

# Market Basket Analysis and Classification using K-Means Clustering Techniques

Muhammad Umar Khan and Noah Adams

Final Project Report

Dr. Ranga

April 8th, 2024

# Abstract

This project explores the enhancement of Market Basket Analysis through the integration and use of the K-Means clustering algorithm, addressing the inefficiency of traditional association rule mining techniques on large data sets. Market Basket Analysis is a data mining technique which aims to help businesses understand customer purchase patterns and to uncover correlations between frequently purchased items in a transaction. However, association rule mining is very computationally expensive and to us has practical limitations when dealing with companies with an extensively large amount of transaction data.

Our research has led us to propose a pre-processing step using the K-Means Clustering algorithm to combat the challenge of impractical association rule mining with large transactional data sets. Based on certain metrics, we can segment data into various clusters to filter out less meaningful information, thus reducing input size for the association rule mining process. We present a novel approach, using the revenue of each product and the revenue of the products purchased alongside it, clustering out the products belonging to the cluster which minimizes these values. We then compare our approach to previous work in this domain and the association rule baseline using a novel value formula. We conclude that our method shows a substantial improvement in regards to efficiency while also improving value generated across five different input sizes on our test dataset.

# Table of Contents

List of Figures/Tables - Page 3

1. Introduction - Page 3

2. Literature Review - Page 4

3. Methodology - Page 9

4. Basic System Architecture - Page 12

5. Implementation - Page 19

6. Testing and Evaluation - Page 25

7. Results - Page 26

8. Conclusion - Page 29

References - Page 30

Appendices - Page 32

- Appendix A - Page 32
- Appendix B - Page 35

# List of Figures/Tables

- Figure 1. Apriori Algorithm - Page 5
- Figure 2. apriori-gen - Page 6
- Figure 3. Lloyd's Algorithm - Page 8
- Figure 4. Association value estimation example - Page 11
- Figure 5. Part 1 of Sequence Diagram - Page 13
- Figure 6. Part 2 of Sequence Diagram - Page 14
- Figure 7. Part 3 of Sequence Diagram - Page 15
- Figure 8. Activity Diagram - Page 17
- Figure 9. Component Diagram - Page 19
- Figure 10. Execution time for each approach - Page 27
- Figure 11. Maximum values generated for top 100 rules - Page 28
- Table 1: Example of Itemset 1 - Page 20

## 1. Introduction

The main goals of a business are to make revenue and profit from producing and/or selling products and/or services. Customers who buy these products and services and thus are what determines whether a business struggles or flourishes. With many large companies such as Amazon, Microsoft, Apple and Tesla serving many customers, there are billions of transactions being handled worldwide every day. As a result, it can be very valuable for companies to focus on analyzing customers' purchase patterns. Knowing what items customers buy together, can tell businesses what connections there are between certain products, which they can then take advantage of to try and increase their sales via marketing strategies, leveraging some items to sell others. Due to the vast amounts of data being worked with, learning these connections requires automated tools which can be used to help determine connections between products with ease.

This is where we introduce Market Basket Analysis, a data mining technique that uses association rule mining to make connections [1] using the frequency of which multiple items are purchased together inside an individual transaction. This data can then help inform businesses to make better decisions with product placement and marketing and show them how they can increase their sales revenue with revised strategies.

However, Association Rule Mining, the core of Market Basket Analysis, has a major challenge, efficiency. The goal of Association Rule Mining, known as the Association Rule Mining problem, is to find all supersets of differing items in transactions belonging to a given itemset that meet a minimum frequency, titled support. However, this means that Association Rule Mining algorithms often have to generate and check a large number of rules. In fact in

2004, Yang proved that there is no polynomial time algorithm which can solve the Association Rule Mining Problem. As a result, all accurate Association Rule Mining algorithms are highly inefficient to employ. This is a problem as with an expansive scale of retail data held by many large companies coupled with the inefficiencies seen in Association Rule Mining, there is a major practical issue when implementing Market Basket Analysis.

In this paper, we propose an approach which uses the K-Means Clustering algorithm along with Association Rule Mining to determine the rules and connections needed to identify customer's behavior and purchase patterns. We use K-Means to filter out items which are unlikely to contribute to rules which are valuable to the business. This in turn means that the Association Rule Mining algorithm will be more efficient and scalable to handle large datasets more easily as the unneeded, invaluable data is being removed prior to its use by these expensive algorithms. Due to K-Means inefficiency, Lloyd's Approximation algorithm is used. Lloyd's algorithm clusters data points into groups represented by centroids [2]. The main difficulty is choosing what data to use to cluster the items. We discuss 2 different approaches. The first is a previously proposed approach and the second is our own novel approach based upon a new method for calculating value. Both approaches are compared to the baseline without K-Means in regard to value of rules generated and efficiency. The first approach focuses on clustering items based on frequency and price, and the second approach focuses on the revenue generated by each item and the items it is purchased with. The other difficulty with using the K-Means algorithm is determining the ideal  $k$  value to be used for implementation. We use both the Dunn Index and the elbow method to determine this. In the end, our novel approach proves very promising as the execution time of running the program is drastically reduced compared to the baseline on 5 different input sizes of a real world itemset while also achieving higher value top rules. Thus our approach proves to be more efficient, scalable and higher performing than previous approaches.

In the following sections we will review literature related to previous works that need to be understood to have full comprehension of this paper. This includes Association Rule Mining, K Means Clustering, their algorithms and how K means can be used to reduce Association Rule Mining algorithms computational expense. We then cover the methodology and rationale explaining our approach and experiments conducted. We will go over the basic system architecture of our implementation covering detailed sequence, activity and component diagrams before going into implementation specifics. We show our testing and evaluation methods and then wrap up with our results and conclusions of our experiments.

## 2. Literature Review

In this section we cover the Association Rule Mining problem as well as its time complexity. We introduce the importance of the Apriori algorithm, the details of its functionality and efficiency. We review the K-means clustering problem as first outlined by Lloyd and its computational expense. We detail Lloyd's algorithm, an approximation of K-means and why it is so commonly used. We then finish with exploring the previous work done using clustering to filter out unimportant association rules.

## 2.1. Association Rule Mining

The association rule mining problem was first proposed by Agrawal, Imielinski and Swami in 1993. It can be formally defined as follows: Let  $I = I_1, I_2, I_3, \dots, I_m$  be a set of distinct binary attributes called items. Let  $T$  be a set of transactions with each transaction,  $t$ , being represented by an  $m$  dimensional binary vector.  $t[k] = 1$  if  $I_k$  was purchased in the transaction  $t$ ,  $t[k] = 0$  otherwise. Let  $X$  be a set of some of the items in  $I$  where a transaction  $t$  satisfies  $X$  if and only if for all items  $I_k \in X$ ,  $t[k] = 1$ . An association rule is defined as  $X \Rightarrow I_j$  where  $j \neq k$  for all items  $I_k \in X$ . An association rule is satisfied by a transaction  $t[k]$  if and only if it satisfies  $X$  and  $t[j] = 1$ . The support of an association rule is the number of transactions in  $t$  which satisfies it. Generate all association rules where  $X \in S$ , the superset of  $I$  and  $I_j$  belongs to all  $I$  that satisfies the given minimum support [3]. The association rule mining problem was proven by Yang in 2004 to be NP-Hard [4] meaning that no polynomial time algorithm can be created to solve the problem. Resulting in all Association rule mining algorithms being very inefficient.

Apriori is an association rule mining algorithm given by Agrawal and Srikant in 1994 [5]. It is popular due its ease of implementation, scalability, efficiency and age. It is a relatively simple algorithm which produces results which are straightforward to interpret. Apriori is relatively efficient compared to other top performing Association rule mining algorithms across all input sizes. It was the first efficient Association rule mining algorithm created only one year after the association rule mining problem, allotting ample time for its study and testing. Apriori is given in the first two figures. Figure 1 outlines the Apriori algorithm's main body while figure 2. describes the apriori-gen subfunction it uses.

```

1)  $L_1 = \{\text{large 1-itemsets}\};$ 
2) for (  $k = 2; L_{k-1} \neq \emptyset; k++$  ) do begin
3)    $C_k = \text{apriori-gen}(L_{k-1});$  // New candidates
4)   forall transactions  $t \in \mathcal{D}$  do begin
5)      $C_t = \text{subset}(C_k, t);$  // Candidates contained in  $t$ 
6)     forall candidates  $c \in C_t$  do
7)        $c.\text{count}++;$ 
8)   end
9)    $L_k = \{c \in C_k \mid c.\text{count} \geq \text{minsup}\}$ 
10) end
11)  $\text{Answer} = \bigcup_k L_k;$ 

```

*Figure 1. The Apriori algorithm*

- 1) Insert into  $C_k$
- 2) Select  $p.item_1, p.item_2, \dots, p.item_{k-1}, q.item_{k-1}$
- 3) From  $L_{k-1} p, L_{k-1} q$
- 4) Where  $p.item_1 = q.item_1, \dots, p.item_{k-2} = q.item_{k-2},$   
 $p.item_{k-1} < q.item_{k-1}$
- 5) For all itemsets  $c \in C_k$  do
- 6)     For all  $(k-1)$ -subsets  $s$  of  $c$  do
- 7)         If  $(s \notin L_{k-1})$  then
- 8)             Delete  $c$  from  $C_k$ ;
- 9) Return  $C_k$

Figure 2. apriori-gen

The Apriori Algorithm takes  $L_1 = \text{large } 1 - \text{itemsets}$  and  $D = \text{the set of transactions}$  as input and returns all supersets of  $L_1$  which contain at least 2 items and meet the minimum support threshold. The *large 1 - itemsets* taken as input is a set where each element is set containing only an element from the set of all distinct items belonging to a transaction in  $D$ . More formally, this can be described as  $\{I\} \in L_1$  if and only if  $\exists q, t_q \in D \mid I \in t_q$  and  $\forall k, k \in L_1 - \{I\}, \{I\} \neq k$ . In lines 5 and 6, the apriori algorithm loops through all candidates in  $L_k$  which are subsets of a given transaction. Candidates, represented by  $C_k$ , are the superset of all items in  $L_1$  whose subsets all meet the minimum support. Line 7 adds 1 to the support value of these candidates. Lines 4 and 8 contain the bounds of a loop that repeats this process for all transactions in  $D$ . In line 9, all candidates which meet a minimum support value are copied into  $L_k$ . In line 3 new candidates are generated using the apriori-gen function from  $L_{k-1}$ . In lines 2 and 10  $k$  is iterated until  $L_{k-1} = 0$ , i.e. none of the candidates generated by apriori-gen in a pass meet the minimum support value. Lines 1 and 11 represent the aforementioned input and output respectively.

Apriori-gen takes  $L_{k-1}$ , the set of all candidates containing exactly  $k - 1$  distinct items, which meet the minimum support. It returns  $C_k$ , the set of all candidates containing exactly  $k$  distinct items whose subsets all meet the minimum support. Lines 1-4 generate all sets of  $k$  items where 2 different sets from  $L_{k-1}$  share their first  $k - 2$  items and add them to  $C_k$ . Lines 5-8 iterate through all  $k - 1$  item subsets of all  $c$  belonging to  $C_k$  and check if they belong to  $L_{k-1}$  and thus meet the minimum support. If any do not,  $c$  is removed from  $C_k$ . Line 9 returns

$C_k$ . The Apriori algorithm has a worst case time complexity of  $O(2^m)$  due to the power set  $2^{m-1}$  representing the maximum total number of candidates that could exist.

## 2.2. K-Means Clustering

In 1957, Lloyd outlined Lloyd's algorithm which takes in an input of data points, a number of centroids,  $k$  and returns  $k$  centroids. Each centroid represents the mean of a cluster where the sum of squared distances between points within a cluster and their centroid is approximated to a local minima [6]. This algorithm laid the foundation for the k means clustering problem. The k means clustering problem can be formally defined as given a set of data points,  $S$  and a number of clusters,  $k$ , find  $k$  subsets,  $Q$  of  $S$  such that each  $x \in S$  belongs to exactly one  $q \in Q$  and  $p_k$  is the mean of all  $x$  belonging to  $q_k$  for each  $q_k$ . Assign each  $x$  to a  $q_k$  such that the sum of euclidean distances between  $x$  and  $p_k$  over all  $x \in S$  is minimized. The K- means problem proven to be NP-Hard even given a fixed number of dimensions by Mahajan, Nimbhorkar and Varadarajan in 2009 [7]. However, Inaba, Katoh and Imai proved that K-means can be solved in polynomial time given both a fixed number of dimensions and the number of clusters  $K$ . Precisely, it can be solved in  $O(n^{dk+1})$  where  $d$  is the number of dimensions [8].

In practice, Lloyd's algorithm is often used for clustering due to its simplicity and consistency in converging to approximate solutions of the K-means problem quickly. Lloyd's algorithm was proved by Hartigan and Wong in 1979 to have a time complexity of  $O(nkdi)$  where  $n$  is the number of data points to be clustered,  $k$  is the number of clusters,  $d$  is the number of dimensions and  $i$  is the number of iterations before convergence [9]. If  $k$  and  $d$  are constants the time complexity can be trivially reduced to  $O(ni)$ . Pseudocode of Lloyd's algorithm with a fixed number of clusters and dimensions is given in figure 3. Lloyd's algorithm takes input of  $k$  as the number of clusters to be generated,  $d$  as the number of dimensions that represent the data and  $X$  as the set of all data points to be clustered where each data point is represented by a  $d$  dimensional vector. It returns  $C$ , the set of centroids where the sum of squares distances between each point and its nearest centroid as measured by Euclidean distance has converged to a local minima. In lines 2 and 3, each centroid is set equal to a unique random data point in  $X$ . Lines 5 and 6 assign each point in  $X$  to their nearest centroid. Lines 7 and 8 recompute the centroids as the means of the points assigned to them. The last two steps are repeated until the algorithm converges via the loop in line 4. At this point  $C$  can no longer improve and is returned as the algorithm's output.

- 1) Create  $k$  clusters  $c \mid c \in C$
- 2) For each  $c$  do
- 3)  $c = \text{some random } x, x \in X \mid \text{each } c \text{ has a unique value}$
- 4) While  $C$  has not yet converged
- 5) For each  $x$  do
- 6)  $x$  is assigned to  $c$  where  $\sqrt{\sum_{i=1}^d (x^i - c^i)^2}$  is minimised
- 7) For each  $c$  do
- 8) Update  $c$  to be the mean of all  $x$  assigned to  $C$
- 9) Return  $C$

*Figure 3. Lloyd's algorithm*

Lu and Zhou have proved that Lloyd's algorithm converges to an exponentially small error after  $\log(n)$  iterations. Therefore, Lloyd's algorithm can be modified to stop upon reaching a threshold of  $\log(n)$  iterations while maintaining a minimal reduction in its performance [10]. This can be done by replacing line 4 in figure 3 with “While  $C$  has not yet converged and less than  $\log|X|$  iterations of this loop have occurred”. Consequently,  $i \leq \log n$  meaning that this variant of Lloyd's algorithms worst case time complexity is  $O(n \log n)$ .

### **2.3. Reducing Association Rule Mining computation time using $K$ means clustering**

In 2021 Kanhere, et al. proposed a novel approach which aims to reduce the computational expense of Association rule mining algorithms by first performing k-means clustering to find and remove data that is unlikely to contribute to any association rules found by association rule mining. This was done by calculating the frequency of each distinct item contained within the set of transactions, i.e. how many association rules it appears in as well as their average prices. K-means clustering is then performed on the set of distinct items using frequency and average price as dimensions with  $k = 4$ . They tested this method on the Glantus database, where one cluster was found to populate approximately 93% of the data. Data points belonging to that centroid had very low frequency values. They were all removed from the set of transactions with association rule mining being performed on the resulting set. This was compared to the traditional method of running association rule mining directly on the initial dataset. They found that their execution time decreased while maintaining all of their generated association rules [11].

### 3. Methodology

In this section we detail exactly what problem this paper aims to solve as well as its importance. We demonstrate how we aim to build and improve upon previous works and approaches done in this subject area. We specify our novel value formula and why it is an accurate depiction of the problem and useful for solving it. We also outline our project's scope and objectives. We conclude with the tools and frameworks we used to accomplish this research and the justifications for their uses.

In 2022, credit cards alone accounted for an estimated 678 billion transactions globally. That is equal to approximately 21,510 transactions every second [12]. The amount of transactional data being produced has reached a size which is far too large to manually analyze for many businesses. As a result, businesses have started taking an automated approach, namely Market Basket Analysis which harnesses Association Rule Mining to find interesting connections which data analysts can leverage for business advantages. The issue with using Association Rule Mining to solve this issue of analyzing great purchase volume is Association Rule Mining itself is very inefficient. It is an NP-Hard problem and thus all algorithms solving it are very slow. Apriori for instance, a popular algorithm which was one of the first to solve this problem has a worst case time complexity of  $O(2^m)$  where  $m$  is the number of distinct items contained within the set of transactions. Therefore, in the worst case a business selling only 1000 unique products would need to make at least  $2^{1000}$  operations to calculate all Association rules meeting the minimum support using Apriori. On an average modern processor in 2024 which is 1-3.8 GHz [13], this would take over  $10^{284}$  years. While a worst case scenario is unlikely to occur, execution is still usually very costly causing many serious practical limitations to running association rule mining algorithms on market basket analysis.

In 2021 Kanhere, et al. proposed a method which aims to reduce time complexity by using clustering to filter out data unlikely to contribute to Association rules prior to using Association rule mining due to having low average prices, frequency of purchase and thus low revenue. Clustering groups similar items together, so this approach filters out the cluster containing the group containing low revenue generating items. However, this method only marginally reduced execution time and was solely designed to work for a specific dataset [11]. Additionally, how insightful a data analyst finds a given association rule is subjective and highly dependent upon circumstantial information which cannot be objectively deciphered. For instance, the insightfulness of an association rule is entirely dependent upon a data analyst being previously unaware of the connection. Consequently, metrics measuring insightfulness in Market Basket Analysis can only be estimates and are often based upon several assumptions. For example, the support metric used by the Apriori Algorithm relies entirely upon the assumption that the insightfulness of a connection is proportional to its frequency.

Due to the uncertainty pertaining to how valuable a given association rule is to a business, it makes sense to instead determine how valuable it could be. The application of Market Basket Analysis is typically to leverage an association rule in some way that makes it more likely for customers that purchase one item in a basket, to also purchase the others. For instance, if a

connection between Root Beer and apples is found, placing them physically nearby in a store makes it more likely for a given customer who travels to the section to buy one of the items, to also see the other and thus buy it. Therefore, the value of a connection can be defined as the increase in company revenue brought by the association between items in a Market Basket. Since this association is typically only leveraged when a customer buys one item in a Market Basket, the maximum revenue generated by a purchase due to the association can be estimated as the average revenue generated per purchase of each item in the Market Basket aside from the item originally sought out by the customer. Therefore, the maximum possible revenue generated overall by the association can be estimated as the sum of the average revenue per purchase of every other item in the Market Basket given each individual item purchased in the Market Basket, across all individual items purchased in the Market Basket.

Formally, this can be defined as given  $I$ , calculate  $\sum_{j \in I} ((\sum_{k \in I | k \neq j} i_k^r / i_k^p) \times i_j^p)$  where for each  $i \in I$ ,  $i^r$

represents the total revenue generated in purchases containing item  $i$  and  $i^p$  represents the total number of units of  $i$  purchased in the dataset. This is the method we use to calculate the value of an association rule in this paper. An example, outlining the steps for calculating the value is outlined in figure 4.

Rule: {Bread, Milk} => {Butter}

Items in Rule (I): {Bread, Milk, Butter}

|                                                                                                                                                                                                                                                             | Bread                                                                                                                                 | Milk                                                                                                                                   | Butter                                                                                                                          |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Total revenue generated in purchases containing item: $i_{item}^r$                                                                                                                                                                                          | $i_{Bread}^r = \$15\ 000$                                                                                                             | $i_{Milk}^r = \$12\ 000$                                                                                                               | $i_{Butter}^r = \$8\ 000$                                                                                                       |
| number of units of $i$ purchased: $i_{item}^p$                                                                                                                                                                                                              | $i_{Bread}^p = 5000$                                                                                                                  | $i_{Milk}^p = 3000$                                                                                                                    | $i_{Butter}^p = 4000$                                                                                                           |
| Average revenue generated per item: $i_{item}^r/i_{item}^p$                                                                                                                                                                                                 | $i_{Bread}^r/i_{Bread}^p = \$3$                                                                                                       | $i_{Milk}^r/i_{Milk}^p = \$4$                                                                                                          | $i_{Butter}^r/i_{Butter}^p = \$2$                                                                                               |
| Average revenue generated per item by other items in the rule:<br>$\sum_{k \in I   k \neq item} i_k^r/i_k^p$                                                                                                                                                | $\sum_{k \in I   k \neq Bread} i_k^r/i_k^p =$<br>$i_{Milk}^r/i_{Milk}^p +$<br>$i_{Butter}^r/i_{Butter}^p =$<br>$\$4 + \$2$<br>$= \$6$ | $\sum_{k \in I   k \neq Milk} i_k^r/i_k^p =$<br>$i_{Bread}^r/i_{Bread}^p +$<br>$i_{Butter}^r/i_{Butter}^p =$<br>$\$3 + \$2$<br>$= \$5$ | $\sum_{k \in I   k \neq Butter} i_k^r/i_k^p =$<br>$i_{Bread}^r/i_{Bread}^p +$<br>$i_{Milk}^r/i_{Milk}^p = \$4 + \$3$<br>$= \$7$ |
| Estimated maximum potential revenue generated by this connection amongst those originally attempting to buy this item (assuming 1 of each item purchased per each item of this purchase):<br>$(\sum_{k \in I   k \neq item} i_k^r/i_k^p) \times i_{item}^p$ | $(\sum_{k \in I   k \neq Bread} i_k^r/i_k^p) \times i_{Bread}^p$<br>$= \$6 \times 5000$<br>$= \$30\ 000$                              | $(\sum_{k \in I   k \neq Milk} i_k^r/i_k^p) \times i_{Milk}^p$<br>$= \$5 \times 3000$<br>$= \$15\ 000$                                 | $(\sum_{k \in I   k \neq Butter} i_k^r/i_k^p) \times i_{Butter}^p$<br>$= \$7 \times 4000$<br>$= \$28\ 000$                      |

Therefore, the total maximum possible revenue generated overall by the association:

$$\sum_{j \in I} ((\sum_{k \in I | k \neq j} i_k^r/i_k^p) \times i_j^p) = \$30\ 000 + \$15\ 000 + \$28\ 000 = \$73\ 000$$

Figure 4. Association value estimation example

As a result, in this paper we propose a general approach which is applicable to all datasets of a particular format, aiming to vastly reduce execution time while generating more valuable Association rules. Precisely, we design and implement an algorithm which takes in any transactional database as input, uses K means to filter out items which are unlikely to contribute to any valuable association rules and uses the Apriori Algorithm to generate these rules. Our objectives for this algorithm are as follows:

1. The generation of more valuable top 100 association rules as ranked by support compared to using the Apriori Algorithm by itself on the same datasets across different databases and input sizes.
2. A vastly lower execution time compared to using the Apriori Algorithm by itself on the same datasets across different support levels and input sizes.

In objective 1 we use support to determine the top 100 rules since that is what the apriori algorithm itself uses and it makes the results consistent across different support values given to the apriori algorithm. We consider only the top 100 as the top rules are typically the ones which researchers analyze. Depending on the dataset and the support value given, rules generated by Apriori can scale beyond the thousands at which point it is impractical to analyze them all. This also ensures relevance of the results among large support values where not many rules are generated.

Programming associated with this project was completed using the Java programming language version 19 and its standard library in the September 2023 Eclipse Integrated Development Environment for Java Developers. This project was developed from the ground up without the help of external libraries to grant as much flexibility and customizability in our implementation as possible. This customizability and flexibility allowed us to easily modify Lloyd's algorithm to increase our algorithm's efficiency as outlined in the Literature Review section. Due to development of the project from scratch, any programming language with reasonably efficient implementation speed would suffice. We chose Java as it is relatively simple to use and the programming language which we were most familiar with at the time. JFrame, a container which supports the creation of graphical interface components such as buttons, text fields and cartesian graphs was used. It is included in Java's standard library and was used to plot the clusterization of items for understanding and testing purposes. It was used due to its ease of use and availability from being in Java's standard library. Version 19 of Java specifically was used due to its feature richness and support of Java Development Kit 19 in Eclipse, which newer versions of Java does not support. Eclipse was used due to its familiarity, reliability, ease of use and effective debugging tools. It is very familiar to us, being the Integrated Development Environment which we have the most experience using. Eclipse is very reliable due to it being very well tested and receiving many updates, it was released in 2001 [14] and has had 35 released versions which have been tested by over 6 million active users as of October 2023 [15] [16]. Eclipse is extremely easy to use and is consequently often used by those who are inexperienced with programming. Eclipse also gives detailed feedback of errors and marks the lines of which they occur, allowing them to be solved more easily and efficiently. The September 2023 version Eclipse was also used due its reliability and feature richness as a result of being the latest released version of Eclipse at the time. All experiments were conducted using a personal computer using an Intel(R) Core(TM) i5-10400F CPU with a 2.90GHz processor on the Windows 10 operating system.

## 4. Basic System Architecture

Diagrams (e.g., UML diagrams) depicting system components and their interactions.

This chapter is to explore the architectural framework and design of our implemented system of Market Basket Analysis. To get an overall layout or structure of the whole system with components and interactions, there are 3 main diagrams to illustrate this architecture.

## 4.1. Sequence Diagram

The sequence diagram (Figure 5-7) illustrates the interaction of the different components and methods of the system during the execution of Market Basket Analysis. The sequence of events are listed, starting from the input of data to output of association rules.

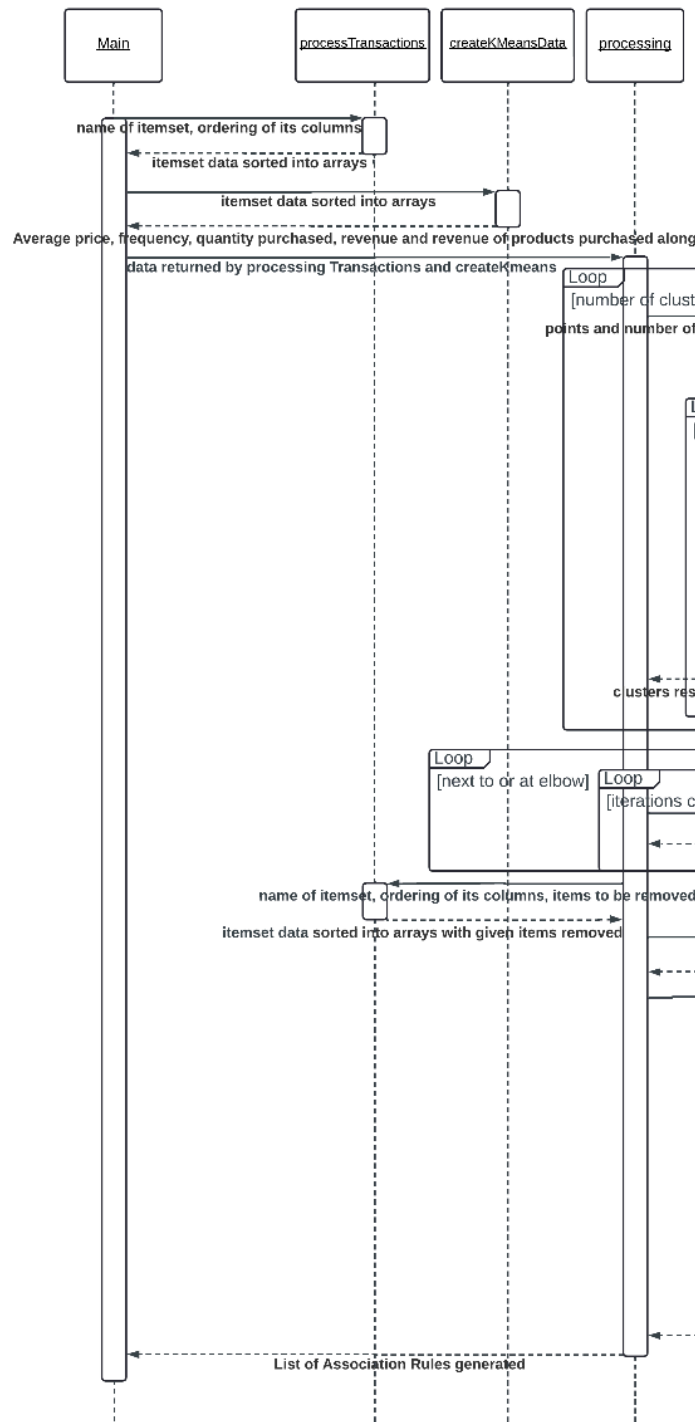


Figure 5. Part 1 of Sequence Diagram (preprocessing)

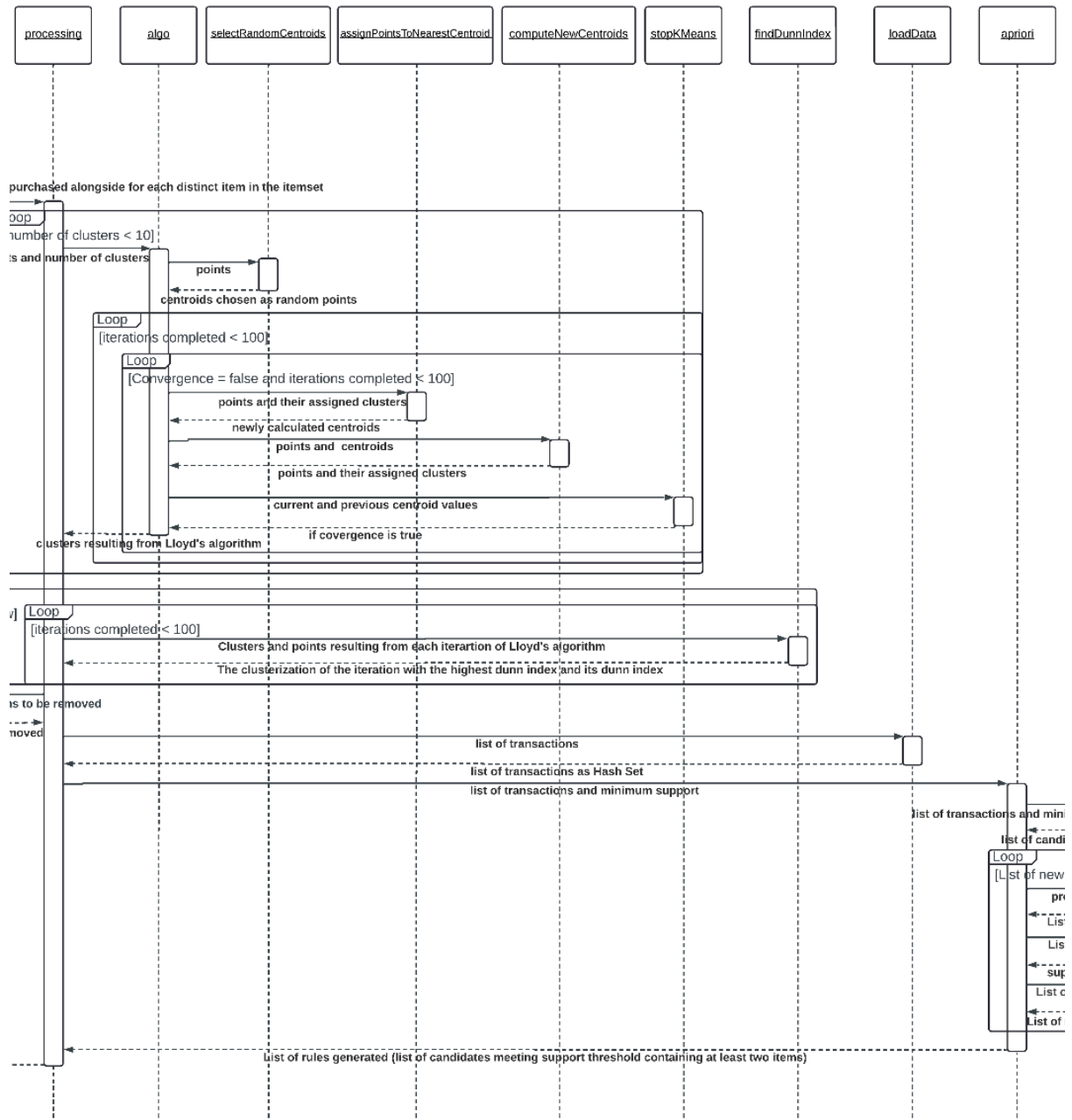


Figure 6. Part 2 of Sequence Diagram (k-means)

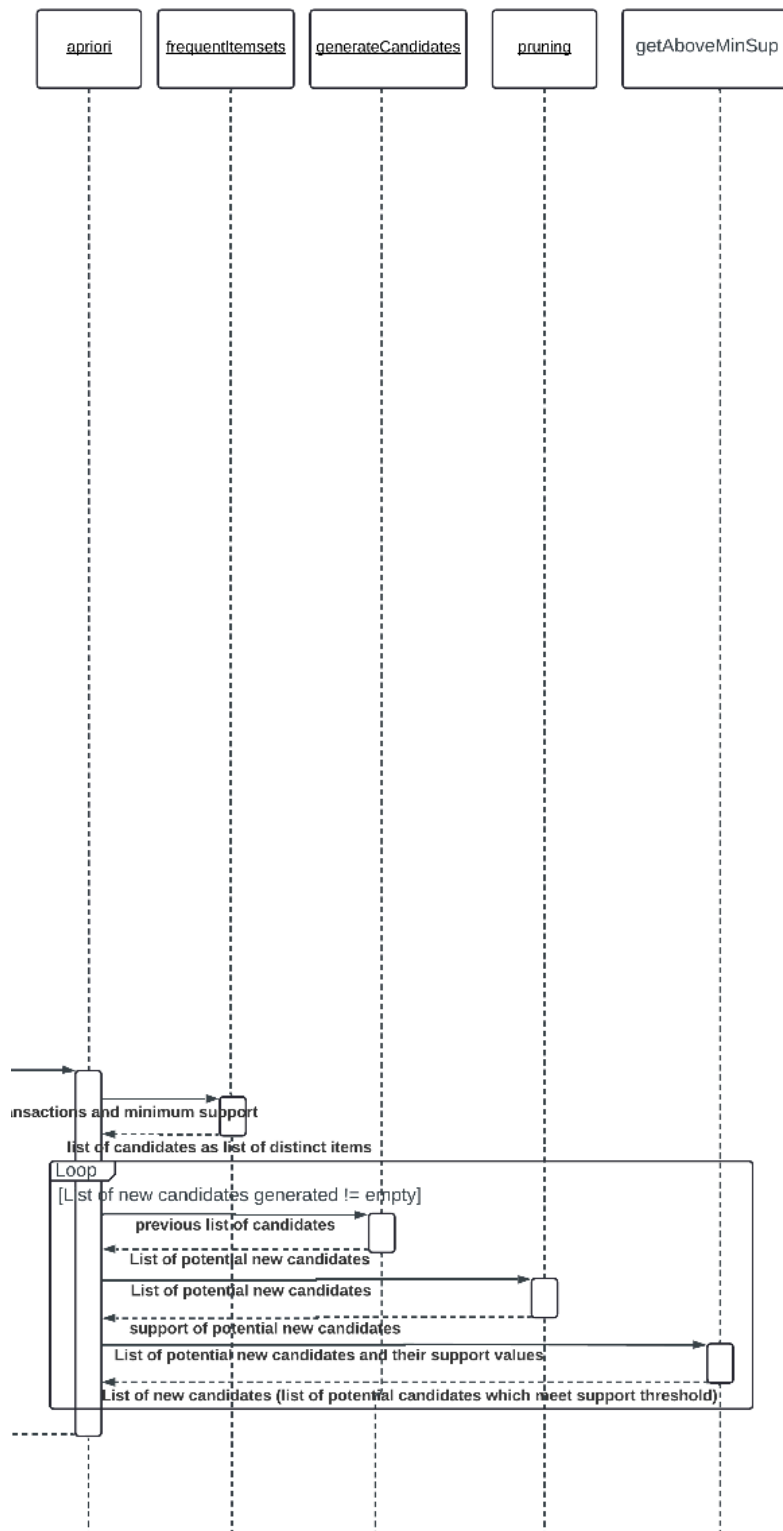


*Figure 7. Part 3 of Sequence Diagram (Apriori Algorithm)*

Each box at the top of the diagram represents the methods (and class) of this program, and the diagram conveys the interactions between the methods and each other. Although separated into parts, the full sequence diagram of figures 5-7 can be described in the following way:

- The dataset is given to processTransactions from the main to process the data. Once processing is done, arrays of itemset data will be sent to main.
- With the arrays with itemset data, it goes to createKMeansData, to turn the raw data into a format suitable for the KMeans. The data returned to main would be the headers needed for KMeans such as: average price, frequency, quantity purchased, revenue of item and revenue of products purchased alongside each distinct item.
- With this suitable format for KMeans, we now begin the actual class program to implement KMeans.
- Iterate through a loop from the number of clusters starting at 2 and iterating till 10. The data points and number of clusters are given to the algo method.
- From the algo method, points are given to selectRandomCentroids and in return, the random points centroids.
- Then we go through a loop to check how many iterations are completed (in this case, our max number of iterations to complete is 100)
- If there is no convergence and the maximum number of iterations has not been reached, then do the following:
  - Assign the points to the nearest centroid by giving the points and cluster data to assignPointsToNearestCentroid method.
  - With the new points and centroids, recompute new ones as the assigned mean of points.
  - Check if there is convergence to break the loop and stop the KMeans algorithm
- The previous step is essentially Lloyd's algorithm, and this results in new clusters to be used back in processing. This is given after the maximum number of iteration and maximum number of clusters to test has been reached
- The program will then proceed to plot the results on a graph as an elbow, as per the elbow method.
- For each iteration, the clusters and points are given to findDunnIndex to calculate the dunn index value for each iteration.
- The K-Means result that has the highest dunn index is returned and selected as the best result with the best cluster.
- The itemset and ordering of items will be given to processTransactions to remove the items that need to be removed.
- With the new list of transactions, we call loadData to process and load the data in as a HashSet to be used for the Apriori algorithm.
- The apriori method is called with the given list of transactions and a minimum support value as specified.
- From the apriori method, it gets the frequent itemsets as a list of candidates with distinct items.
- Then we have a loop that will iterate until the new generate candidate list is not empty
  - First, generate the candidates.
  - Prune the candidate set to get the potential support.
  - Process the candidate set and remove the ones that do not meet minimum support threshold.
- Once the new generate candidate list is not empty, we return the list of candidates that contains at least 2 items (these must meet minimum support threshold)
- Finally, the association rules are then generated.
- End the program

This is a detailed description of how the program is implemented and how the methods (and class) of the program are interacting with one another.

## 4.2. Activity Diagram

The activity diagram (figure 8) shows a dynamic overview of the system's workflow depicting a sequence of activities carried out during the Market Basket Analysis process. It shows a clear understanding of the process carried out for the implementation from start to finish such as setting the parameters and running the K-Means algorithm to setting up and running the Apriori algorithm to generate the result and ending the program.

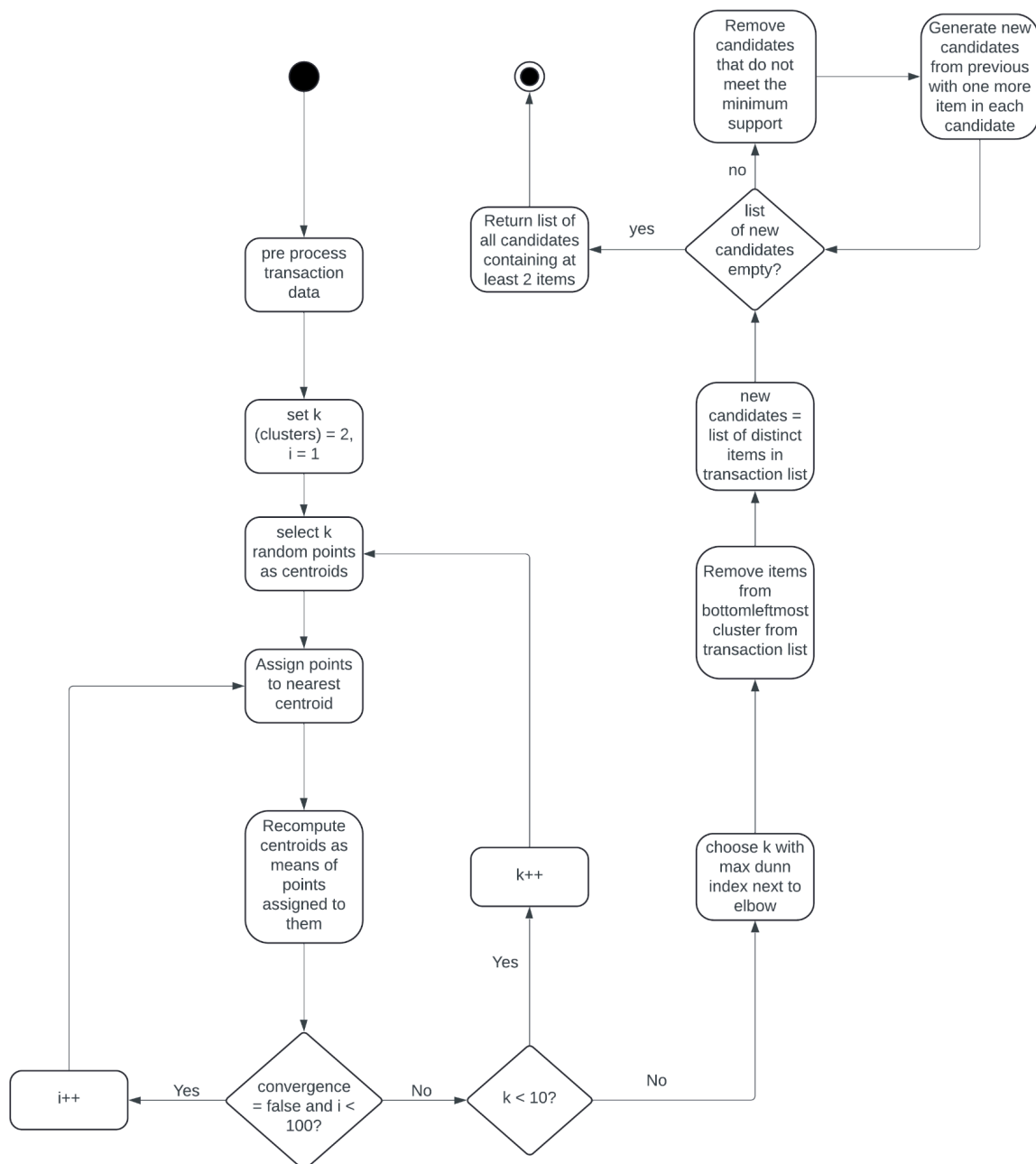


*Figure 8. Activity Diagram*

Figure 8 shows step by step process on how the implementation is done.

1. From the start point (the black circle), the transactional raw data is first preprocessed. This also includes readying the K-Means algorithm with a formatable dataset
2. We need to determine the best number of clusters for the K-Mean clustering, so the number of clusters is first set to 2, and  $i$  set to 1, where  $i$  is the current iteration.
3. Select a random amount of centroids.
4. Assign points to the nearest centroid.
5. Recompute the centroids as the mean of points assigned. This will effectively make the points in the clusters closer and compact as the centroids will be closer.
6. Then, check for if the convergence is ended (i.e. check if there's no significant change between the centroids and the points) and if we are still below the maximum number of iterations needed being 1000.
7. If it is the case, then increase  $i$ , the current number of iterations by 1, and repeat steps 4 and 5, and check for step 6 again.
8. If it is not the case, then check for the number of clusters being less than the max number of clusters we wish to test for being 10.
9. If it is the case, increase the number of clusters by 1, and repeat steps 3-6.
10. If it is not the case, then we must choose the cluster that has the maximum dunn index(i.e best clusters)
11. Remove the items from the bottom left most cluster. This is done because it is assumed to be the worst cluster.
12. Choose and create a candidate set by the list of distinct items in the transaction list
13. Check if this new candidate list is empty.
14. If it is not, then remove the candidates that do not meet the minimum support.
15. And generate new candidates from the previous list with an additional candidate and check if the new list is empty.
16. If the current list is empty, then return all candidates that contains at least 2 items
17. End

This explains the steps and how the flow of the program works as visualized by the activity diagram. Steps 2-9 is the K-Means algorithm portion while steps 12-16 is the Apriori algorithm portion. Steps 10 and 11 are used to gain the best K-Means result and remove the worst cluster in that run, which will then be used for Apriori. The reason why at the end we return the candidates with at least 2 items is because it would be depicted as a transaction, and we wish to generate association rules based on the multiple items in the transaction to find the connection.

### 4.3. Component Diagram

The component diagram (figure 9) provides a high level system structure overview. It showcases the primary components of the system and their interconnections which highlights the modular nature.

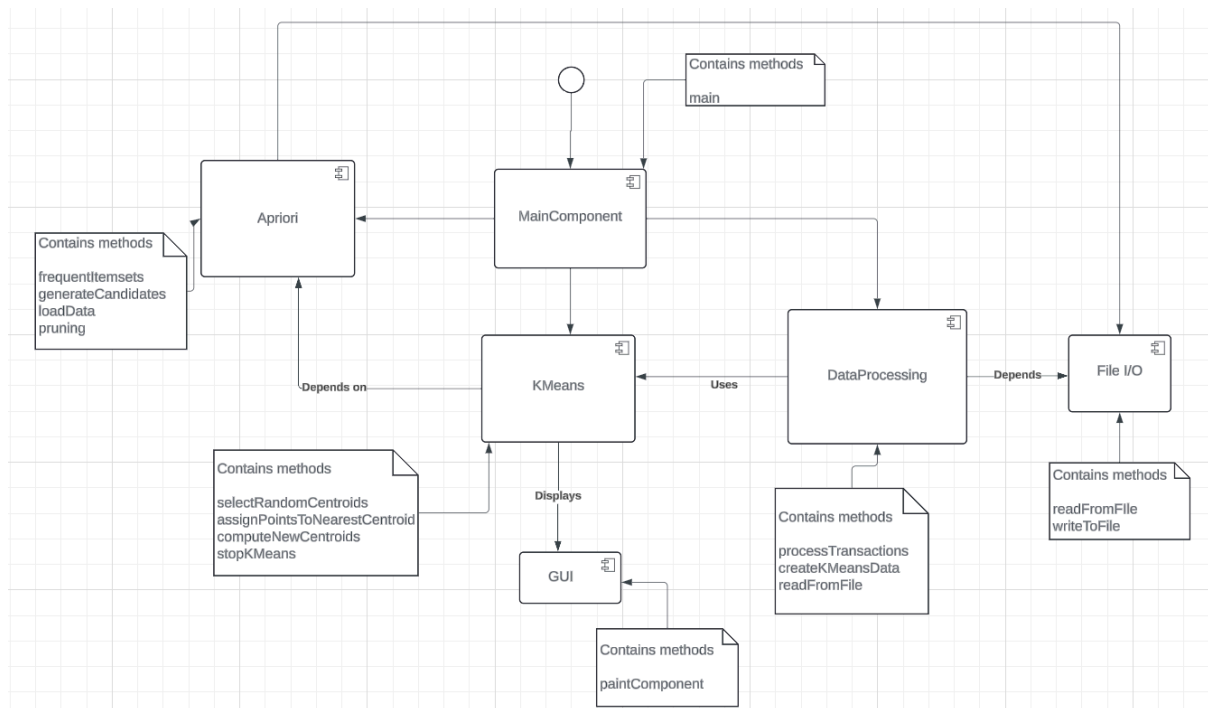


*Figure 9. Component Diagram*

This is a simple diagram that features the interactions happening with each component. Each component is listed with the methods that they are implemented with.

- From the start, the **MainComponent** initializes and interacts with all the other components (with the exception of **File I/O**).
- **Apriori**, **KMeans**, and **DataProcessing** have their own respective methods used for implementation.
- The **Main** component initializes the **Apriori**, **KMeans** and **DataProcessing** components.
- The **DataProcessing** depends on **File I/O** to retrieve and write data.
- It is then brought towards the **KMeans** algorithm to process and set the data ready to run and gain results.
- It is then branched to the **GUI** component to where the several **KMeans** runs and clusters will be displayed, to get a visualization of the clusters, to then get the best result and cluster based on the elbow method.
- After the **KMeans** is done, it is then carried to the **Apriori** component where it is highly dependent.
- After the **Apriori** is done, then it goes to **File I/O** where it will write the data into another file that contains the association rules and transaction list.

This is a more definitive view on how the various components are interacting with one another.

## 5. Implementation

Implementing both the K-Means clustering algorithm as well as the Apriori algorithm was carried out using Java. The goal for this project and research was to create a more effective version of the K-Means algorithm and Apriori algorithm to implement beside Market Basket

Analysis. Although Python contains libraries such as pandas and scikit-learn to implement machine learning techniques and algorithms, we needed to create the implementation from scratch so language was based on comfortability. For overcomplicated algorithms and programs, Java was more reasonable as it was faster than Python in terms of execution speed, performance capability and is very scalable when dealing with large datasets. In this section, there will be an in depth detail on the 3 main steps of implementing our algorithm: preprocessing, K-Means algorithm, and Apriori algorithm.

## 5.1. Preprocessing

Preprocessing the datasets was a vital part of our implementation so we can process each data header and information as shown in the datasets used in testing, and then can further go from there without experiencing issues. Table 1 displays what a transaction looked like for 1 customer where each row represents a transaction record, with each column representing a different transaction attribute. This data is a part of one of the original datasets used for this implementation. Refer to Appendix B for insight to the dataset.

| Number | Date     | Customer_ID | Transaction_ID | SKU_Category | SKU   | Quantity | Sales_Amount |
|--------|----------|-------------|----------------|--------------|-------|----------|--------------|
| 1      | 2/1/2016 | 2547        | 1              | X52          | 0EM7L | 1        | 3.13         |
| 2      | 2/1/2016 | 822         | 2              | 2ML          | 68BRQ | 1        | 5.46         |
| 3      | 2/1/2016 | 3686        | 3              | 0H2          | CZUZX | 1        | 6.35         |
| 4      | 2/1/2016 | 3719        | 4              | 0H2          | 549KK | 1        | 5.59         |
| 5      | 2/1/2016 | 9200        | 5              | 0H2          | K8EHH | 1        | 6.88         |
| 6      | 2/1/2016 | 5010        | 6              | JPI          | GVBRC | 1        | 10.77        |
| 7      | 2/1/2016 | 1666        | 7              | XG4          | AHAE7 | 1        | 3.65         |
| 8      | 2/1/2016 | 1666        | 7              | FEW          | AHZNS | 1        | 8.21         |
| 9      | 2/1/2016 | 1253        | 8              | 0H2          | 9STQJ | 1        | 8.25         |

|    |          |      |    |     |       |   |      |
|----|----------|------|----|-----|-------|---|------|
| 10 | 2/1/2016 | 5541 | 9  | N5F | 7IE9S | 1 | 8.18 |
| 11 | 2/1/2016 | 5541 | 9  | H8O | M15RG | 1 | 6.35 |
| 12 | 2/1/2016 | 7548 | 10 | N8U | UNJKW | 1 | 2.11 |
| 13 | 2/1/2016 | 7548 | 10 | JR5 | KWDJZ | 1 | 5.38 |
| 14 | 2/1/2016 | 6865 | 11 | I4Y | 8ETY5 | 1 | 8.04 |
| 15 | 2/1/2016 | 6044 | 12 | N8U | EMJ1S | 1 | 3.62 |
| 16 | 2/1/2016 | 6044 | 12 | X52 | CRK31 | 1 | 3.96 |

*Table 1. Example of Itemset 1*

After the dataset has been preprocessed with the respectful headers as indicated in the first row of the dataset, we then create the data used for the K-Means algorithm.

## 5.2. K-Means Algorithm

The K-Means algorithm is the important aspect of the implementation as this is where the data is processed into to deliver our results for connections in Market Basket Analysis. To run and implement the K-Means algorithm, we must first prepare the data information. This is how our method that we are using transforms raw transaction data into a format for K-Means:

1. Take several parameters including the array for transaction numbers, data, and indices for the specific columns or data headers in the data array. The information is aggregated based on transaction numbers
2. For each transaction in the inputted data, check if the main K-Means data array contains the transaction number.
3. If it does contain the transaction number, update the existing entry with the quantity and price for the respective fields (2 headers from the preprocessed data). Then the average price of the items bought alongside the current item will be calculated.
4. If it does not contain the transaction number, then create a new entry in the K-Means data array containing the fields: transaction number, item count, total price, total quantity (4 headers from the processed data)
5. After all the data has been processed, the average price per item and the average price of all items bought alongside is calculated for each transaction.

This will result in an array that is suited for clustering using the K-Means algorithm where each entry represents a unique transaction with aggregated information about the items purchased and their prices.

Implementing and running the K-Means to obtain the results require multiple runs as the main focus are the clusters. Running the K-Means with one number of clusters can have significantly varying results from running the K-Means with another number of clusters, higher or lower. To determine the best cluster ranges, the K-Means algorithm is run from 2 to 9 clusters to which the inertia is calculated for each set of clusters. The elbow method is then employed to situate the different results on a graph to determine the most ideal number of clusters needed for the K-Means algorithm. The code snippet below shows the implementation of the K-Mean algorithm.

```
public static void algo (int c [][],int numofClusters, double values[], int numofPoints, int
nearestCentroidopt[], double interClusterDistance[], int r, boolean dunn, int dimensions, double
minInertia[]){
    int nearestCentroid[]=new int[numofPoints];
    for(int p=0;p<r;p++){
        double [][] centroids = new double [numofClusters][dimensions];
        selectRandomCentroids(centroids,numofClusters,dimensions,values,numofPoints);
        double distance[]= new double [numofClusters];
        int distanceCount[]= new int [numofClusters];
        long inertia=0;
        double centroid[][][]= new double[numofClusters][1000][dimensions];
        boolean isdone=false;
        int q=0;

        while(isdone==false && q<1000){
            inertia=assignPointsToNearestCentroid( inertia, numofClusters, numofPoints,
dimensions, nearestCentroid, centroids,values);
            computeNewCentroids(numofClusters, dimensions, numofPoints,
nearestCentroid, values, centroids);

            for (int j=0;j<numofClusters;j++)
            {
                for (int k=0;k<dimensions;k++)
                    centroid[j][q][k]=centroids[j][k];
            }
            isdone=stopKMeans(q,centroid,numofClusters, dimensions);
            q++;
        }
        if(minInertia[numofClusters]>inertia||p==0)
            minInertia[numofClusters]=inertia;
        for (int i=0;i<numofPoints;i++)
```

```

        c[numofClusters][p][i]=nearestCentroid[i];
    }
}

```

This implementation works in the following steps:

1. Randomly select initial centroids to be used for the clusters
2. For each point, find the nearest centroid for each point and respectfully assign the point to the cluster
3. Next, update the centroids in each cluster to be the mean of the points in each cluster
4. Repeat the previous 2 steps until there is no significant change for the centroids between iterations or a maximum number of iterations have been reached

After the algorithm finishes or the maximum number of iterations have been reached, the final centroids and assignments of points to clusters are stored as a result. This algorithm is run several times from 2 clusters to 9 clusters. After every run has been completed the implementation uses the Dunn index as a means to select the result with the highest dunn value (Refer to Appendix A to understand more about the code). This means after the K-means algorithm has been run on the different cluster number ranges with the result stored, determined by the Dunn index there will be one result with the best clustering. Refer to Appendix A for better understanding of the implementation of the Dunn index.

### 5.3. Apriori Algorithm

Taking our best cluster result, there will be multiple clusters with data points, with each point having a low inertia (low distance to the centroid). With the points within the cluster being more compacted as they are closer to the centroid means the Apriori algorithm can be implemented for a more focused analysis to discover the frequent itemsets and association rules. This way, we can use our support metric to decide whether certain clusters should be kept or not due to the points within each cluster having similar characteristics or in this case, support value. Below is the code snippet for how the Apriori implementation is done.

```

public Map<Set<String>, Integer> apriori(Set<Set<String>> transactions, int minSup) {
    Map<Set<String>, Integer> freqItems = new HashMap<>();
    Map<Set<String>, Integer> minusOne = frequentItemsets(transactions, minSup);
    freqItems.putAll(minusOne);
    for (int k = 2; !minusOne.isEmpty(); k++) {
        Set<Set<String>> ck = generateCandidates(minusOne.keySet(), k);
        Set<Set<String>> freqCands = pruning(ck, minusOne.keySet());
        Map<Set<String>, Integer> lk = getAboveMinSup(freqCands, transactions,
minSup);
        freqItems.putAll(lk);
        minusOne = lk;
    }
    return freqItems;
}

```

```
}
```

This implementation works in the following way:

1. A set of transactions is taken as the input where each transaction is a set of items.
2. The frequent itemsets are found by the method identifying the itemsets that occur frequently in the transaction if there appears a minimum number of transactions specified by the minimum support value.
3. Candidate itemsets of size k based on itemsets found in the previous itemsets.
4. The candidate itemsets are then pruned to remove any infrequent subset.
5. Determines the frequency of the itemsets by checking for minimum support value.
6. These are new frequent itemsets found, so this adds to the total frequent itemsets found.
7. Steps 3 to 6 are repeated until no new itemsets can be found.

The pruning step is a vital part of the Apriori algorithm as this is how the algorithm can deduce itemsets that are important or not important to keep based on frequency. Below is the code snippet for the pruning step to check if an itemset is considered frequent.

```
public Set<Set<String>> pruning(Set<Set<String>> cK, Set<Set<String>> lk1) {
    Set<Set<String>> freq = new HashSet<>();
    for (Set<String> Cand : cK) {
        boolean isfreq = true;
        for (String can : Cand) {
            Set<String> sub = new HashSet<>(Cand);
            sub.remove(can);
            if (!lk1.contains(sub)) {
                isfreq = false;
                break;
            }
        }
        if (isfreq) freq.add(Cand);
    }
    return freq;
}
```

After the Apriori algorithm finishes, what we are expecting is a map of frequent itemsets with their support values. From this map, we are able to generate our association rules. With the association rules, businesses will be able to understand the relationship between items and find the connections they need regarding customer behavior to make better business decisions.

## 6. Testing and Evaluation

In the section we detail the items set used for our experiments. We discuss why we chose it and challenges faced with it. We talk about the approaches being compared in our experiments and our rationale for our own novel approach. Finally, we detail our experiments and provide justification for how they are run.

We tested our implementation using a dataset sourced from Kaggle, a platform used by over 17 million people for Artificial Intelligence related discussions, competitions, code and dataset sharing and collaboration [17]. The itemset is a database of 32 000 transactions taken from an international retail store chain between 2010 and 2016 [18]. It lists every individual item contained in each purchase on an individual line. Each line lists a unique transaction number pertaining to the purchase that the item on that line was bought in. All items belonging to the same purchase are contained in sequential lines. This transactional itemset was chosen for experimentation as it is an easily accessible real-world database containing every item purchased in each transaction as well as their quantities and prices. This allowed us to use item revenue as part of our clustering approach and to measure the value of association rules generated. A challenge of this dataset is that the item names given were the SKU codes. While this did allow us the certainty that no differing items were categorized under the same name, it also obscured what connections were actually being made as we do not know what any of the items actually are.

We compared apriori without k-means to our own approaches using k-means based upon efficiency and value of rules generated. In order to fairly compare the approaches, our same implementation of apriori was used for all of them. We experimented, using the metrics provided by Kanhere et al. 's as well as our own. Kanhere et al. 's approach clusters all distinct items based upon their frequency and price. This was done to remove the bottom leftmost cluster which aims to contain the items with the least revenue. Our approach clusters all distinct items based upon revenue and revenue of items purchased alongside. This aims to capture the total revenue the item generated, not only including its revenue itself, but also the revenue of the items that may have been purchased because of this item. The hypothesis of this approach is that items are at least partially responsible for the revenue generated by items purchased with them. This follows the inherent logic of association rule mining, which is that one item commonly being purchased by another means that the sale of one of the items is causing the sale of the other. Once again we filtered out the bottom leftmost cluster in an attempt to remove the items which generate the least revenue and thus are likely the least valuable to the business.

Some preprocessing was done in the implementation. This was done to allow the program to work with any itemset that has the required information. The raw data from the dataset is in a CSV file, so it was simple to handle and all that was needed was to assign the various columns to different columns in an array. Where in the row each needed part of the data was given to the function preprocessing the data for the assigning to be done effectively. The next part of the preprocessing step was to loop through all the given transactions and list every distinct item while adding up several metrics used by the approaches. These metrics are frequency, quantity, average price, revenue and revenue of items purchased alongside. This prepared the

data to be fed into k means with the item names still attached to the data so that the transactions could be processed again afterwards to remove the items to be filtered out.

We tested and compared the three approaches using the first 2000, 4000, 6000, 8000 and 10000 lines from the dataset where one line represents one distinct item from each transaction. This was done to effectively grant multiple itemsets to test as well as to directly compare how the approaches performed given differing amounts of data. We started at 2000 since less lines being used would result in less than 100 rules being generated which would not allow us to properly conduct our experiments. Also, a lower sample size can cause much more variation in the data due a small sample size. The upper bound was kept below the full dataset in order to minimize the time required to run the experiments as apriori's execution time increases exponentially as the number of lines increases. This also reduced the likelihood of any biases making the data unreliable due to only using a specific sample size. Any number of clusters between 2 and 10 may be generated depending on the itemset and how much of it is used. The number of clusters is determined using the sum of squared euclidean distances between each point and their nearest centroid at the termination of Lloyd's algorithm with respect to the number of clusters. The number of clusters where the change in slope between the previous and next number of clusters is the greatest is called the elbow. The elbow is then compared to the elbow-1 and the elbow+1 using the dunn index with the number of clusters with the largest dunn index being chosen. Lloyd's algorithm was run 100 times for each cluster size with only the clustering with the highest sum of squared euclidean distances and dunn indices being chosen to get as close to the ideal solution of the k-means clustering problem as possible. We used the formula outlined in the methodology

section  $(\sum_{j \in I} ((\sum_{k \in I | k \neq j} i_k^r / i_k^p) \times i_j^p))$  to evaluate the effectiveness of various approaches as well

as execution time in seconds to evaluate their efficiency. Execution time was measured using a timer within the program. This allowed us to track the time accurately down to the millisecond as well as to deal with an issue with JFrame. JFrame may execute a program any number of times sequentially. While we can terminate manually execution after one iteration, tracking the execution time of a single iteration necessitates the use of a timer. A separate program based upon our value formula was used to calculate the value of the top 100 rules. This program calculated the value of each item based upon the entire dataset even when only part of it was used in the calculations. This allowed our experiments to evaluate the necessity of using the entire database with this method compared to using a subsection of it which may contain all the data needed and further reduce computation time. However, due to sample size constraints this is an area which requires further research beyond the scope of this paper.

## 7. Results

In the section, we discuss the experimental results, comparing the baseline to both of our approaches. We first outlined the approaches before comparing the execution time and providing theories, which explain the results. We then do the same with the values generated. Finally, we discuss the meaning of the results, and what we believe they mean for the future of this approach.

In our experiments, we compare the baseline approach using only apriori with approaches augmented with k means. The first of these approaches clusters data based upon frequency and average price, and attempts to cluster data into different categories of revenue as proposed by Kahere et al. The second approach is our novel approach which clusters based upon revenue generated and revenue generated by other items in the same purchases in order to try to remove data which is least valuable to the business. Both methods remove the bottom left most cluster, which is seen as the least valuable as it has the lowest values in its dimensions. Figures 10 and 11 outline the execution time and maximum possible value generated with respect to the lines used for each approach.

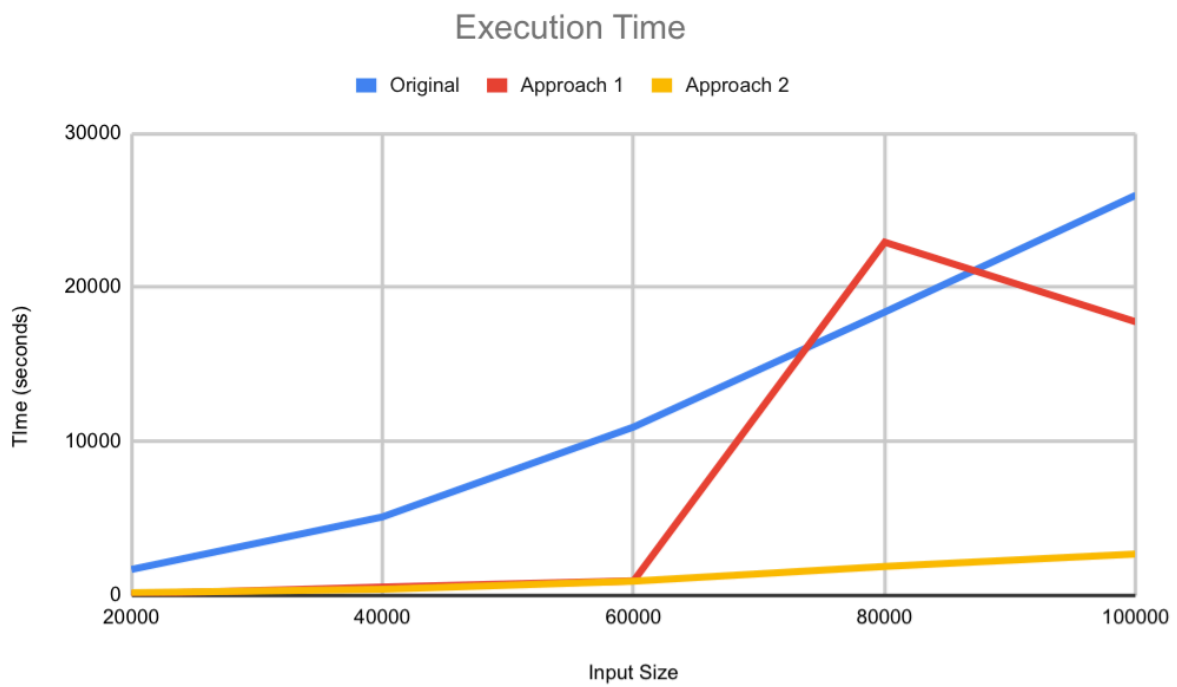


*Figure 10. Execution time for each approach*

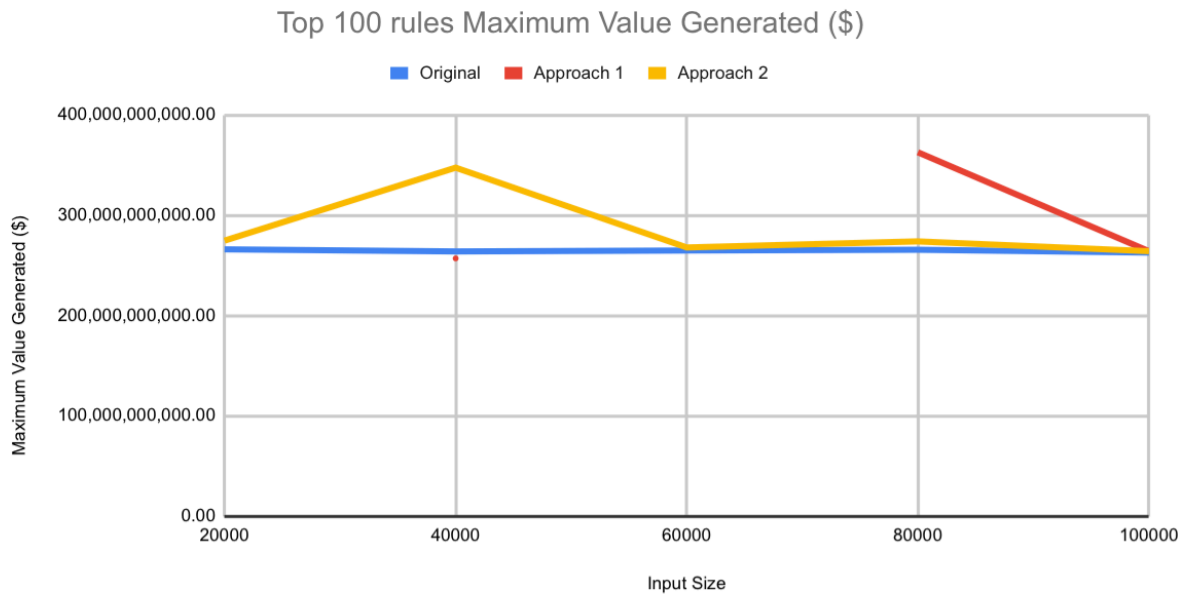


*Figure 11. Maximum values generated for top 100 rules*

Expectedly approach 2 had a much lower execution time than the baseline approach without k-means, which has its time complexity, increasing exponentially as the size of the itemset grew compared to approach 2 which had a much lower increase relative to the size of the itemset. Surprisingly, however, approach one had an extremely variable time complexity. For the 20,000 40,000 and 60,000 input sizes approach 2 had an extremely similar execution time to approach 1. When reaching 80,000, approach 1's execution time increased dramatically, surpassing that of the baseline. At 100,000 it decreased, dropping below the baseline but still much higher than approach 1. The likely explanation for this is fairly simple. The amount of data being clustered out by the clustering part of approach one is likely extremely variable. This is likely due to the Dunn index approach, which may find there to be as many as nine clusters with more clusters, likely leading to less data being in each cluster. Consequently, less data is in the bottom left cluster being filtered out, and thus more is being sent to Apriori which is very computationally expensive. This can even cause this approach to take longer than the baseline, since it also has to do calculations for the K means even if no data is clustered out. As for why this issue doesn't affect approach 1, it is likely because of the dimensions chosen, that the data is distributed in a way that will lead to lots of data always being in the bottom left most cluster either through the data being dense in that area or being distinctly in general leading to a fixed number of clusters.

Our hypothesis leading into this experiment was that the value generated by approach 2 would exceed the other methods. While approach 2 on average is marginally better performing, all algorithms perform approximately the same. The most surprising part, however, is that the results are consistent across all input sizes. This means that roughly the same value is generated from the transaction rules given only a 20,000 input size as is from a 100,000 input size. Explanation for this could be that even though the rules generated in the top 100 mostly changed, the new rules do not tend to actually have any more value as the input size increases. This could be because besides the top few rules, which are very valuable. There are a lot of rules with very similar values. Much more research needs to be done into

this phenomenon particularly with other itemsets and top X metrics to see if this holds consistently true. The reasoning for variation could be due to the occasional presence of three item rules, which inflate value since the value formula sums over all three items, instead of just two of them, which can lead to large increases in the overall value. The missing data for some of approach one is due to less than 100 rules being generated in those cases, this is another issue with approach 1 that is not seen by approach 2.

Overall, our novel proposed approach vastly outperforms the baseline in terms of execution while also marginally exceeding it in terms of value generated across all input sizes on our sample itemset. Compared to the previously proposed dimensions, but by Kanhere et al. Our approach is much more consistent and reliable while maintaining the same best results. Our approach yields promising returns as being a potential future method used in place of the current base line, to lower computational expense to a manageable level for associational mining on large scale itemsets, while not losing any of the quality of results. More research and experiments will need to be conducted on different itemsets before the true usefulness of this approach is deciphered.

## 8. Conclusion

In this paper, we present an approach to improve the efficiency and effectiveness of Market Basket Analysis for large datasets through the implementation of K-Means Clustering for Association Rule Mining. Our novel approach is using K-Means Clustering to process the raw data of an itemset into clusters based upon revenue and revenue of items purchased alongside. We then apply the Apriori Algorithm to filter out any low value itemsets clusters to generate the Association Rules needed to uncover connections between items in customers' transactions. Our approach has proven very effective through much better efficiency while also improving value generated by top rules compared to the baseline and previous methods across various input sizes.

### 8.1. Key Findings

Our findings from our testing suggests that implementing the K-Means clustering reduced the computational load for the Apriori algorithm association rule mining. By segmenting the transactional data into clusters into clusters and filtering out the less meaningful or significant clusters, a more manageable dataset has been created without compromising the integrity and usefulness of the final association rules generated by the implemented Apriori algorithm.

Since this algorithm is a non-deterministic problem, there is no one single outcome or solution for an input. This was proven by the K-Means algorithm generating different results based on different numbers of clusters. Furthermore, the result when executing the K-Means algorithm for one number of clusters varied as the clusters could either have high or low inertia depending on the dataset given and the number of centroid presented as it can have a random amount per iteration, meaning the generated association rules could vary as a result.

## 8.2. Contributions

This project contributes to data mining to provide a scalable solution particularly for large companies with large volumes of transactional data. Integrating clustering techniques with Market Basket Analysis has overcome the limitations foreseen by Association Rule Mining and similar algorithms such as costs and efficiency.

Two approaches were used when testing the implementation with K-Means and Apriori, focusing on 2 different criteria; frequency and price, and revenue generated. With several different tests and runs, it was concluded that the second approach that handles and focuses on revenue is more promising than the first approach that deals with the frequency and price of items. During observation, the second approach has a much more consistent reduced execution time by a drastic margin and attains slightly better results produced without optimizing K-Means clustering. This would conclude that even without K-Means clustering optimization, the approach still kept the results intact while also reducing time when analyzing the data which makes it a more promising solution.

## 8.3. Future Work

For the future, there can be several potential improvements that can be made to boost the effectiveness of our approach for Market Basket Analysis integration. Testing on varieties of datasets that includes larger transactional data to showcase and validate the effectiveness and performance of the approach across multiple domains and datasets. While K-Means clustering has proven to be an effective approach to cluster the data into useful and insightful groups of data, it will be good to explore and experiment forms of clustering algorithms and methodologies such as mean shift clustering and hierarchical clustering among others. The result and outcome could also be influenced by varying the parameters in the K-Means clustering as well as experimenting with different metrics for the possibility of a more refined outcome and association rules that are better suited for the dataset.

In the end, our proposed approach has proven to generate better value and much better efficiency compared to previous methods. More research should be done into our value function and its validity as well as ensuring the practical usefulness in different use cases and business contexts. Of course, being a new area of research, there are still many improvements that could be made to achieve even greater results.

## References

List of all cited references following a specific citation style (e.g., APA, IEEE).

[1] Market basket analysis definition.

<https://www.techtarget.com/searchcustomerexperience/definition/market-basket-analysis>

[2] Lloyd's algorithm. <https://datasciencelab.wordpress.com/tag/lloyds-algorithm/>

[3] Agrawal, R., Imieliński, T., & Swami, A. (1993). Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2), 207-216.

<https://dl.acm.org/doi/pdf/10.1145/170036.170072>

- [4] The Association Rule Mining problem is NP Hard.  
<https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=38eba3983bd71d03f6bbe35d3bd998c27bab7ce9>
- [5] Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. Proceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, 487-499. <https://www.vldb.org/conf/1994/P487.PDF>
- [6] MacQueen, J. B. (1967). Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, 1, 281-297.  
<https://ieeexplore.ieee.org/stampPDF/getPDF.jsp?tp=&arnumber=1056489>
- [7] Mahajan, M., Nimbhorkar, P., & Varadarajan, K. (2009). The planar k-means problem is NP-hard. Theoretical Computer Science, 442, 13-21.  
<https://www.imsc.res.in/~meena/papers/kmeans.pdf>
- [8] Inaba, M., Katoh, N., & Imai, H. (1994). Applications of weighted Voronoi diagrams and randomization to variance-based k-clustering: (extended abstract). Proceedings of the tenth annual symposium on Computational geometry, 332-339.  
<https://dl.acm.org/doi/10.1145/177424.178042>
- [9] Lloyd, S. P. (1982). Least squares quantization in PCM. IEEE Transactions on Information Theory, 28(2), 129-137. <https://www.jstor.org/stable/2346830>
- [10] Arthur, D., & Vassilvitskii, S. (2006). How slow is the k-means method? Proceedings of the twenty-second annual symposium on Computational geometry, 144-153.  
<http://www.stat.yale.edu/~hz68/Lloyd.pdf>
- [11] Arthur, D., & Vassilvitskii, S. (2021). k-means++: The advantages of careful seeding. ACM Transactions on Knowledge Discovery from Data, 15(2), 1-28.  
<https://ieeexplore.ieee.org/document/9347577>
- [12] Number of credit card transactions.  
<https://capitaloneshopping.com/research/number-of-credit-card-transactions/>
- [13] What is processor speed?  
<https://www.lenovo.com/ca/en/glossary/what-is-processor-speed>
- [14] Eclipse.org consortium. <https://www.eclipse.org/org/>
- [15] Simultaneous Release.  
[https://github.com/eclipse-simrel/.github/wiki/Simultaneous\\_Release.md](https://github.com/eclipse-simrel/.github/wiki/Simultaneous_Release.md)
- [16] Eclipse Foundation Corporate Presentation - v2023.  
<https://www.eclipse.org/org/documents/>
- [17] Kaggle: Your Machine Learning and Data Science Community. <https://www.kaggle.com/>
- [18] Market Basket Analysis. <https://www.kaggle.com/datasets>

# Appendices

## Appendix A

The code below demonstrates how the K-Means algorithm is run for each number of clusters between 2 and 10 (inclusive).

```
for (int numofClusters=2; numofClusters<10;numofClusters++)
{
    double[] interClusterDistance= new double [100];
    algo (c,numofClusters, values, numofPoints, nearestCentroidopt,
interClusterDistance,r,false,dimensions,inertia);
    int mindis=0;
    for (int y=1;y<100;y++)
    {
        if(interClusterDistance[y]<interClusterDistance[mindis])
            mindis=y;
    }
    System.out.println("Number of clusters: "+numofClusters + "
Inertia: "+inertia[numofClusters]);
}
```

The code calls and executes the K-Means algorithm with the current number of clusters and calculates the inertia. This is done until convergence has reached and the maximum number of iterations have been reached

The code below shows how the Dunn index is calculated for the results and how the best one is selected.

```
int elbow=3;
for (int i=4;i<15;i++)
{
    if((inertia[i]-inertia[i-1])/(inertia[i+1]-inertia[i])>(inertia[elbow]-inertia[elbow-1])/(inertia[
elbow+1]-inertia[elbow]))
        elbow=i;
}
System.out.println("elbow: "+elbow);

double maxDunn[]=new double[3];

for (int j=elbow-1;j<elbow+2;j++)
{
```

```

        for(int i=0;i<r;i++)
        {
            double
dunn=findDunnIndex(c[j][i],values,dimensions,numofPoints,j);
            //System.out.println(j+ "::"+dunn);
            if(dunn>maxDunn[j-elbow+1])
            {
                maxDunn[j-elbow+1]=dunn;
                for(int k=0;k<numofPoints;k++)
                {
                    nearestCentroidopt[j][k]=c[j][i][k];
                }
            }
        }
    }

    int bestnumofClusters=elbow-1;
    System.out.println(elbow-1 + " " +maxDunn[0]);
    for(int i=elbow;i<elbow+2;i++)
    {
        System.out.println(i + " " +maxDunn[i-elbow+1]);
        if(maxDunn[i-elbow+1]>maxDunn[bestnumofClusters-elbow+1])
            bestnumofClusters=i;
    }
    elbow=bestnumofClusters;

```

The elbow is set to 3, which states this is the elbow on the graph, and the result close to or next to this elbow is chosen at the best. We check the inertia of each result and find the dunn index respectively. The program tracks what result currently has the best dunn index value and the nearest centroid is updated as well. When the best dunn index has been found, then that will also be our best number of clusters needed as run by the K-Means, so the elbow or our best choice is the number of clusters obtained from the best result.

The code below is an implementation of the Dunn index.

```

public static double findDunnIndex(int nearestCentroid[],double values[][],int
dimensions,int numofPoints,int numofClusters)
{
    double maxInterClusterDistance=0;
    double minIntraClusterDistance=0;
    double dunnIndex=0;
    double distance[]=new double [numofClusters];
    double distanceCount[]=new double [numofClusters];

```

```

double raDistance[][]=new double [numofClusters][numofClusters];
double raDistanceCount[][]=new double [numofClusters][numofClusters];
for (int i=0;i<numofPoints-1;i++)
{
    for (int j=i+1;j<numofPoints;j++)
    {
        double apart=0;
        for (int k=0;k<dimensions;k++)
            apart+=(values[i][k]-values[j][k])*(values[i][k]-values[j][k]);
        if(nearestCentroid[i]==nearestCentroid[j])
        {
            distance[nearestCentroid[i]]+=Math.sqrt(apart);
            distanceCount[nearestCentroid[i]]++;
        }
        else
        {
            raDistance[nearestCentroid[i]][nearestCentroid[j]]+=Math.sqrt(apart);
            raDistanceCount[nearestCentroid[i]][nearestCentroid[j]]++;
        }
    }
}

minIntraClusterDistance=(distance[0]/distanceCount[0]);
maxInterClusterDistance=(raDistance[0][1]/raDistanceCount[0][1]);
for (int i=1;i<numofClusters;i++)
{
    if (distance[i]/distanceCount[i]>minIntraClusterDistance)
        minIntraClusterDistance=(distance[i]/distanceCount[i]);
    for (int j=i+1;j<numofClusters;j++)
    {
        if(raDistance[i][j]/raDistanceCount[i][j]<maxInterClusterDistance)
            maxInterClusterDistance=(raDistance[i][j]/raDistanceCount[i][j]);
    }
}

//System.out.println(numofClusters+": "+maxIntraClusterDistance+":
"+minInterClusterDistance);

dunnIndex =maxInterClusterDistance/minIntraClusterDistance;
return dunnIndex;

```

}

This is how the dunn index is found to gain the best result. The Dunn index is a metric to evaluate the quality of a cluster. It is defined as the smallest distance between observations in different clusters and largest intra cluster distance. Formally it is calculated as the inter cluster distance divided by the intracluster distance.

The method iterates through each pair of points and calculates the euclidean distance. If the points belong to the same cluster, the points are added to the distance array. If the points do not belong to the same cluster, they are added to the raDistance array. After iteration, the average distance between the points in the same cluster and the average distance between points in different clusters are calculated. The minimum intra-cluster and maximum inter-cluster distances are then found.

## Appendix B

Below is the dataset used for this project and research, with the starting data to the end data records.

|    | Date     | Customer | Transaction | SKU_Categ | SKU   | Quantity | Sales_Amount |  |  |
|----|----------|----------|-------------|-----------|-------|----------|--------------|--|--|
| 1  | 2/1/2016 | 2547     | 1           | X52       | OEM7L | 1        | 3.13         |  |  |
| 2  | 2/1/2016 | 822      | 2           | 2ML       | 68BRQ | 1        | 5.46         |  |  |
| 3  | 2/1/2016 | 3686     | 3           | 0H2       | CZUZX | 1        | 6.35         |  |  |
| 4  | 2/1/2016 | 3719     | 4           | 0H2       | 549KK | 1        | 5.59         |  |  |
| 5  | 2/1/2016 | 9200     | 5           | 0H2       | K8EHH | 1        | 6.88         |  |  |
| 6  | 2/1/2016 | 5010     | 6           | JPI       | GVBRQ | 1        | 10.77        |  |  |
| 7  | 2/1/2016 | 1666     | 7           | XG4       | AHAE7 | 1        | 3.65         |  |  |
| 8  | 2/1/2016 | 1666     | 7           | FEW       | AHZNS | 1        | 8.21         |  |  |
| 9  | 2/1/2016 | 1253     | 8           | 0H2       | 9STQJ | 1        | 8.25         |  |  |
| 10 | 2/1/2016 | 5541     | 9           | N5F       | 7IE9S | 1        | 8.18         |  |  |
| 11 | 2/1/2016 | 5541     | 9           | H8O       | M15RG | 1        | 6.35         |  |  |
| 12 | 2/1/2016 | 7548     | 10          | N8U       | UNJKW | 1        | 2.11         |  |  |
| 13 | 2/1/2016 | 7548     | 10          | JR5       | KWDJZ | 1        | 5.38         |  |  |
| 14 | 2/1/2016 | 6865     | 11          | I4Y       | 8ETY5 | 1        | 8.04         |  |  |
| 15 | 2/1/2016 | 6044     | 12          | N8U       | EMJ1S | 1        | 3.62         |  |  |
| 16 | 2/1/2016 | 6044     | 12          | X52       | CRK31 | 1        | 3.96         |  |  |
| 17 | 2/1/2016 | 592      | 13          | YMJ       | YJTMP | 1        | 11.65        |  |  |
| 18 | 2/1/2016 | 592      | 13          | P42       | B2IW9 | 1        | 9            |  |  |
| 19 | 2/1/2016 | 6450     | 14          | N8U       | 4A2M0 | 1        | 7.02         |  |  |
| 20 | 2/1/2016 | 3874     | 15          | FEW       | F79YP | 1        | 16.91        |  |  |
| 21 | 2/1/2016 | 3874     | 15          | SFC       | 26XOE | 2        | 4.75         |  |  |
| 22 | 2/1/2016 | 5599     | 16          | 7C6       | OZEQ8 | 1        | 7.85         |  |  |
| 23 | 2/1/2016 | 5599     | 16          | OQA       | DWJC4 | 1        | 12.87        |  |  |
| 24 | 2/1/2016 | 9223     | 17          | TW8       | E63V3 | 1        | 1.62         |  |  |
| 25 | 2/1/2016 | 9223     | 17          | Q4N       | EGB8E | 1        | 10.5         |  |  |
| 26 | 2/1/2016 | 4921     | 18          | H15       | NEYQ4 | 1        | 3.75         |  |  |

|        |          |       |       |     |       |   |       |  |  |  |
|--------|----------|-------|-------|-----|-------|---|-------|--|--|--|
| 131680 | 1/8/2016 | 6812  | 37238 | R6E | F90L2 | 1 | 2.37  |  |  |  |
| 131681 | 1/8/2016 | 6812  | 37238 | R6E | EIX2K | 1 | 3.69  |  |  |  |
| 131682 | 4/7/2016 | 11042 | 32891 | 8HU | YRULF | 1 | 6.81  |  |  |  |
| 131683 | 4/7/2016 | 11042 | 32891 | LPF | NOWXS | 2 | 64.23 |  |  |  |
| 131684 | 4/7/2016 | 10126 | 32892 | QV7 | CG8C3 | 1 | 4.11  |  |  |  |
| 131685 | 4/7/2016 | 17588 | 32893 | QV7 | RBZMC | 1 | 2.4   |  |  |  |
| 131686 | 4/7/2016 | 19901 | 32894 | C3R | N7AW3 | 1 | 27.93 |  |  |  |
| 131687 | 4/7/2016 | 19901 | 32894 | C3R | 9E9GV | 1 | 23.13 |  |  |  |
| 131688 | 1/8/2016 | 17567 | 37239 | NTA | C6TXL | 1 | 4.5   |  |  |  |
| 131689 | 1/8/2016 | 17567 | 37239 | IEV | 3PWIY | 1 | 6.7   |  |  |  |
| 131690 | 1/8/2016 | 17567 | 37239 | M7X | 2A2WH | 1 | 11.79 |  |  |  |
| 131691 | 4/7/2016 | 16563 | 32895 | XG4 | IQNBU | 1 | 6.62  |  |  |  |
| 131692 | 4/7/2016 | 16563 | 32895 | Q4N | 9TF6B | 1 | 4.06  |  |  |  |
| 131693 | 4/7/2016 | 10501 | 32896 | ILF | 5TPXK | 1 | 2.15  |  |  |  |
| 131694 | 4/7/2016 | 10764 | 32897 | X52 | F9JVE | 1 | 3.74  |  |  |  |
| 131695 | 4/7/2016 | 10764 | 32897 | R6E | DUV2Y | 1 | 8     |  |  |  |
| 131696 | 4/7/2016 | 11733 | 32898 | N8U | X5TFY | 1 | 4.57  |  |  |  |
| 131697 | 4/7/2016 | 10468 | 32899 | 1VL | USW0M | 1 | 5.87  |  |  |  |
| 131698 | 4/7/2016 | 10468 | 32899 | RML | EDZ1Y | 1 | 6.07  |  |  |  |
| 131699 | 4/7/2016 | 10468 | 32899 | LSD | 4AZHE | 1 | 7.73  |  |  |  |
| 131700 | 4/7/2016 | 20203 | 32900 | J4R | LI0IX | 1 | 6.25  |  |  |  |
| 131701 | 4/7/2016 | 20203 | 32900 | U5F | F7FQ5 | 3 | 7.27  |  |  |  |
| 131702 | 4/7/2016 | 20203 | 32900 | IEV | FO112 | 3 | 6.46  |  |  |  |
| 131703 | 4/7/2016 | 20203 | 32900 | N8U | I36F2 | 1 | 4.5   |  |  |  |
| 131704 | 4/7/2016 | 20203 | 32900 | U5F | 4X8P4 | 1 | 5.19  |  |  |  |
| 131705 | 4/7/2016 | 20203 | 32900 | OH2 | ZVTO4 | 1 | 4.57  |  |  |  |
| 131706 | 4/7/2016 | 20203 | 32900 | Q4N | QM9BP | 1 | 13.68 |  |  |  |

This dataset has approximately 131,706 transactions, with appropriate headers and columns used for the Market Basket Analysis data mining integration.