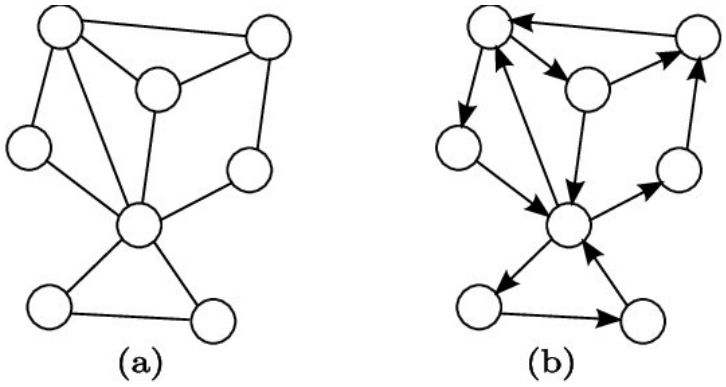

Figure 2.1: Undirected and Directed Graphs

In discrete mathematics and more specifically - graph theory, a graph is a data structure that contains a set of nodes (i.e. vertices) connected by lines (i.e. edges). These edges may be directed - such as in Figure 2.1:(a), or undirected - such as in Figure 2.1:(b). The edges may contain values (i.e. weights) between the two vertices it connects. A graph G with a set of vertices V and a set of edges E can be

represented via the notation $G = (V, E)$. For the scope of this project, we will be building undirected graphs, where the weight between two vertices is the same in both directions.

2.2.1 Minimum Spanning Trees

Given an undirected $G = (V, E)$, a *spanning tree* can be described as a subgraph that is a tree which includes all the vertices V of G with the minimum number of edges required. A *minimum spanning tree* (MST) is the spanning tree with the smallest sum of edge weights. This means that if the graph has n vertices, each spanning tree - including the minimum spanning tree - will have $n - 1$ edges. Since a minimum spanning tree captures the lowest weights in a graph, with modifications it could be an excellent candidate to capture the opposite as well: the strongest associations between products. There are two widely used algorithms to extract the minimum spanning tree from a graph: Prim's algorithm and Kruskal's algorithm.

2.2.2 Prim's Algorithm

Independently discovered by three authors, Prim's algorithm **(Prim 1957)(Jarník 1930)(Dijkstra 1959)** is a greedy algorithm[1] to find the minimum spanning tree of an undirected, weighted graph G. To successfully implement the algorithm, three sets need to be maintained: a set of *discovered* edges, and two sets of vertices: a set of *undiscovered* vertices, and a set of *discovered* vertices. Figure 2.2 illustrates Prim's algorithm being applied to a graph. The algorithm is as follows:
Initialize an empty set of discovered edges: E, and two sets of vertices: an empty set D of the discovered vertices, and UD as the set of undiscovered vertices.

- Pick an arbitrary vertex as a starting point (in the case of Figure 2.2, the top right node). Add this vertex to D and remove it from UD.
- While UD is not empty:
  - Find the edge e_i with the smallest weight such that it connects together a vertex V_1 in D and V_2 in UD (to avoid forming cycles).
  - Append V_2 to D and remove it from UD (i.e. V_2 is now discovered).
  - Append e_i to E.

[1] Selecting the locally optimal choice at each iteration of the solution

Once D contains all the vertices of G, the algorithm terminates, and the set D represents the minimum spanning tree, and $\sum_{i=1}^{n} e_i$ is the weight of the MST. The time complexity of this algorithm is $O(V^2)$.

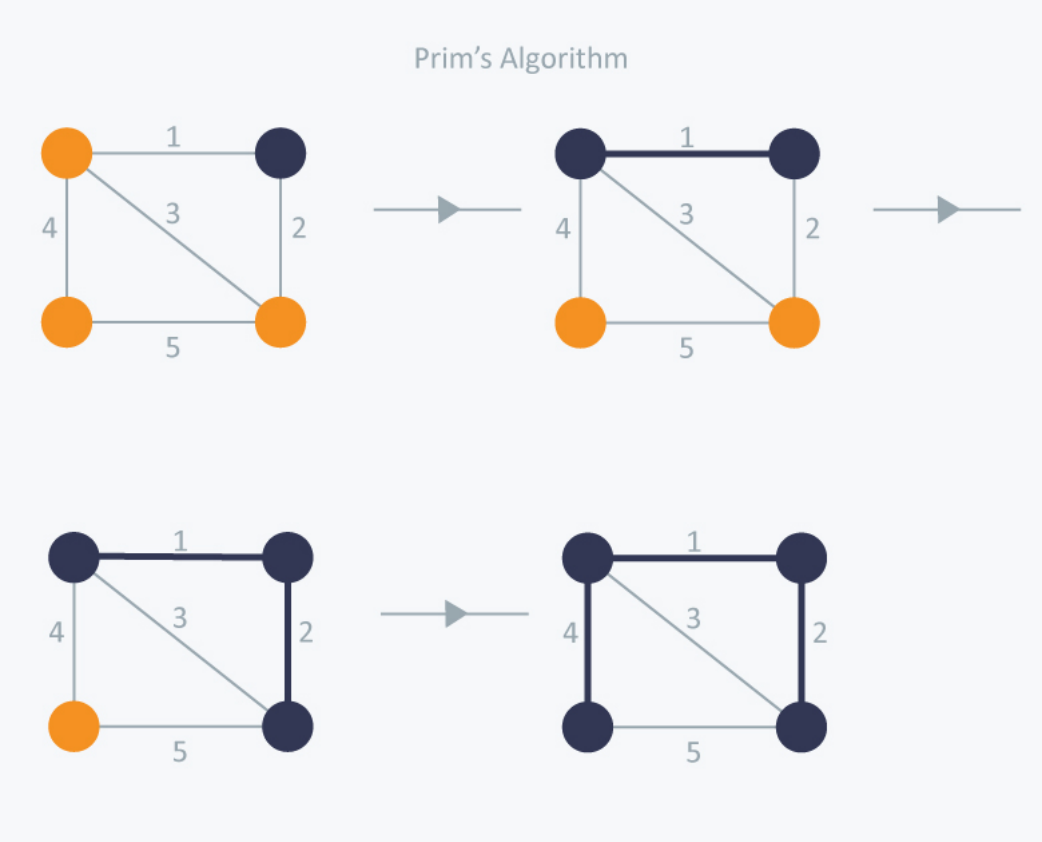

Figure 2.2: Prim's Algorithm applied to a graph **(Abdelnabi 2018)**

2.2.3 Kruskal's Algorithm

Another greedy algorithm, Kruskal's algorithm **(Kruskal 1956)** also extracts the MST from a graph. Unlike Prim's algorithm, Kruskal's doesn't select an edge that connects directly to the already built spanning tree, but rather picks the global optimal solution. The algorithm is as follows:
Maintaining a set of edges E, and an initially empty set of chosen edges C:

- Sort the set of edges E in ascending order.
- While the number of elements in C is not $n - 1$:
  - Select the smallest edge e_i from E.
  - If adding it does not form a cycle with the spanning tree formed so far, append e_i to C.

– Remove e_i from E.

The algorithm will terminate when $n-1$ edges have been selected. The time complexity for this algorithm is $O(ElogE)$.

2.3 Related Work

2.3.1 Extracting Minimum Spanning Trees using K-Means

(Zhong et al. 2015) proposed a novel framework to extract the minimum spanning tree of a graph based on the K-Means clustering algorithm. Their methodology can be separated into two distinct phases. In the first phase, the data is partitioned into $\sqrt{n}$ clusters via K-Means and the Kruskal's algorithm is applied to each of the clusters individually. Once the $\sqrt{n}$ MSTs have been constructed, they are combined to form an approximate MST. In the second phase, new partitions are constructed based on the borders of the clusters identified in phase 1. Based on these new partitions, a second approximate MST is constructed. Finally, both graphs are merged such that the resulting graph has $2(n-1)$ edges. The Kruskal's algorithm is run on this graph to get the final approximation of the MST.

Critical Analysis

The authors have proposed an efficient way to approximate a minimum spanning tree, with their methodology having a complexity of $O(N^{1.5})$, which is faster than the standard MST algorithm which has a complexity of $O(N^2)$. For clarity, we have illustrated the disparity between the author's algorithm and the standard algorithm on Figure 2.3.

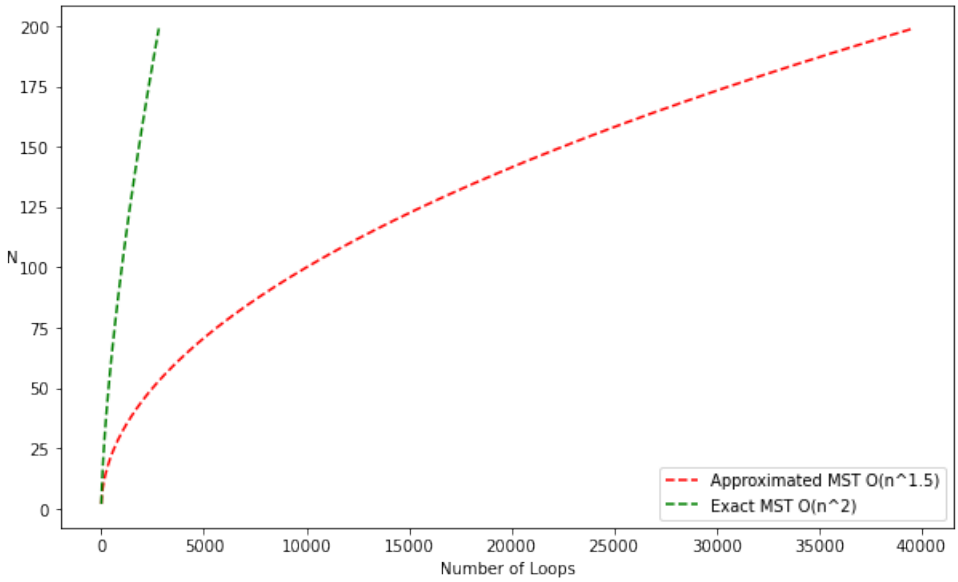

Figure 2.3: Efficiency of K-Means optimized MST vs. exact MST

The K-Means optimized algorithm is $\frac{\sqrt{n}-1}{\sqrt{n}}\%$ faster than the standard MST algorithm.

2.3.2 Markov Clustering

(Dongen 1969) introduced the novel Markov Clustering Algorithm (MCL), a graph clustering algorithm based on Markov Processes **(Markov 2006)** that favours sparse graphs (i.e. graphs where the average degree is smaller than the number of nodes). The reason for this follows the notion that dense regions in sparse graphs correspond with regions in which the number of k-length paths is relatively large - and therefore - paths whose beginnings and ends are in the same dense region have a higher probability for random walks[2] of length k as opposed to other paths. In other words, random walks originating from a node in a dense region have a high probability of ending in that same dense region. Given a non-negative weighted directional graph G, the MCL simulates flow in the graph. It does this by first mapping the graph G in a generic way onto a Markov matrix M_1. Once completed, the set of transition probabilities (TP's) are iteratively recomputed via expansion and contraction, resulting in an array of Markov Graphs. In the expansion stage, higher TP's are calculated, whereas for the contraction phase, a new Markov Graph is created by rewarding high TP's and penalizing low TP's. The belief is that by doing so, the flow between dense regions that are sparsely connected will be removed, leaving only distinct and unconnected dense regions (i.e. clusters). The author tested the MCL

[2] A random walk is a stochastic process of of successively random steps along a space, in this case: a directed graph.

algorithm against randomly generated graphs which were known to possess a natural cluster structure. The authors noted that the MCL algorithm successfully managed to capture the segregation present in the graphs, and concluded that the algorithm was capable of handling graphs with large natural clusters.

Critical Analysis

The author proposed a novel way to apply the mathematics behind Markov processes to successfully segment and isolate densely connected segments of a graph. The results from this paper have inspired us to use the clustering algorithm in our own work.

2.3.3 AIS Algorithm

(Agrawal, Imieliński, et al. 1993) proposed a novel algorithm to generate all statistically significant association rules between items in a database, laying the foundation for association rule mining. Given a set of items $I = \{I_1, I_2, I_3, \ldots, I_m\}$, the authors define an association rule to be of the form $X \rightarrow I_j$ where X is a set of items such that $X \subset I, I_j \notin X$. The hypothetical database stated was a list of transactions, T, where each transaction t was a binary vector of length m, as described in Section 2.1.1. The authors define two constraints for assessing association rules:

Support

The support of an association rule is the proportion of transactions in which the itemsets in the rule are present. For a set of transactions T, where $T(i)$ denotes the set of transactions in which the set of items $i \subseteq I$ was purchased:

$$\text{support}(i) = \frac{T(i)}{T}$$

Similarly, where $T(i_k, i_j)$ represents the set of transactions in which the itemsets $\{i_k \subset I, i_j \subset I\}$ were purchased, the support for the association rule $i_k \rightarrow i_j$ can be represented as:

$$\text{support}(i_k \rightarrow i_j) = \frac{T(i_k, i_j)}{T}$$

Support scores correspond with the statistical significance of a rule, and rules with low support scores may not occur frequently enough to draw reasonable conclusions from.

Confidence

Confidence is the conditional probability of an itemset $i_j;\ \ i_j \subset I$ being present in a transaction given that itemset $i_k;\ \ i_k \subset I$ is present in the same transaction. The confidence of the association rule $i_k \rightarrow i_j$ can be represented as:

$$\text{confidence}(i_k \rightarrow i_j) = \frac{T(i_k, i_j)}{T(i_k)} \equiv \frac{\text{support}(i_k \rightarrow i_j)}{\text{support}(i_k)}$$

The confidence of an association rule can be thought of as the rule's *strength*.
With these constraints defined, the authors state that their methodology for association rule mining can be split into two discrete steps:

1. The generation of candidate itemsets.

2. The generation of statistically significant association rules from the itemsets.

Candidate Itemset Generation

To generate candidate itemsets, the authors first generate all possible itemsets from the database, defining those whose support score was above a support constraint $min_{support}$ as *large itemsets*. The authors note that a brute-force check[3] would be sub-optimal, taking up to 2^m passes of the database (where m is the number of items in the itemset I). Therefore, they introduced a methodology where they would only observe itemsets of length k on the k^{th} pass of the dataset, to see if the itemsets satisfied the support constraint. On the $(k+1)^{th}$ pass of the dataset, they need only check itemsets that are *1-extensions* (i.e. itemsets extended by only one item) or the *large itemsets* discovered in the previous pass. Their reasoning is now commonly known as *The Apriori Principle*, where they state that if an itemset i_k is *large* (i.e. satisfies the support constraint), then any subset $i_j \subset i_k$ will also be *large*. This reasoning also applies that if an itemset i_j is found to be *small* (i.e. did not satisfy the support constraint), then any superset $i_k; i_j \subset i_k$ will also be *small*. This allows the authors to prune the number of association rules whose support scores need to be computed, as if they find i_j to be *small*, any superset $i_k; i_j \subset i_k$ need not have its support score computed as it is known to be *small*.
However, if an itemset i_j is indeed found to be *large*, then another multiple passes over the dataset will be required to check the support scores for subsets of i_j. To avoid this, the authors devised a measure to calculate the expected support $\bar{s}$ of an itemset. The expected support is used to estimate the support of $i_j = (i_p + i_q)$, not only when i_j is expected to be *large*, but also when i_p is expected to be *large* yet $(i_p + i_q)$ is expected to be *small*. This estimation further prunes the number of rules whose support scores

[3] checking every possible itemset iteratively.

need to be computed.

Association Rule Generation

To generate association rules, the authors used the following technique:
for each *large* itemset $Y = \{i_1, i_2, \ldots, i_k\}; k \geq 2$ from the set of non-pruned *large* itemsets, generate a set of association rules in the form $X \rightarrow i_j; X \subset Y, i_j \notin X$ such that X is of length $k-1$. Therefore, each *large* itemset will produce k rules. From the generated rules, the authors discarded those rules whose confidence scores fell below the confidence constraint $min_{confidence}$.

Evaluation

The authors tested their methodology on a sales dataset with $46,783$ transactions, with 63 distinct departments. Their configuration was composed of a support constraint of 1% (i.e. $min_{support} = 0.01$) and a confidence constraint of 50% (i.e. $min_{confidence} = 0.5$). The authors note that the rules produced follow what general intuition might suggest. For example:

$$\{\text{Auto Accessories, Tires}\} \rightarrow \{\text{Automotive Services}\}$$

Furthermore, the authors assessed the accuracy of their support estimation metric $\bar{s}$ by observing the ratio of correctly estimated itemsets for both *small* and *large* against various values for the support constraint. They were able to conclude that their estimation accuracy was satisfactory, as their accuracy was 96% and above for support thresholds.

Critical Analysis

The authors have proposed a novel methodology that has been the bedrock of numerous research publications, including most of the papers in this literature review. Their estimation function performed with high accuracy, meaning it can reduce the number of passes through a database the algorithm has to take by a significant amount. Additionally, their pruning techniques allowed them to eliminate a large proportion of itemsets from the space. Even after the significant pruning of rules, a major drawback of this methodology is the large number of rules produced, although one could argue that only the highest performing rules need be observed in further detail. Finally, the algorithm only allows the consequent to have one item, thereby limiting the type and quality of rules produced.

2.3.4 Apriori Algorithm

(Agrawal and Srikant 1994) improved on their previous work with **(Agrawal, Imieliński, et al. 1993)** by introducing the Apriori algorithm, which - in addition to being faster than the AIS algorithm - can produce association rules where the consequent has more than one item. The structure for the Apriori algorithm follows closely to to the AIS algorithm in that it uses *large* itemsets to generate the association rules. The algorithm generates candidate itemsets in the k^{th} pass only from the itemsets found to be *large* in the $(k-1)^{th}$ pass, following the intuition of *The Apriori Principle*, where any subset of a *large* itemset must itself be *large*. As a result, the candidate itemsets that have k items can be generated from the *large* itemsets having $(k-1)$ itemsets, and any such itemsets that contain a subset that is *small* are discarded. For every *large* itemset l, all non-empty subsets of l are gathered. For every subset a, it generates a rule in the form:

$$a \rightarrow (l - a)$$

if the support and confidence constraints of the rule are met.

Critical Analysis

The authors have further improved upon their AIS algorithm in the form of the Apriori Algorithm, which is much better known than the former due to its inherent ability to generate more complex rules, and its efficiency in doing so. The ability to generate more complex rules makes it a good benchmark to test our own algorithm against, to see whether it can either serve as an alternative or even a complement to the Apriori Algorithm.

2.3.5 Subjective Measurement of Association Rules

(Zekic-Susac and Has 2015) proposed a novel measure for the *interestingness* of association rules, identifying that a dominant, universally used measure did not exist. The authors' goal was to combine objective measures such as the support, confidence and lift scores with more subjective measures. Instead of the Apriori approach, their methodology has them generate association rules via the *tree-building technique* - which compresses a large database into a Frequent-Pattern tree, citing that this technique was more efficient than the Apriori algorithm. The authors employed the heuristical unexpectedness measure[4] and the heuristical actionability measure[5] as their subjective measures, and a minimum confidence

[4] How significantly a rule contradicts a user's prior beliefs.
[5] If the user believes they can use the information to their advantage (e.g. a promotion).

threshold of 51% as their objective measure. Since a subjective measure would require a human subject, the authors' used the estimations of a sales manager from a Croatian retail chain, and stored his responses in binary format for the subjective measures (i.e. 0 if a rule was unexpected else 1, 0 if a rule is not useful else 1). The dataset used for this paper was a real transactional dataset with 14,012 transactions and a set of 1,230 unique items (which was later pruned to 7,006 transactions and a set of 278 products) from the same Croatian retail chain their test subject worked at. The authors then generated association rules from the first-level hierarchical grouping of items from the dataset (items with a minimum support of 25%), of which 36 rules were identified as statistically significant. From this set of rules, only two rules satisfied both subjective measures and the confidence constraint, and therefore these two rules were identified as highly interesting. The authors then generated association rules from the second-level hierarchical grouping of items, where items that represented the same product (but had different a manufacturer, brand etc.) were grouped together. Of the rules generated, 15 satisfied their confidence constraints and had a support value able 10%. 5 rules from this set satisfied both their subjective and objective measures, more than the previous experiment The authors were able to conclude that the increase in accuracy and number of interesting rules resulted from the second level of grouping which generalized the products.

Critical Analysis

Wheras the original measure of statistical significance introduced by **(Agrawal, Imieliński, et al. 1993)** was purely objective, the authors of this paper have presented a well thought out approach to combining the subjective metrics with objective ones to produce a human-verified association rule set. A few caveats to note, however: their study only involved one subject, which is rarely regarded to be statistically acceptable. An ideal study would require multiple, randomly chosen subjects to offset any bias that the singular subject would have had, and in addition, the larger their subject size, the closer their collective estimations will model the total population's. Another drawback of their approach is that by using human intuition as a metric, they're promoting association rules that satisfy pre-existing notions about human behavior (e.g. if someone buys milk, they'll *probably* get eggs too), however these types of rules are usually regarded as common knowledge, whereas the beauty of association rule mining is in its ability to surface association rules that - while true - seem unintuitive, and therefore are less likely to be known by the management of such organizations.

2.3.6 Association Rules from Minimum Spanning Trees

(Valle et al. 2018) proposed a novel methodology to study the structure and behavior of consumer market baskets from the topology of a minimum spanning tree which represented the interdependencies between products, and use this information to complement the association rule generation process. The input to their proposed methodology was a correlation matrix between the set of all binary purchase vectors for every transaction. The dataset used for the MST construction was a list of $1,046,804$ transactions containing a set of $3,240$ unique products from a large supermarket chain branch in Santiago, Chile. When building this correlation matrix, the authors opted to use the Pearson's Coefficient **(Pearson 1895)** - which is equivalent to the coefficient ϕ for binary data **(Ernest C and El-Sanhurry 1991)** - over the traditionally used Jaccard distance to compute the similarity between the binary product vectors, as the former provides both a positive and negative association between products. Additionally, they used the distance function $d_{ij} = \sqrt{2(1-\phi_{ij})}$ to transform the correlation matrix into a distance measurement (i.e. the weight of the edges)[6].

MST Analysis

The authors constructed a MST for 220 product subcategories, and noted that there was a significant level of grouping between product sub-categories that belonged to the same parent category. To remove edges from the MST that were not statistically significant, the authors used the mutual information **(Cover and Thomas 2006)** measure $\sum_{x,y} log_2 \frac{r(x,y)}{p(x)q(y)}$ between product subcategories p and q, and were able to prune 14 edges, all of which were connected to a terminal node, therefore effectively pruning 14 vertices from the MST too. To identify the most influential regions of the MST, the authors defined an influence zone of distances that were in the 10^{th} percentile. To generate meaningful association rules, for each MST product i, the authors ran a search for the set of all association rules R_i such that $P_i \rightarrow P_j (i \neq q)$. Then from the resulting set of rules, they searched for rules that obeyed $P_i \rightarrow P_m$ where m a product node connected to the product i in the minimum spanning tree. For both resulting sets of rules for each product, the mean of their lift scores were observed, and the authors determined that the rules that were reinforced by the MST had a higher mean, and that a majority of these rules had a lift score above the 90^{th} percentile.

Inherent Clustering

To identify the clusters each of the products should be identified under, the authors constructed a

[6] ϕ_{ij} is the correlation score between products at indexes i and j on the correlation matrix.

hierarchical tree using the average linkage clustering method, and by using an unspecified cut distance, they were able to produce 17 taxonomic groups (i.e. clusters). Cross-referencing their results with the actual parent categories of the products, they were able to conclude that the MST did indeed categorize the product sub-categories into clusters with a reasonable degree of accuracy.

Comparing to an alternative methodology

The authors then compared their MST to another methodology, namely the structured association map (SAM) **(Kim 2017)**, using the Jaccard distance as a measure of similarity, and were able to generate interesting 2x2 rules (i.e. $\{A, B\} \rightarrow \{C, D\}$), all with lift scores above 1.0, with one rule even having a lift score of 106.46. They concluded that while both approaches provided different information, they both visually identify the strongest relationships between the products, and provide useful information to reduce the search space for association rules.

Critical Analysis

The authors' approach seems to be novel, thorough and well structured. Their methodology successfully employed the use of minimum spanning trees to complement the association rule generation process with sound results. One caveat of their approach is that they only used the MST to generate single-element rules (i.e. $\{A\} \rightarrow \{B\}$, where the antecedent and consequent contain only one element each). Using the distance score in conjunction with the importance function they defined (i.e. $\sum_{k \in K_u} \frac{1}{w_{uk}}$), they could have defined a system to produce multi-element rules (i.e. where either/both the antecedent and the consequent have more than one element) While single-element rules are easily understandable and tend to have high lift values when extracted from the MST, multi-element rules would provide an additional layer of insight as to how a range of products (perhaps a cluster) relate to another.

2.3.7 Summary

In conclusion, **(Agrawal, Imieliński, et al. 1993)** introduced a formal system for association rules, and a method to generate them from a transactional database, which was further improved when **(Agrawal and Srikant 1994)** introduced the Apriori Algorithm. Instrumental to the success of this algorithm is its ability to prune a bulk of the rules such that their support scores need not be checked. **(Pennsylvania**

2020) defines the equation for calculating the number of possible rules for an itemset $\boldsymbol{d}$ as:

$$number\ of\ rules = \sum_{k=1}^{d-1} \left(\binom{d}{k} \times \sum_{j=1}^{d-k} \binom{d-k}{j} \right)$$

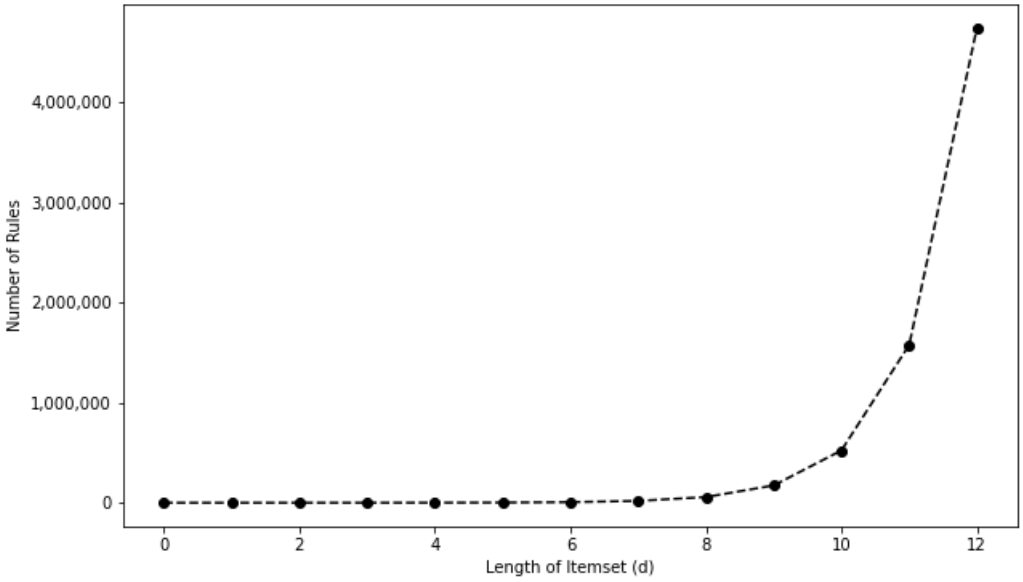

Figure 2.4: Number of Association Rules for an Itemset

We have plotted Figure 2.4 to illustrate the exponential growth of rules given an itemset's length $\boldsymbol{d}$. This exponential growth reinforces the need for the rule pruning techniques introduced in **(Agrawal, Imieliński, et al. 1993)** to be able to complete the computations within a reasonable time frame, and will therefore be incorporated into our methodology. However, while these papers only based the interestingness of an association rule on objective measures such as the support, confidence and lift scores of these rules, **(Zekic-Susac and Has 2015)** proposed a methodology where subjective human input was used to validate the interestingness of these rules. A drawback of the above methods, however, is the large number of rules produced. **(Zhong et al. 2015)** proposed a solution to the relatively slow computation time that the Prims's algorithm and Kruskal's algorithm offer, where the Kruskal's algorithm was optimized using K-Means, leading to a significant performance increase as illustrated in Figure 2.3. The methodology proposed in this paper was a framework, where Kruskal's Algorithm could be substituted with any MST algorithm, such as Prim's Algorithm. **(Dongen 1969)** introduced a novel

way to identify and isolate dense regions within a graph, and we have used their algorithm to cluster our own graphs. **(Valle et al. 2018)** introduced a methodology to extract high value association rules from a minimum spanning tree, used to complement the rules produced by the Apriori algorithm. A caveat of this approach, however, is that it can only produce rules such that the antecedent and consequent are sets with one element each. This paper has been the primary motivation for our work, and therefore we have incorporated into our work several techniques that the authors used and/or introduced.

As a note, **(Brin et al. 1997)** defined another metric to assess association rules - *conviction* - now commonly referred to as *lift*. The lift of an association rule $i_k \rightarrow i_j$ is the rise that i_k gives to the confidence of the rule. The formula for lift is:

$$\text{lift}(i_k \rightarrow i_j) = \frac{\text{confidence}(i_k \rightarrow i_j)}{\text{support}(i_j)}$$

All the research conducted above has served as the inspiration for this project. We will be implementing an affinity from transactional data in a fashion inspired by **(Valle et al. 2018)**, and will be using the algorithm proposed by **(Kruskal 1956)** to derive a minimum spanning tree from the affinity graph. We will also be using the metrics such as support and confidence described in **(Agrawal, Imieliński, et al. 1993)** and rule pruning techniques similar to those they have mentioned. Lastly, our algorithm will be benchmarked against the Apriori algorithm **(Agrawal and Srikant 1994)** to evaluate its usefulness.

Chapter 3

The CGRG Algorithm

In this section, we introduce the main contribution of this paper: the Clustered-Graph Rule Generation (CGRG) algorithm. The CGRG algorithm derives association rules from a clustered minimum spanning tree. The input to the algorithm is a minimum spanning tree, where the vertices are a set of items and the edges are the associations between them. Section 3.1 describes how the minimum spanning tree is segmented into clusters. Using these clusters, Section 3.2 details how itemsets for the rules will be generated from these clusters. Sections 3.3 and 3.4 describe how the algorithm will generate two distinct classes of rules (bi-cluster and intra-cluster, respectively). Section 3.5 describes how the rules will then be pruned such that only the statistically significant rules will remain. Figure 3.1 illustrates the complete flow for the CGRG algorithm.

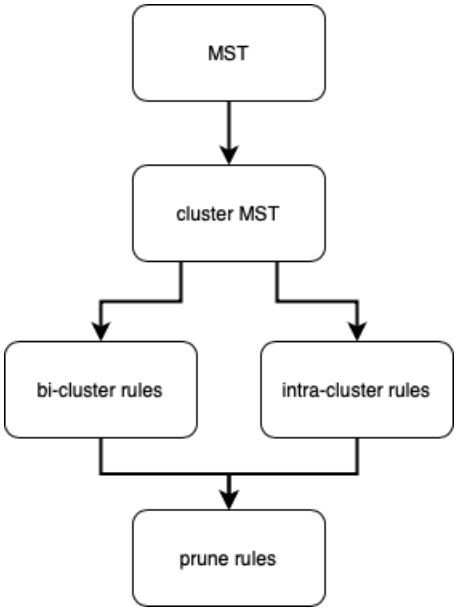

Figure 3.1: CGRG algorithm

3.1 Markov Clustering

The MST will be clustered using the Markov Clustering (MCL) algorithm. Multiple cluster configurations will be generated using the algorithm with varying inflation scores to identify the most modular configuration[1], which will then be used to segment the MST.

3.2 Itemset Generation

For every cluster Cl_i identified by the MCL algorithm, all proper subsets cl such that $cl \subseteq Cl_i$ are generated. The set of proper subsets are stored against the key i, which denotes the index of cluster Cl_i, effectively assigning each cluster its proper subsets. Listing 3.1 is the pseudo-code for this operation.

```
given CLUSTERS = {Cl_1, Cl_2, ..., Cl_n}
define itemsets_by_cluster as {∅}
for Cl_i in CLUSTERS
    to items_by_cluster add ({all cl if cl ⊆ Cl_i} at index i)
```

Listing 3.1: Cluster Itemset Generation

3.3 Bi-Cluster Rule Generation

Bi-cluster rules are those where the antecedent and the consequent originate from distinct and separate clusters. All possible bi-cluster rules are generated such that for any two given clusters Cl_k and Cl_j:

$$cl_k \rightarrow cl_j; \;\; cl_k \subset Cl_k, \; cl_j \subset Cl_j$$

Listing 3.2 presents the pseudo-code for the bi-cluster rule generation. All bi-cluster combinations are generated and stored as a pair of indices (i.e. (i, j) for clusters Cl_i and Cl_j). For every cluster pair generated, every configuration itemsets associated with each cluster (calculated in Listing 3.1) is added to the ruleset such that a rule exists for every itemset c_k associated with cluster C_k and every itemset c_j associated with cluster C_j.

[1] Where the connectedness between dense clusters is minimal

```
given CLUSTERS = {Cl_1, Cl_2, ..., Cl_n}
given itemsets_by_cluster = {...} // already populated
define rules as ∅
define combs as ∅
to combs add all ^nC_2 combinations from {1, 2, ..., n}

for comb in combs:
    define antecedent_itemsets as (itemsets_by_cluster at (c at index 1)) // first index
     is 1
    define consequent_itemsets as (itemsets_by_cluster at (c at index 2))
    for antecedent in antecedent_itemsets:
        for consequent_item in consequent:
            to rules add (antecedent → consequent)
            to rules add (consequent → antecedent)
```

Listing 3.2: Bi-Cluster Rule Generation

3.4 Intra-Cluster Rule Generation

Intra-cluster rules are those where itemsets in both the antecedent and consequent originate from the same cluster. This can be represented as:

$$cl_{k1} \rightarrow cl_{k2}; \quad cl_{k1} \subseteq Cl_k,\ cl_{k2} \subseteq Cl_k,\ cl_{k1} \cap cl_{k2} = \emptyset$$

```
given CLUSTERS = {Cl_1, Cl_2, ..., Cl_n}
given itemsets_by_cluster = {...} // already populated
define rules as ∅
for i in itemsets_by_cluster
    define itemsets as (itemsets_by_cluster at index i) // itemsets at i^th cluster.
    define combs = ∅
    to combs add all ^pC_2 combinations from itemsets // For all p elements in itemsets

    for comb in combs
        define antecedent as (comb at index 1) // first index is 1
        define consequent as (comb at index 2)
        to rules add (antecedent → consequent)
        to rules add (consequent → antecedent)
```

Listing 3.3: Intra-Cluster Rule Generation

Listing 3.3 is the pseudo-code for the intra-cluster rule generation. A rule is generated from every configuration of two subsets from the subsets associated with a given cluster.

3.5 Rule Pruning

```
given rules = {...}                 // already populated
sort rules ascending by antecedent // sort by number of items in antecedent
define min_support as float         // user-specified value
define min_confidence as float      // user-specified value
define valid_rules as ∅
define below_threshold as ∅
for r in rules
    antecedent = (r at index 1) // first index is 1
    consequent = (r at index 2)
    define is_above_threshold = True
    // check if support lookup can be skipped
    for itemset in below_threshold
        if itemset ⊂ r
            is_above_threshold = false
            break
    if is_above_threshold
        // calculate all supports
        define support_antecedent as float
        define support_consequent as float
        define support_r as float
        define confidence as (support_r ÷ support_antecedent)
        define lift as (confidence ÷ support_consequent)
        if support_antecedent < min_support
            to below_threshold add support_antecedent
        if support_consequent < min_support
            to below_threshold add support_consequent
        if confidence < min_confidence:
            next loop // move to next rule in for loop
        if support_r < min_support
            next loop
        // If all constraints met, add rule
        to valid_rules add r
```

Listing 3.4: Rule Pruning

Listing 3.4 presents the pseudo-code for rule pruning, employing the use of the *Apriori Principle* described in Section 2.3.3, which states that *the support of a set is - at most - equal to the support of its subsets.* We first sort the rules in ascending order by the number of items in the antecedent of each rule. When iterating through the rules, if the support of either the antecedent or consequent is found to be below the user specified support constraint, it is added `below_threshold` - the set of itemsets found to be below our constraint. For every subsequent iteration, if it is found that any set from `below_threshold` is a subset of the current rule, we can discard the rule and avoid having to calculate its support as we know it will be below our support constraint. Our final rules are those that remain after pruning.

Chapter 4

Experiments and Analysis

4.1 Environment

All the code for this algorithm was written in Python **(Python 2020)** and Rust **(Rust 2020)**. Python was used to read and sanitize the initial dataset, and Rust was used to generate the binary purchase vectors. We opted to use Rust instead of Python for this task because the generation of this vectors has a time complexity of $O(N^2)$ at worst, and therefore may take an order of magnitude longer to run on a slower language such as Python. Python was used for the rest of the tasks, which included the graph generation and clustering, as well as the itemset generation and rule pruning. The following external Python libraries were used to aid development:

- `pandas`: Data manipulation.
- `numpy`: Data manipulation.
- `matplotlib`: Plotting library.
- `seaborn`: Plotting library.
- `networkx`: Graph generation and plotting.
- `markov_clustering`: Markov Clustering Algorithm.

4.2 Data Transformation

Given an itemset $I = \{I_1, I_2, \ldots, I_m\}$ and a set of tranactions $T = \{T_1, T_2, \ldots T_n\}$ such that $T_i \subseteq I$, every transaction T_i will be transformed into a binary purchase vector, as described in Section 2.1.1. An $m \times m$ correlation matrix will then be computed using the Pearson's Correlation Coefficient from the set of binary purchase vectors, composed of the correlation every item has against every other item. A graph $G = (V, E)$ will be constructed from this correlation matrix, such that the vertices $V = I$, and every edge

will be the correlation value between the two items it connects. A minimum spanning tree is a sub-graph composed of the *shortest* path connecting all vertices (i.e. the lowest sum of weights), and therefore if we construct our graph from the correlation matrix, the MST would only capture strongly negative and weakly positive relationships only, whereas we want to capture the strongest relationships between products, both negative and positive. Therefore to extract a MST with the strongest associations, the values in the correlation matrix must first be transformed before a graph is constructed from them.

4.3 Distance Function

As mentioned in Section 2.1.1, the Pearson's Correlation Coefficient is equivalent to the Phi Coefficient (ϕ) for binary variables, as is the case for our binary purchase vectors. Given ϕ_{ij} for items I_i and I_j in the correlation matrix, **(Valle et al. 2018)** used the distance function

$$\phi_{ij} = \sqrt{2(1 - \phi_{ij})}$$

to transform their correlation matrix. Given that we are using the Pearson's Correlation Coefficient that gives us a correlation score in the interval [-1,1], this equation is not suitable for us as it discards the strongly negative associations by outputting a larger weight (e.g. a correlation score of -1 would be transformed to 2) which would not be captured in the MST. Therefore, we modify the equation to be

$$\phi_{ij} = \sqrt{2(1 - |\phi_{ij}|)}$$

allowing us to capture both positive and negative associations alike. Figure 4.3 illustrates the effect of our distance function, where the stronger the correlation of the input, the lower the output. Given that our initial correlation values were bound to the interval $[-1, 1]$, the output of the distance function is similarly bound to the interval $[0, \sqrt{2}]$.

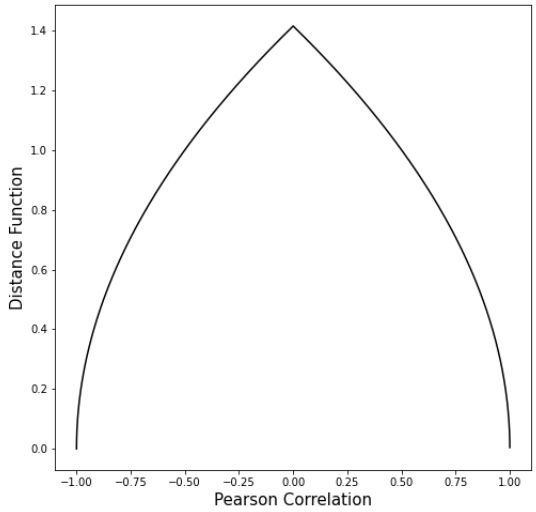

Figure 4.1: The effect of the distance function $\sqrt{2(1-|x|)}$

Once the correlation matrix has been transformed via this function, the graph $G = (V, E)$ can be constructed.

4.4 Dataset Pre-Processing

The dataset used in this study is the sales data of $4,152,919$ transactions and 39 unique product categories from a chain of Brazilian gas-station stores **(marcio486 2020)**. Each row in the dataset represents the purchase of a specific product as part of a transaction - and as such, each row corresponds to the following columns:

- Company Code
- Order Number
- Employee
- Product
- Product Category
- Client
- Sale Date Time

- Product Cost
- Discount Amount

All personal and corporate names were exchanged for fictitious names by the author of the dataset in order to preserve the anonymity of those whose who could have otherwise been identified through the dataset. Only the Product, Product Category, Client City and Discount Amount columns were retained for the purposes of our algorithm, the rest were discarded. Before employing the dataset, sanitary procedures were carried out to ensure that the dataset was error-free and in a format suitable for graph generation. The steps have been detailed below.

1. **Transaction Identifier**

   The *Order Number* field showed discrepancies, where a given order number could reference distinct transactions in different stores and cities, and at different dates and times. This could be due to the stores maintaining their own order numbers, and also because the order numbers may reset after a predetermined limit. A unique transaction identifier - named `basket_id` - was created by concatenating the order number and the date, thereby mitigating the occurrence of a identifier that references multiple transactions.

2. **Binary Purchase Vector transformation**

   The dataset was then transformed such that each transaction was represented by a binary purchase vector - as described in Section 4.2 - wherein each column represents a product category. The product categories were chosen for the graph representation over the products themselves as it would give a more generalized view on the associations between them, and the categories themselves were deemed specific enough that they would not be parent to children of significant variance.

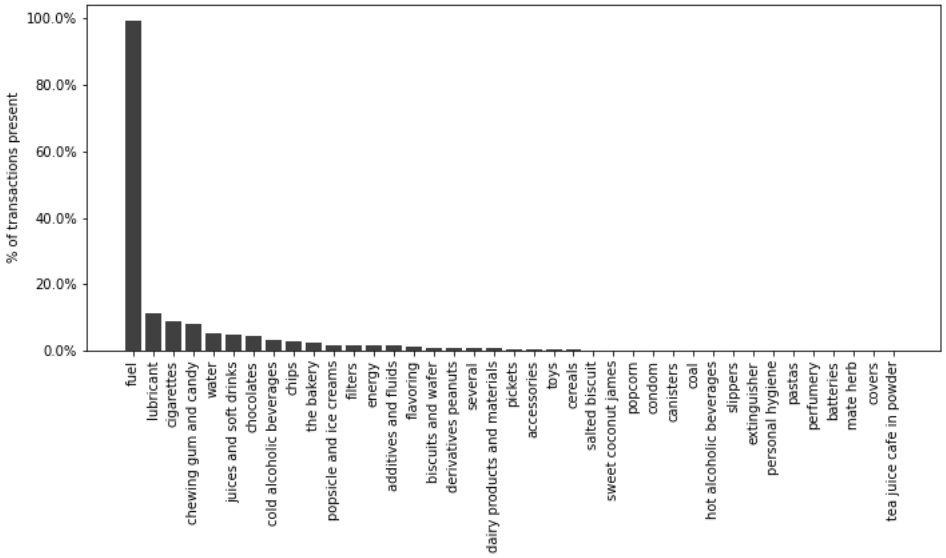

Figure 4.2: Category Distribution

The metrics used to assess the association rules - support, lift and confidence - are based on the proportional presence of a given itemset in the transactions. Since our dataset is from a gas-station store chain, fuel products dominate the transactional presence by a significant factor. Figure 4.2 highlights the disparity between the presence of *fuel* products and the others, with *fuel* being present in 99.28% of all transactions. To avoid the association rules being dominated by the *fuel* category - which should inherently understood to be a key product for gas stations - the fuel category was purged from the dataset, reducing the dataset to $1,362,617$ transactions. The correlation matrix for the 38 remaining product categories was then computed using Pearson's Correlation Coefficient, and is illustrated in Figure 4.3. Since the correlation matrix is known to be diagonally symmetrical, only the values below the diagonal have been illustrated.

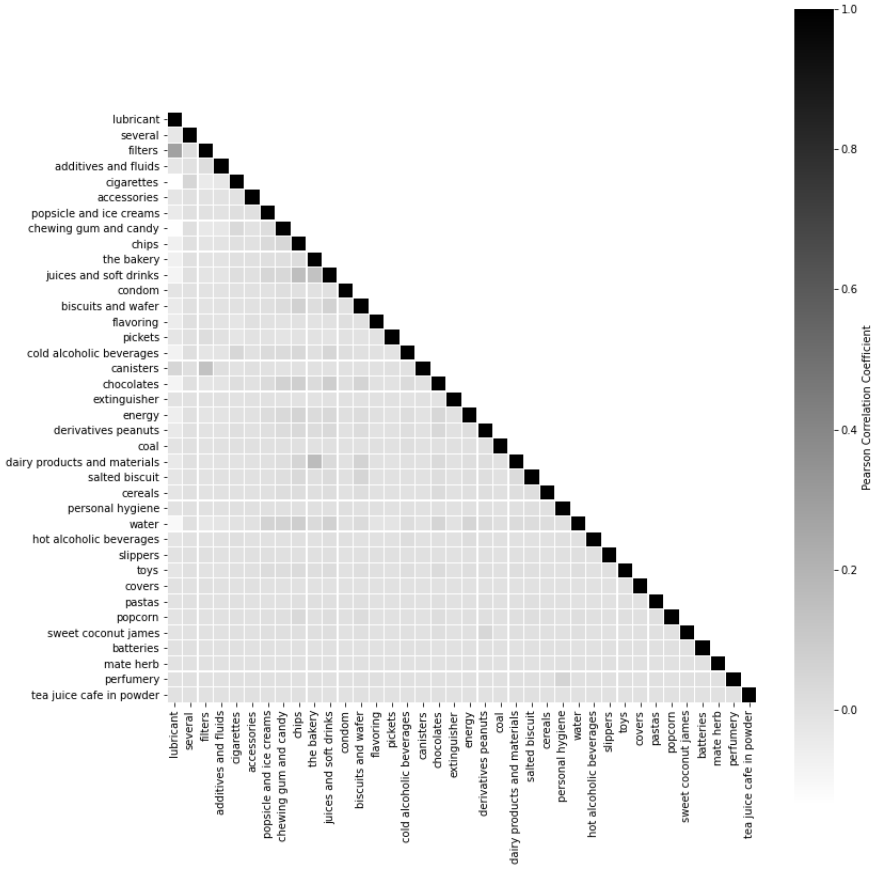

Figure 4.3: Correlation Matrix from Binary Purchase Vectors

4.5 MST Generation

As described in 4.3, the distance function $\sqrt{2(1-|\phi_{ij}|)}$ was then applied to the correlation matrix to transform the values such that the strongest associations have the lowest values. The graph $G = (V, E)$ was then constructed such that the vertices V represent the product categories, and the weights of the edges E are the transformed correlation values between the vertices the edges connect. The minimum spanning tree was then extracted from this graph using Kruskal's algorithm. Both the complete graph and the MST are illustrated in Figure 4.4. The value of each node is an integer, which corresponds to the index of the product category in the binary purchase vector dataset. The length of each edge is directly proportionate to its weight, such that the greater the weight, the greater the length of the edge.

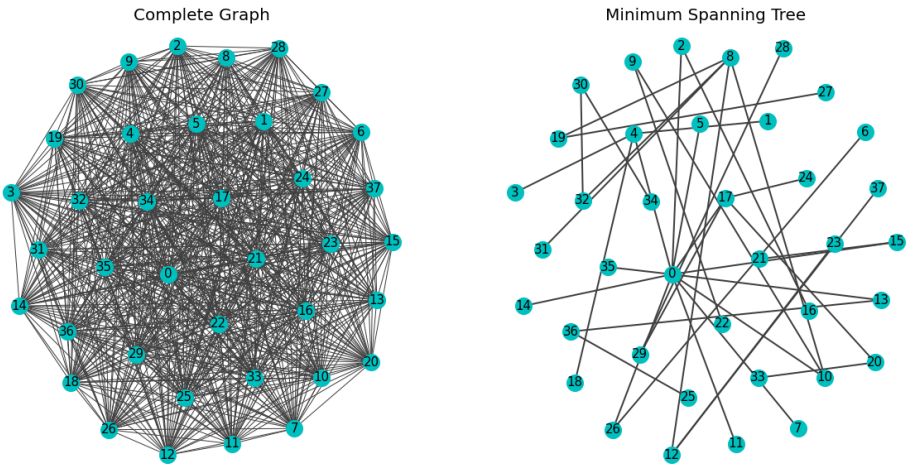

Figure 4.4: Product Category Graph and MST

4.6 Markov Clustering

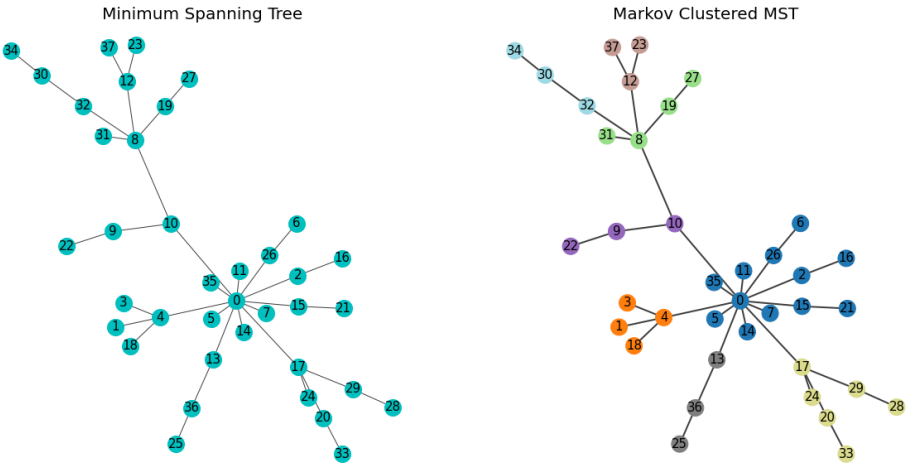

Figure 4.5: MST before and after Markov Clustering

The MST was then clustered using the Markov Clustering algorithm. To identify the most modular clustering configuration, we implemented the algorithm several times with inflation scores between 1.5

and 2.5 (inclusive) at increments of 0.1. In doing so, we discovered that an inflation score of 1.6 resulted in the most optimal modularity. The Markov Clustering configuration produced using this inflation score was therefore chosen, and the results of this clustering are illustrated in Figure 4.5. Note that while the disposition of nodes differs from that illustrated in Figure 4.4, the nodes and the edge weights are the same. The Markov Clustering algorithm segmented the nodes into 8 distinct clusters. Figure 4.6 illustrates the names of the product categories in each cluster, color-coded in accordance with the MSTs in Figure 4.5. Observing the groupings produced by the MCL algorithm, we can see that with the exception of the largest cluster in blue, the groupings do have an underlying similarity (e.g. {`biscuits and wafer, salted biscuit, tea juice cafe in powder`}).

Figure 4.6: Product categories by cluster

4.7 Comparison of CGRG and Apriori Algorithm

Once the MST was clustered, we used both the CGRG and the Apriori Algorithm to generate rules for our dataset, with the support constraint at 0.1% and the confidence constraint at 25%. Given these constraints, the Apriori algorithm generated $1,222$ rules, and the CGRG generated 123 bi-cluster rules as well as 60 intra-cluster rules, totalling 183 rules.

Apriori Rules

We present the first 5 rules generated by the Apriori Algorithm, ranked by highest support.

Antecedent	Consequent	Support	Confidence	Lift	Type
{chewing gum and candy}	{cigarettes}	0.0726	0.2986	1.0985	Apriori
{cigarettes}	{chewing gum and candy}	0.0726	0.2671	1.0985	Apriori
{water}	{chewing gum and candy}	0.0473	0.3052	1.2554	Apriori
{filters}	{lubricant}	0.0468	0.9381	2.6850	Apriori
{chocolates}	{chewing gum and candy}	0.0435	0.3117	1.2820	Apriori

Table 4.1: Apriori Rules

CGRG Rules

Similarly, we present the first 5 rules generated by the CGRG, ranked by highest support. The type of rule has also been annotated (i.e. bi-cluster or intra-cluster).

Antecedent	Consequent	Support	Confidence	Lift	Type
{chewing gum and candy}	{cigarettes}	0.0726	0.2986	1.0985	bi-cluster
{cigarettes}	{chewing gum and candy}	0.0726	0.2671	1.0985	bi-cluster
{water}	{chewing gum and candy}	0.0473	0.3052	1.2554	intra-cluster
{filters}	{lubricant}	0.0468	0.9381	2.6850	intra-cluster
{chocolates}	{chewing gum and candy}	0.0435	0.3117	1.2820	bi-cluster

Table 4.2: CGRG rules

Analysis

We observe that there is a 100% overlap between the top five rules generated by the Apriori algorithm and the CGRG. The 100% overlap stands true for up until the first 13 rules for both algorithms, after which a decline can be observed.

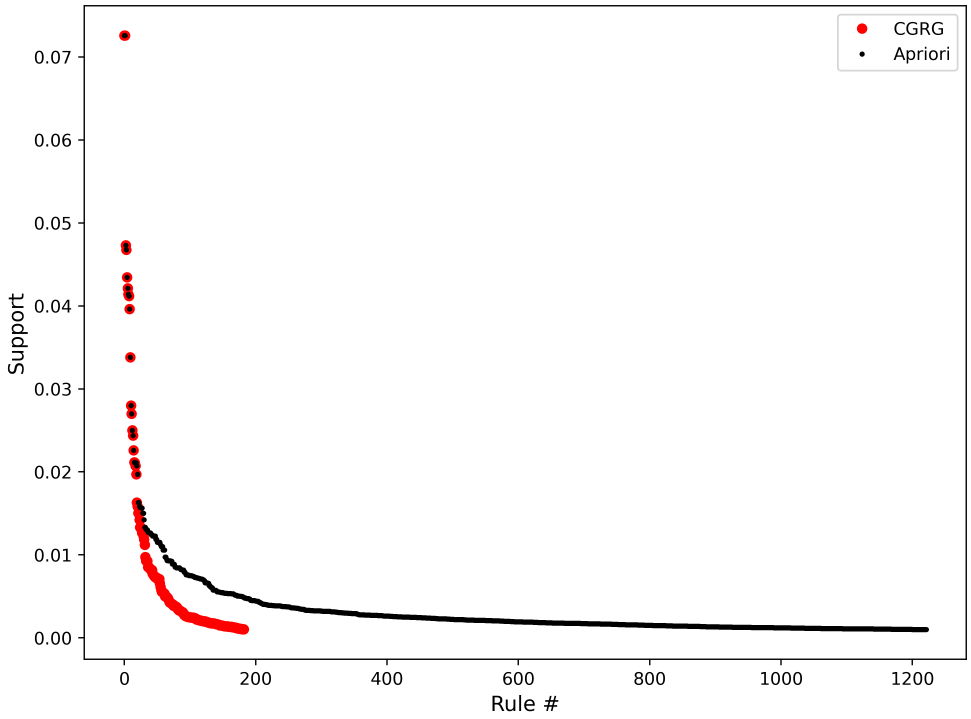

Figure 4.7: Ranked supports for both the CGRG and Apriori Algorithm

Figure 4.7 illustrates the support values for the x^{th} rule for both the CGRG and Apriori when they are sorted by the highest support score to the lowest. We can observe that when provided with the same support and confidence constraints, the Apriori algorithm produces far greater rules, whereas the CGRG algorithm captures mostly the those rules that have a relatively higher support, with the exception being at the elbow of both curves, where the CGRG algorithm captured rules with a lower support than the Apriori algorithm.

We can infer from this figure that the CGRG algorithm captures the strongest rules present in the Apriori ruleset, which leads us to theorize that *rules where the itemsets each originate from the same cluster are likelier to have a higher support score than those which don't.* In other words, given clusters A and B, and subsets of any given cluster cl_i and cl_j, rules which follow:

$$cl_i \rightarrow cl_j; \quad cl_i \subset A, \quad (cl_j \subset B \mid cl_j \subset A)$$

are likelier to have a higher support score than rules whose antecedent and/or consequent are composed of items from multiple clusters. To further reinforce this theory, we present the averages of the support, confidence and lift scores generated by both algorithms:

Algorithm	Mean Support	Mean Confidence	Mean Lift
Apriori	0.0034	0.3737	1.9096
CGRG	0.0075	0.3683	1.8855

Table 4.3: Averages for both algorithms

We can see that while the mean confidence and lift scores from the Apriori algorithm are only marginally higher those from the CGRG algorithm, the mean support for the CGRG algorithm is significantly greater than that of the Apriori algorithm by a factor of approximately 2.2, indicating that the composition of the rules generated by the CGRG algorithm is far more concentric in high-value rules than the Apriori generated ruleset.

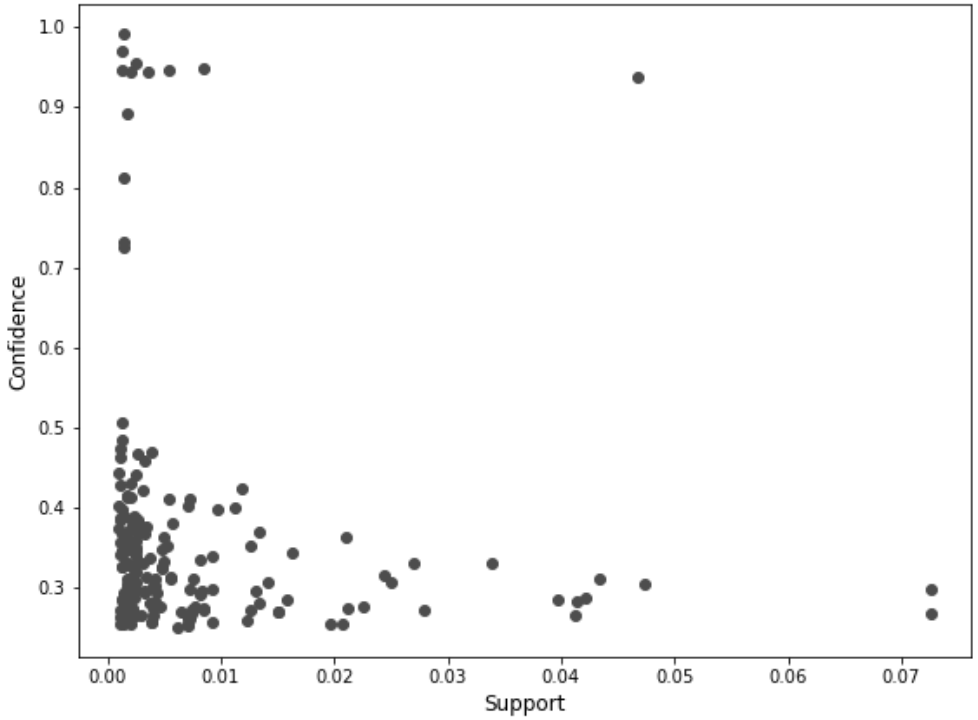

Figure 4.8: Relationship between confidence and support for CGRG

Figure 4.8 illustrates the relationship between the confidence and support for the rules generated by the CGRG algorithm. The majority of rules generated are concentric around the support and confidence constraints of 0.001 and 0.25 respectively. The outliers indicate a negative relationship between the two metrics: the rules with the highest confidence had a relatively low support, and vice versa.

As **(Brin et al. 1997)** argued, the *lift* score may be a better metric to assess the value of a rule than support. Figure 4.9 illustrates the lift values for the x^{th} rule for both the CGRG and Apriori when they are sorted by the highest lift score to the lowest. The image has been zoomed to illustrate only the first 183 rules (the length of the CGRG ruleset) to further illustrate the disparity at the elbows of the curves.

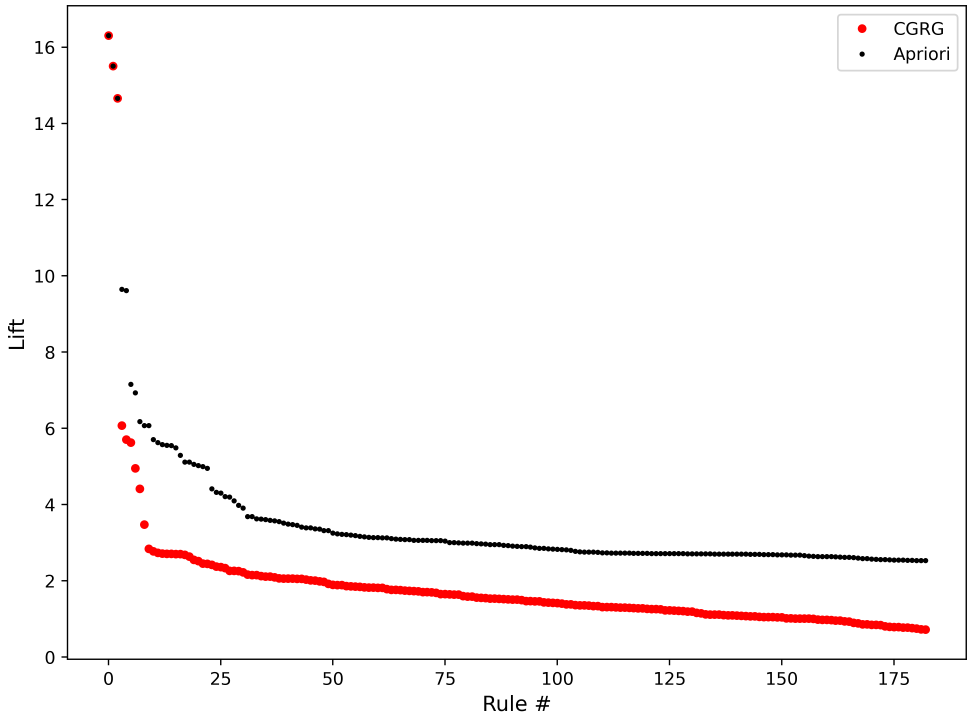

Figure 4.9: Ranked lift scores for both the CGRG and Apriori Algorithm

The figure is similar to the ranked support scores in Figure 4.7, however the disparity at the elbow of the curves is much greater. This may indicate that the antecedent may give a higher rise to the confidence of a rule when the composition of the antecedent and/or the consequent consists of items from multiple clusters. To illustrate the high lift rules our algorithm has failed to capture, the first five multi-element rules generated by both algorithms and ranked by lift scores have been detailed below:

Antecedent	Consequent	Support	Confidence	Lift	Type
{canisters,lubricant}	{filters}	0.0014	0.8128	16.3071	Apriori
{canisters}	{filters,lubricant}	0.0014	0.7250	15.5054	Apriori
{canisters}	{filters}	0.0014	0.7308	14.6611	Apriori
{chips,filters}	{juices and soft drinks,lubricant}	0.0011	0.3528	9.6487	Apriori

{filters,the bakery}	{juices and soft drinks,lubricant}	0.0011	0.3515	9.6143	Apriori

Table 4.4: Apriori Rules ranked by Lift

Antecedent	Consequent	Support	Confidence	Lift	Type
{lubricant,canisters}	{filters}	0.0014	0.8128	16.3071	intra-cluster
{canisters}	{lubricant,filters}	0.0014	0.7250	15.5054	intra-cluster
{canisters}	{filters}	0.0014	0.7308	14.6611	intra-cluster
{filters,water}	{lubricant,chewing gum and candy}	0.0020	0.3525	6.0734	intra-cluster
{lubricant,pickets}	{filters}	0.0013	0.2844	5.7052	intra-cluster

Table 4.5: CGRG Rules ranked by Lift

The first three rules with the highest lift are present in the rulesets of both algorithms, however the CGRG did not capture the fourth and fifth rule in Table 4.4.

Chapter 5

Conclusions

5.1 Achievements

In this paper, we have introduced a new algorithm for association rule generation - the CGRG algorithm: an extension of the methodology proposed by **(Valle et al. 2018)** that also employs the techniques and principles outlined in **(Agrawal and Srikant 1994)** and **(Dongen 1969)**. The CGRG algorithm produces bi-cluster rules where the antecedent and consequent are from separate clusters, and intra-cluster rules where both the antecedent and consequent originate from the same cluster. In doing so, it appears to capture the high-support rules from all possible rules within any support and confidence constraints. We have also theorized that association rules that are bi-cluster or intra-cluster tend to have a higher support score than those rules that are not. The rules generated by the CGRG algorithm seem to also follow general intuition (see: Table 4.2) such as {`cigarettes`} → {`chewing gum and candy`}, and {`filters`} → {`lubricant`}, further reinforced by their existence in the rules generated by the Apriori algorithm as well.

5.2 Limitations

Dataset Bias

All analysis was conducted on the topological structures generated from a singular dataset. This may not be wholly representative of the rules the CGRG algorithm may generate, and perhaps a different dataset may have yielded different results.

Clustering

As seen in Figure 4.6 on Page 30, the Markov Clustering configuration yielded some clusters that seem appropriate, yet some (for example, the largest) were perhaps too broad to classify under a single cluster.

Rule Structure

The CGRG algorithm's key feature is both its greatest strength and drawback. Due to the fact that

our algorithm only selects those rules where the items in a given itemset are from the same cluster, the algorithm overlooks rules where the antecedent and consequent have a composition of items from multiple clusters. Figures 4.7 and 4.9 indicate that the CGRG algorithm did not generate several higher-value rules present at the elbow of the Apriori rule curve. However, we have also concluded via Figure 4.7 that this constraint captures a majority of high-value (i.e. high support) rules and does not capture the lower-value rules within the support and confidence constraints.

5.3 Future Works

As mentioned, a key limitation of our analysis is that it only applies to the dataset we used. Future projects involving different datasets may produce new insights into the advantages and/or limitations of the algorithm. Additionally, alternative clustering algorithms may be considered as a replacement for the Markov Clustering algorithm.

References

Abdelnabi, **Omar Khaled Abdelaziz** (2018). *Minimum Spanning Tree*. URL: https://www.hackerearth.com/practice/algorithms/graphs/minimum-spanning-tree/tutorial/.

Agrawal, **Rakesh**, **Tomasz Imieliński**, et al. (1993). "Mining Association Rules between Sets of Items in Large Databases". In: *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. Association for Computing Machinery, pp. 207–216. ISBN: 0897915925. DOI: 10.1145/170035.170072. URL: https://doi.org/10.1145/170035.170072.

Agrawal, **Rakesh** and **Ramakrishnan Srikant** (1994). "Fast Algorithms for Mining Association Rules". In: *Proceedings of the 20th International Conference on Very Large Databases*, pp. 487–489.

Brin, **Sergey** et al. (June 1997). "Dynamic Itemset Counting and Implication Rules for Market Basket Data". In: *SIGMOD Rec.* 26.2, pp. 255–264. ISSN: 0163-5808. DOI: 10.1145/253262.253325.

Cover, **Thomas M** and **Joy A Thomas** (2006). *Elements of information theory*.

Dijkstra, **E. W.** (1959). "A note on two problems in connexion with graphs". In: *Numerische Mathematik* 1. ISSN: 1. DOI: 10.1007/BF01386390.

Dongen, **Stijn Marinus van** (Sept. 1969). "Graph Clustering by Flow Simulation". Dutch. Dissertation. Universiteit Utrecht. Chap. 1, p. 5. URL: http://www.library.uu.nl/digiarchief/dip/diss/1895620/full.pdf.

Ernest C Jr, **Davenport** and **Nader A. El-Sanhurry** (1991). "Phi/Phimax: Review and Synthesis". In: *Educational and Psychological Measurement* 51.4, pp. 821–828. DOI: 10.1177/001316449105100403.

Han, **Jiawei** et al. (May 2000). "Mining Frequent Patterns without Candidate Generation". In: vol. 29. 2. New York, NY, USA: Association for Computing Machinery, pp. 1–12. DOI: 10.1145/335191.335372. URL: https://doi.org/10.1145/335191.335372.

Jarník, **Vojtěch** (1930). "O jistém problému minimálním". In: *Práce Moravské Přírodovědecké Společnosti*, pp. 57–63.

Kim, **Jun Woo** (2017). "Construction and evaluation of structured association map for visual exploration of association rules". In: *Expert Systems with Applications* 74. DOI: 10.1016/j.eswa.2017.01.007.

Kruskal Jr, **Joseph Bernard** (1956). "On the Shortest Spanning Subtree of a Graph and the Traveling Salesman Problem". In: *Proceedings of the American Mathematical Society* 7, pp. 48–50. DOI: 10.1090/S0002-9939-1956-0078686-7.

marcio486 (2020). *Sales data for a chain of Brazilian stores*. Kaggle. URL: https://www.kaggle.com/marcio486/sales-data-for-a-chain-of-brazilian-stores.

Markov, **A.A.** (2006). "Extension of the law of large numbers to quantities, depending on each other (1906). Reprint." rus. In: *Journal Électronique d'Histoire des Probabilités et de la Statistique [electronic only]* 2.1b, Article 10, 12. URL: http://eudml.org/doc/128778.

Pearson, **Karl** (1895). "Note on Regression and Inheritance in the Case of Two Parents". In: *Proceedings of the Royal Society of London* 58, pp. 240–242. ISSN: 03701662.

Pennsylvania, **Junita College** (2020). *Generating Association Rules*. Junita College Pennsylvania. URL: http://faculty.juniata.edu/rhodes/ml/assocRules.html.

Prim, **Robert C.** (1957). "Shortest Connection Networks and Some Generalizations". In: *Bell System Technical Journal* 6 (6), pp. 1389–1401. DOI: 10.1002/j.1538-7305.1957.tb01515.x.

Python (2020). *Python 3.8.3*. URL: https://www.python.org/downloads/release/python-383/.

Rust (2020). *Rust 1.49.0*. URL: https://www.rust-lang.org.

Valle, **Mauricio A.** et al. (May 2018). "Market basket analysis: Complementing Association Rules with Minimum Spanning Trees". In: *Expert Systems with Applications* 97, pp. 146–162. DOI: 10.1016/j.eswa.2017.12.028.

Zekic-Susac, **Marijana** and **Adela Has** (Oct. 2015). "Discovering market basket patterns using hierarchical association rules". In: *Croatian Operational Research Review* 6, pp. 475–487. DOI: 10.17535/crorr.2015.0036.

Zhong, **Caiming** et al. (2015). "A fast minimum spanning tree algorithm based on K-means". In: *Information Sciences* 295, pp. 1–17. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2014.10.012. URL: http://www.sciencedirect.com/science/article/pii/S0020025514009943.