MASTER'S THESIS 2020

Personalized Product Recommendations

Alexander Pålsson, Enis Hajzeri

ISSN 1650-2884
LU-CS-EX 2020-10

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2020-10

Personalized Product Recommendations

Alexander Pålsson, Enis Hajzeri

Personalized Product Recommendations

(Personalized Product Recommendation Engines For Retailing)

Alexander Pålsson
dat14apa@student.lu.se

Enis Hajzeri
dat14eha@student.lu.se

March 19, 2020

Master's thesis work carried out at Qlik .

Supervisors: Pierre Nugues, pierre.nugues@cs.lth.se
José Díaz López, jose.diazlopez@qlik.com

Examiner: Jacek Malec, jacek.malec@cs.lth.se

Abstract

Recommendation engines are today commonly used by companies that provide a large collection of products to a large number of customers. The goal is to help the customers navigate through the store by proposing products that fit their needs.

This Master's thesis presents two personalized recommendation engines that can be used in different scenarios in *Market Basket Analysis*. The two systems explore different priorities between accuracy, cost and efficiency and are evaluated against *Frequent Pattern Growth* (FPG), the industry standard *frequent item mining* algorithm. The evaluation methods used are precision, recall and F1-score.

ARM-KC is implemented by combining FPG with the clustering algorithm *K-mode*. The association rules provided by the FPG algorithm are used as customer characteristics by the K-mode algorithm. This system results into better recommendations for some customers but takes longer time to train than a solution utilizing only FPG. ARM-KC might therefore be used for specializing in the needs of some customers provided only their purchase history as input.

The other system is an implementation of *association rule mining* by utilizing the *Qlik Associative Engine* (QIX) API. Unlike FPG, this solution requires no training and provides very fast recommendations for a few customers at a time. The system might therefore be used in real-time applications such as frequently altering products and customers.

Keywords: market basket analysis, association rule mining, frequent item mining, clustering, recommendation engine

Acknowledgements

We would like to express our gratitude to our supervisors Pierre Nugues at LTH and José Díaz López at Qlik for believing in the project and providing invaluable feedback and guidance during the entire process.

We would like to thank Jonas Hörström for logistical and practical assistance. Thank you Jonas and our colleagues at Qlik for being very welcoming.

We would also like to thank Yolanda Perdomo at Ikano Bank for invaluable idea generation regarding problem formulation and development.

Contents

Chapter 1
Introduction

Large companies, such as IKEA, provide a large collection of products to a large number of customers. It can be very difficult for the customers to navigate through the assortment. Furthermore, companies often make money from unplanned purchases, meaning that the customer visits the store to purchase certain products and realises that s/he needs other products as well. This is where recommendation engines come in.

There are many ways a recommendation engine could be implemented but the main goal is always to propose products or service. For a company selling products to customers, the goal of a recommendation engine is to predict what products the customers are potentially interested in.

1.1 Problem statement

Since recommendation engines could be applied on many areas, a difficult task is defining your exact needs. The next step is to decide on the most appropriate tools. Furthermore, a prioritisation must be made between different objectives, such as efficiency, cost and accuracy.

A recommendation engine can be implemented with simple approaches such as recommending the most frequently purchased products. The time it takes to compute a recommendation with this approach is insignificant. A more advanced approach is to have individual predictions. To implement it, one needs to determine categories of products given the customer characteristics and history of purchases. An approach to solve this kind of problem is to use neural networks. The training phase for such an approach can be several days or even weeks, which might be problematic if the data is frequently changing. Although the accuracy might increase, the ability to add new products or customers might disappear due to the long training phase needed.

One area where recommendation engines are often used is market basket analysis. The industry working with market basket analysis must utilize appropriate algorithms when build-

ing recommendation engines. Since there is no correct answer on where the balance between cost, efficiency and accuracy lies, approaches used today should be modified in order to explore potentially better approaches.

1.2 Objective

To solve this problem we have split our work into four objectives:

1. Implement a recommendation engine for market basket analysis with increased accuracy compared to implementations with industry standard algorithms, while maintaining the low cost.

2. Implement customer segmentation given data without any customer characteristics such as *parent*, *house owner* and *car owner* but only purchase history.

3. Implement a recommendation engine which can be used in real-time applications with constantly differing products, customers or features.

4. Implement a recommendation engine with the possibility of considering all available products instead of the most frequent ones.

1.3 Related research

The design of recommendation systems to compute product/customer recommendations based on historical purchase data is a very explored field; thus there are numerous research articles on the matter.

Agrawal et al. (1994) propose an algorithm called *Apriori* in the context of mining association rules. The Apriori algorithm has since worked as a popular benchmark for association rule mining and basket analysis algorithms.

Hong-Wei et al. (2004) went one step further to make recommendations by first creating their own virtual mall to simulate the behaviours of customers. Then with a combination of machine learning techniques, such as *fuzzy set* and *case-base reasoning*, they created customer tailored recommendations. They also brought up an important point, namely that with e-commerce there is just too many different products for customer to handle. Therefore, customer specific recommendations get more and more important.

Abbas et al. (2013) provide a real life example of how a recommendation engine could be implemented in a retail store. Using a store selling sport equipment, they implement the market basket algorithm *Apriori*. Based on historical data they mined product relationships. Based on the product relationships, they created recommendations on how the sport store should be organized to increase sales.

Minh and Wu (2019) explored topology based user recommendations. With a movie dataset they utilized path based semantic similarities between customers based on their movie ratings in order to get personalized movie recommendations.

A Master's thesis done by Glas (2015) compared customer recommendation machine learning algorithms. The thesis included the algorithms: *K-Means Clustering*, *Decision Tree*,

C4.5, *k-Nearest Neighbors* and *Apriori*. These algorithms were discussed based on their use cases, strengths and weaknesses.

Multiple studies have been done on combining association rules with other algorithms. In many of those studies, there has been a focus on classifying frequent patterns, in order to get a deeper understanding of datasets.

Cao et al. (2011) handled the topic of integrated frequent pattern mining with classification to generate frequent pattern-based classifiers. The paper does not propose any specific algorithms. Instead, they have a broader approach where they try to summarize frameworks, paradigms, and basic process. They finish the paper by doing a real-life study using the techniques they discussed to prevent government debt by identify alarming patterns.

Yanchang et al. (2008) were more specific and proposed techniques to combine association rules with different techniques. The study evaluates three different ideas, namely combining different associations rules, combining rule pairs and lastly clustering association rules.

Although there have been a lot of research about both recommendation engines and combining frequent pattern mining with clustering algorithms, the field of customer segmentation based on their buying patterns is unexplored. Especially when using categorical data clustering and more specifically the K-mode algorithm.

1.4 Limitations

One limitation for us when we wanted to make interpretations of the clustering was that we had to work with an anonymized data set. Most variables, including product and customer names, were translated into Id-numbers or unrecognizable names. For example products in the decoration section had names like Decoration - product 1, Decoration - product 2 and so on.

The anonymization led to several problems such as the loss of the ability to provide the customers with meaningful labels such as *Parent* and *House Owner*. Another problem is the reduced ability to decide which categories of products are the most popular ones. There is the possibility to utilize the anonymous product numbers for this but it would not be as meaningful as knowing what the products actually are.

All implementation and computation were done on laptops with 32GB RAM memory and 6-core processors. This led to some time needed to train and evaluate the algorithms. Due to the time limit of the project, the time spent waiting affected what kind of algorithms that we tried and how many different solutions and evaluations we produced.

1.5 Contributions

The experiments presented in this thesis provide creative implementations of recommendation engines that explore other levels of balance between accuracy, cost and efficiency than the industry standards.

The experiments also provide a way of clustering customers with the only prior knowledge about them being their purchase history.

1.6 Report outline

The report consists of five chapters, namely Approach, Development, Results, Discussion and Conclusion.

- Approach: This chapter includes the theory behind vital parts mentioned on the other chapters, such as the different algorithms we used, different tools used to process data, and evaluation metrics.
- Development: This chapter presents the different parts of the system and how they were implemented. Libraries used during the development are presented and explained.
- Results: This chapter consists of results and evaluations from running the different parts of the system in the form of graphs and tables. The chapter also includes an interpretation of the graphs and tables.
- Discussion: This chapter discusses the different parts of the system and the results of the evaluation. The outcome is also compared to what it should have been and what might have caused the difference.
- Conclusion: This chapter highlights the most important findings of the thesis.

Chapter 2
Approach

2.1 Data Format

The data set consists of receipts of transactions from the IKEA furniture chain. The receipt data was divided into four different tables:

- Customer - containing 103,995 customers,
- Purchase - containing 1,724,153 purchases,
- Product - containing 45,141 products,
- Purchase_details - containing 13,072,392 details about every item bought.

Figure 2.1 shows a more complete description of the tables.

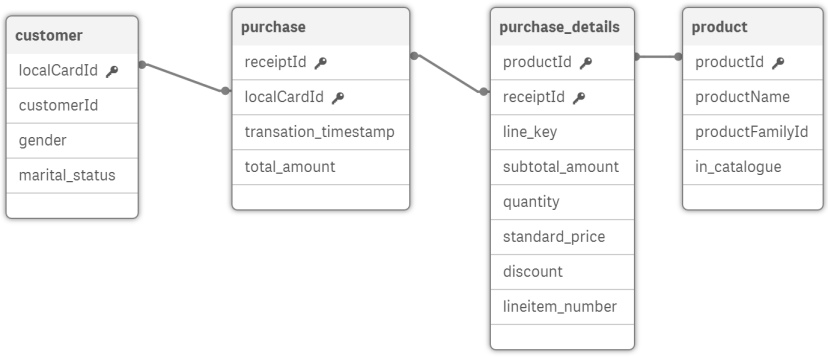

Figure 2.1: Structure of the receipt data

Because of personal information protection laws, the data was anonymized, which made it impossible to distinguish products from each other and likewise for customers. As can be seen in Figure 2.2, it is impossible to see what products the table is referring to.

productId	productName	productFamilyId	in_catalogue
39999	Home organisation. - product 1	1	NULL
41802	Home organisation. - product 2	1	NULL
22514	store Food - product 1	1	NULL
40295	Home textiles - product 1	2	NULL
22476	Home textiles - product 2	2	NULL
39376	Decoration - product 1	3	NULL
40551	Decoration - product 2	4	NULL

Figure 2.2: First items in the anonymized products table

2.2 Clustering

Clustering is a method of grouping objects in a data set depending on similarities.

Clustering corresponds to partitioning data points or data objects into groups called *clusters*. Clusters consist of multiple objects where all the objects have related characteristics. According to Xu and Wunsch (2009), there is no general definition of a cluster, however, Everitt (1980) summarizes clustering with his general description:

> A cluster is a set of entities which are alike, and entities from different clusters are not alike.

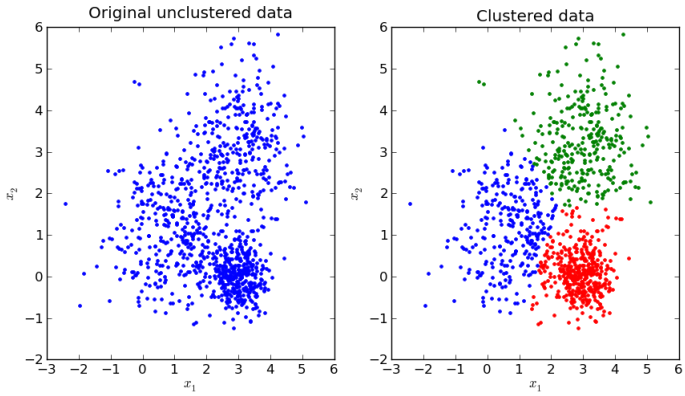

Figure 2.3: Data points with and without clustering

To solve the clustering problem, many algorithms have been proposed and the three most common are:

1. Density-based clustering.
2. Hierarchical clustering.
3. Centroid-based clustering.

2.2.1 Hierarchical Clustering

Hierarchical clustering (Johnson, 1967) incrementally builds a hierarchy within the data set. The two points with the lowest distance between them will form a pair. This pair is then

merged into a new point. The algorithm proceeds to merge other pairs of data points, reducing by one the number of points at each iteration. The resulting object is called a *dendrogram*.

The data objects are built into a dendrogram depending on their distance to other data points. Furthermore, this pair will merge with the object or sub dendrogram with the smallest distance, forming a pair of clusters.

The first and standard hierarchical clustering algorithm has a $O(n^3)$ complexity, which makes it slow for large datasets. However, there are more recent algorithms which address the time complexity problem; for example SLINK (Sibson, 1973). SLINK lowers the run time of the algorithm to $O(n^2)$.

2.2.2 Centroid-based clustering

The centroid-based clustering algorithms calculate centroids, which are the centers of the clusters, and assign every data object to the closest centroid. The centroids then form clusters. There are many variations of centroid-based clustering but they all share the basic goal of minimizing the distances between the centroids and the data points. In order to find the closest distance, three measuring methods are frequently used, namely *Euclidean, Minkowski* and *Manhattan distance* (Santosh, 2014). Hamming distance is another measuring method that is used less frequently, but will be important in our application.

The **Euclidean distance** is the straight-line distance between two data points in the Euclidean space. The general computation of the Euclidean distance between two vectors $\mathbf{q}, \mathbf{p}$ is calculated by using the Pythagorean formula, which looks as following:

$$d(\mathbf{q}, \mathbf{p}) = \sqrt{(\mathbf{q} - \mathbf{p})^2} \tag{2.1}$$

The properties of Euclidean distance gives the algorithm a disadvantage; it becomes very sensitive to outliers. Cleaning the data by removing outliers reduces the impact. However, excessive cleaning might create distortion within the data.

The **Hamming distance** measures the binary similarity between two strings (Hamming, 1950). Hamming distance excels when clustering categorized data. In this case, Euclidean distance is useless since the data points have no meaningful distance between them. An example is data consisting of letters instead of numbers.

Given three encoded binary numbers 101, 1000 and 1010; the $d_{Hamming}(0101, 1000) = 3$, $d_{Hamming}(0101, 1020) = 4$ and $d_{Hamming}(1000, 1010) = 1$. Given $a = 10$ and $b = 01$ calculating the similarity between the strings a, b, ab the Hamming distances are given by $d_{Hamming}(a, b) = d_{Hamming}(10, 01) = 2$ and $d_{Hamming}(ab, a) = d_{Hamming}(11, 10) = 1$.

2.2.3 NP-Hard

The metric distance based centroid problems are computationally hard and known to be be NP-Hard (Mahajan et al., 2009). Due to NP-hardness, the optimal centroids that minimize the distance between the centroids and data sets cannot be computed. Therefore, clustering algorithms are constructed to find an approximate solution.

2.2.4 K-means

One of the most common algorithms to find the local optimum is Lloyds algorithm, also known as K-means (Lloyd, 1982). Local optimum is the optimal solution given its closest neighbours. In order to find the local optimum, clustering algorithms consequently iterate through different centroids. The K-means clustering algorithm looks as follows:

1. Start with initial guesses for cluster centers (centroids)
2. For each data point, find closest cluster center (partitioning step)
3. Replace each centroid by the center of gravity of data points in its partition
4. Iterate 1+2 until convergence

2.2.5 One-Hot Encoding Variables

In 1957 Suits (1957) introduced one-hot encoding variables. One-hot encoding variables are a numerical representations of categorical data. Originally one-hot encoding variables were used in linear regression but can be effectively used in clustering. One-hot encoding variables can take two values 0 or 1. For K items the number of one-hot encoding variables needed is $k-1$. The first step in creating one-hot encoding variables is to count the number of unique categories or unique items within the category. Then assign each of the items or categories with a unique nominal value. With a category C and the items: i_1, i_2,i_3 the items assigned nominal values can be seen in Figure 2.1. To represent these items as one-hot encoding variables, each item is represented as a column with either 0 or 1 if present. This makes detecting unknown items simple, since all unknown items would be represented by an all zero vector. One-hot encoding for the three items can be seen in Figure 2.2.

Item	Nominal value
I_1	1
I_2	2
I_3	3

Table 2.1: Nominal values of category items

	I_1	I_2	I_3
I_1	1	0	0
I_2	0	1	0
I_3	0	0	1
$I_{Unknown}$	0	0	0

Table 2.2

2.2.6 Clustering Categorical Data

In K-means, clusters are formed by minimizing the Euclidean distance between data points, which requires all objects to have numerical values. Hence, clustering categorical data is infeasible with a regular K-means algorithm. Ralambondrainy (1995) suggests transforming categories into elements in vectors using one-hot encoding variables as described in Section 2.2.5. Given the Table 2.3, each customer will be represented by a vector with 3 elements where the elements are the different furniture present in the set. This results in the one-hot encoding matrix shown in Table 2.4.

Customer	Furniture
C_1	Table, Chair
C_2	Chair
C_3	Lamp, Table

Table 2.3: Assignment of rule codes

	Table	Chair	Lamp
C_1	1	1	0
C_2	0	1	0
C_3	1	0	1

Table 2.4: Assignment of rule codes

The vectors require as many dimensions as there are categories in the data set. This method used on high-dimensional categorical data leads to the problem of having very large matrices to run K means on.

An example of a potential implementation could be on a catalogue of IKEA products. IKEA has more around 45,000 articles, meaning that each cluster centroid would be represented by a 45,000 dimensional vector, which would give a various number of problem;

1. Firstly, clustering a 50,000 dimensional vector would require powerful computers.

2. Secondly, wide and sparse matrices could potentially encounter a problem called *The Curse of Dimensionality* (Bellman et al., 1957), which is a gathering of phenomena. The Curse of Dimensionality occurs when the number of the dimensions gets higher. The consequence of this is that the number of data needed to support the result needs to grow. However, curse of dimensionality is a discussed subject and Domingos (2012) presents something called *the blessing of non-uniformity* as a counterpoint to the curse of dimensionality. The claim is based on that data is not uniformly dispersed within the feature space, therefore it is possible to gain traction by identifying the ways in which the data is organized.

2.2.7 K-mode

Huang (2009) presents an algorithm, called K-mode, which is a modified version of K-means to handle categorical data. While K-means tries to find similarities between objects by cal-

culating and comparing the Euclidean distance, K-mode tries to find dissimilarities by computing the Hamming distance. The definition of the K-mode algorithm can be seen in Eqs 2.2 and 2.3.

$$d(X, Y) = \sum_{j=1}^{m} \delta(x_j, y_j) \tag{2.2}$$

where

$$\delta(x_j, y_j) = \begin{cases} 0, & x_j = y_j \\ 1, & x_j \neq y_j \end{cases} \tag{2.3}$$

2.3 Affinity analysis

Affinity analysis, often called *market basket analysis* in the retail industry, is a data mining technique for finding patterns between items or groups within a data set (Gutierrez, 2006). Market basket analysis is used to map and create an understanding of customer behaviour. Understanding customer behaviours is important in retail industries since advertisement, product placement and product development are often based upon such information.

2.3.1 Association rule mining

Association rule mining or association rule learning is an affinity analysis machine learning algorithm that finds patterns in data sets. Agrawal et al. (1993) introduced the problem of finding associations between item sets in a big basket of data. Given a large set of items, the purpose of the algorithm is to find relations between them by finding association rules.

Agrawal et al. (1993) formally defined the problem with : $I = \{i_1, i_2, \ldots, i_n\}$ where I is a set of items $i_1, i_2, i_3, ..., i_n$ and $T = \{t_1, t_2, \ldots, t_n\}$ where T is a set of transactions $t_1, t_2, t_3, ..., t_n$

All the transactions contain a subset of I and all items in I occurs at least once in T. A rule is the implication $X \Rightarrow Y$, where X and Y are subsets of items from I. In a specific rule, the same items can not be present in both X and Y. X is called a antecedent and Y is called a consequent.

To illustrate this, supermarket receipts could be used and an item set could be Bread, Butter, Jam. The rules would then be as follows; $\{Bread\} \Rightarrow \{Butter\}$, $\{Bread, Jam\} \Rightarrow \{Butter\}$ ... $\{Bread\} \Rightarrow \{Butter, Jam\}$. $\{Bread\} \Rightarrow \{Butter, Jam\}$ implies that if a customer bought bread, s/he would also buy butter and jam.

To comprehend the significance of the different association rules, different measurements can be used.

Support is the frequency of an item set in the data set. The support is given by:

$$supp(X) = \frac{|t \in T; X \in t|}{|T|} \tag{2.4}$$

For the transactions $\{Bread, Butter\}, \{Butter, Jam\}$, $supp(Bread)$ is 1/2 = 0.5 or 50% since it appears in 1 out of 2 transactions. Support is variously presented with percent or with the number of times it occurs.

A low support means that the item set doesn't occur frequently in the data set. Therefore, it also implies that rules shouldn't consist of that item set since it might be an outlier rather than a rule. Therefore in all Frequent Itemset Mining algorithms, support is used to filter what set should be examined. Setting the support limit to 10% means that an item set must occur in 10% of the transactions in order to be taken into account.

Confidence is defined as follows:

$$conf(X \Rightarrow Y) = \frac{supp(X \cup Y)}{supp(X)} \tag{2.5}$$

Confidence is the conditional probability of occurrence of a consequent given the antecedent, meaning that it is the likeliness of a rule to hold true. For example if a customer buys bread, how likely is it that s/he also buys butter. Using the previous example $\{Bread, Butter\}$, $\{Butter, Jam\}$, $conf(Bread \Rightarrow Butter) = \frac{1}{1} = 100\%$ since every time bread is bought butter is also bought.

2.3.2 Frequent Pattern Mining

According to Han et al. (2007), to generate association rules, frequent itemsets can be utilized. There are three frequent pattern mining algorithms that provide frequent itemsets, namely Join-Based, Tree-Based and Recursive Suffix-Based.

Recursive Suffix-Based Growth

According to Han et al. (2007), Join-Based and Tree-Based algorithms use the prefix, which is a set of characters at the beginning of a word, to build a tree of frequent items. Recursive suffix-based growth on the other hand, extends the suffix, which is a set of characters at the end of the frequent patterns.

The conditional transaction database, a database containing only frequent extensions of the current suffix, is sent as an input in every iteration of the recursion. In every recursion, another frequent item is added in the projected database. In the suffix based approach, the frequent items are ordered in descending order.

Frequent Pattern Growth

One algorithm using Recursive Suffix-Based Growth is Frequent Pattern-Growth. FP-growth utilizes a FP-tree. The FP-tree uses compressed suffixes, making the approach more efficient.

According to Han et al. (2000), the definition of the FP-tree looks as follows:

1. A FP-tree consists of one root labeled as "*null*", a set of item prefix subtrees as the children of the root, and a frequent-item header table.

2. Each node in the item prefix subtree consists of three fields: item name, count, and nodelink, where item name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none.

3. Each entry in the frequent-item header table consists of two fields, (1) item name and (2) head of node-link, which points to the first node in the FP-tree carrying the item-name.

2.4 Qlik's Associative Difference

Qlik Associative Difference is a patented system to handle and analyse big data sets (Qlik, 2020b). The system is the foundation of Qlik's products and therefore all details of how the system work are not publicly available. There are two major reasons that Qlik is faster then a lot of SQL based tools (Qlik, 2020a).

1. The first is that Qlik associative engine (QIX) which is the calculation motor is not built with SQL. Therefore when working across different tables with QIX, no expensive outer joins is required.

2. The second thing that makes it faster is that the model is loaded into the RAM memory and therefore is easily accessible. This makes selection faster and makes it possible for frequently used data to be stored in the cache. This means calculations in datasets that are centred around a few but often used items can be done very fast.

2.5 Customer Segmentation

Customer segmentation is a branch within market segmentation. Market segmentation have been practiced long before it became a theoretical subject(Fullerton, 2016). There are evidence of market segmentation practiced dating to the bronze age (Alberti, 2016). Since beginning of 19th century it's been researched and widely practiced in the reselling industry(Fullerton, 2016). Market segmentation is dividing the market into groups of customer, segments. The aim is to create segments were the properties of the segments have similar characteristics. This makes it possible to make a conclusion about unique customers based on the properties of the segment.

A common strategy within customer segmentation is the *S-T-P approach* (Segmentation ⇒ targeting ⇒ positioning). First the market is divided into segments, then a number of segments are targeted with a position to attract the selected segment.

In the retail business, this is used to differentiate business/advertisement strategies based on customer segment. For a companies like IKEA customer segmentation is important and two customer segments could potentially be parents and retired people. These customer groups have widely different needs and therefore customer segmentation is important to target these customers with the products they are more likely to buy.

A big part of the customer segmentation is to identify the most profitable segment for the company.

2.6 Evaluation measures

According to Goutte and Gaussier (2005), in order to get a complete view of a system's performance, one might compute a precision- recall curve and their harmonic mean, F1-score

In order to define precision, recall and F1-score, four variables need to be explained:

- TP - True Positive stands for an object which is recommended by the system and is relevant.
- TN - True Negative stands for an object that is not recommended by the system and is not relevant.
- FP - False positive stands for an object that is recommended by the system but is not relevant.
- FN - False negative stands for an object that is not recommended by the system but is relevant.

2.6.1 Precision

Precision is defined as follows:

$$p = \frac{TP}{TP + FP} \tag{2.6}$$

An interpretation of this definition is that precision is the probability that an object that the system returns is relevant.

2.6.2 Recall

Recall is defined as follows:

$$r = \frac{TP}{TP + FN} \tag{2.7}$$

A interpretation of this definition is that recall is the probability that a relevant object is returned by the system.

2.6.3 F1-score

The F1-score is the harmonic mean between precision and recall and is calculated as follows:

$$F1 = \frac{2pr}{p + r} \tag{2.8}$$

Chapter 3
Development

The development chapter presents how the system was implemented. The system consists of four major parts:

1. An implementation of frequent pattern growth (FPG).
2. Our application, which is a association rule mining with K-mode clustering (ARM-KC).
3. An Qlik Associative Engine (QIX) implementation.
4. An evaluation section in which we calculate precision, recall and F1-scores of the different algorithms.

The FPG implementation serves two purposes. Firstly, it works as a baseline in order to evaluate other algorithms. The second function is that it works as the first step in the ARM-KC application. We designed ARM-KC to further increase the ability to make predictions with association rule mining by utilizing clustering to segment customers.

The QIX implementation is an application designed to make real-time predictions without the need of training the algorithm. In the evaluation part, we evaluate both FPGs and ARM-KCs capabilities to make accurate predictions.

We implemented the system in Python and used external libraries for the K-Mode and FPG algorithms.

3.1 pandas

A lot of the data preparation was done using the open source python library *Python Data Analysis Library* (pandas). The purpose of pandas is to provide tools for working with data analysis in Python. We used Pandas for the following purposes:

- Read and write csv files
- Create pandas dataframe out of pickle files
- Manipulate dataframes such as concatenating several dataframes, drop and add rows, sum, count and locating values.

3.2 Data Cleaning

Because of the anonymized structure of the data, the first step of the cleaning process was to remove irrelevant columns. As can be seen in Figure 3.1, only a few relevant attributes remained.

In the second step of the cleaning process, we removed $NULL$ values and duplicates.

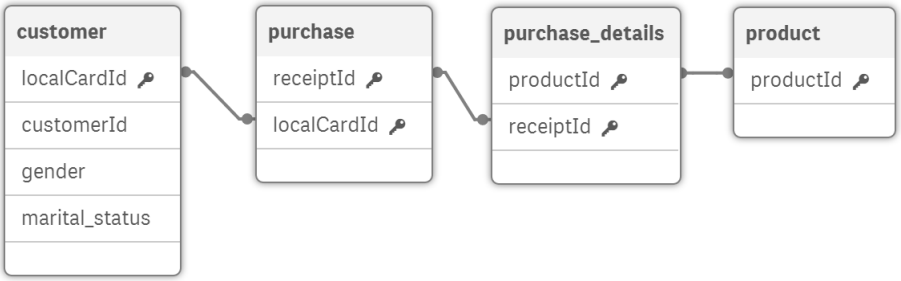

Figure 3.1: Structure of the receipt data after process

3.3 Frequent Pattern Growth

The implementation of Frequent Pattern Growth (FPG) serves two purposes:

- Firstly, to work as a baseline when comparing precision and recall between algorithms. FPG is one of the best known association rule mining algorithms and faster than the classic Apriori algorithm Garg and Gulia (2015). Therefore, it was a natural choice as baseline.
- The second purpose, and the most important, was to take the association rule set that FPG creates and use it on further applications. FPG makes a one dimensional set of relations between products. The goal was to make a customer segmentation using these item relations. So it was of greatest importance that the rules from the FPG algorithm could easily be leveraged in further researches and algorithm implementations.

The number of rules the FPG algorithm generates is dependent on what percentage of support and confidence is provided to the algorithm. Different support and confidence may vary the output from none to millions of rules, with high support and confidence giving less rules. The performance of the algorithm is also dependent on support and confidence. Because of time limits and hardware limitations, we set these parameters to relatively high values.

Since the performance was not a priority, we used the standard Python machine learning library `pyspark.mlllib.fpm.FPGrowth`. This library is an implementation of Han et al. (2000) description of FPGrowth.

The output from the FPG algorithm is a set of rules describing relations within the tables. Table 3.1 is a snippet from our implementation describing relations between products. The first line describes that products 1065 and 78 occur 2048 times with product 2748. It also shows that when 1065 and 78 appear together on a receipt, there is a 0.9 probability that 2748 is also present on that receipt.

From	To	Support	Confidence
{1065, 78}	{2748}	2048	0.90
{78, 2153, 1065}	{2748}	1866	0.91
{2748, 1065}	{78, 2153}	1866	0.83
{741}	{2748, 78}	1831	0.32

Table 3.1: Assignment of rule codes

An example can be seen in Equation 3.1

$$15, 17 \Rightarrow 49, support = 0.2, confidence = 0.1 \tag{3.1}$$

3.4 ARM-KC

The main goal with the implementation of a K-mode algorithm is to make a broad customer segmentation. The FPG provides association rules that describe the relationship between different rules. However, it does not make any assumptions about the customers. Our implementation aims to cluster customers who fulfil similar association rules. The implementation also utilizes the K-mode algorithm's categorical clustering in order to tie customers with similar buying patterns to each other.

Our implementation can be divided into 6 steps as shown in Figure 3.2, starting with data cleaning and FPG rule generation as described in the previous sections. Step 3-6 is our further development of creating a recommendation engine based on customer segmentation, utilizing the K-mode algorithm. The output of the solution is a customer recommendation based on this customer's specific purchase history.

3.4.1 Data Preparation

In order to transfer the association rules into the K-mode algorithm, the data must be in the appropriate format. The K-mode algorithm uses one or more columns with multiple values when clustering. Therefore association rules in their regular form are unusable for the K-mode algorithm.

Utilizing one-hot encoding variables, we can create orthogonal vectors for each of the different customers, making a matrix that tells which customers satisfy each rule.

1. The first step in creating this matrix is to assign each unique rule to a corresponding, unique rule number or rule code. Table 3.2 shows the first three rules, named $\boldsymbol{R}_1$, $\boldsymbol{R}_2$ and $\boldsymbol{R}_3$.

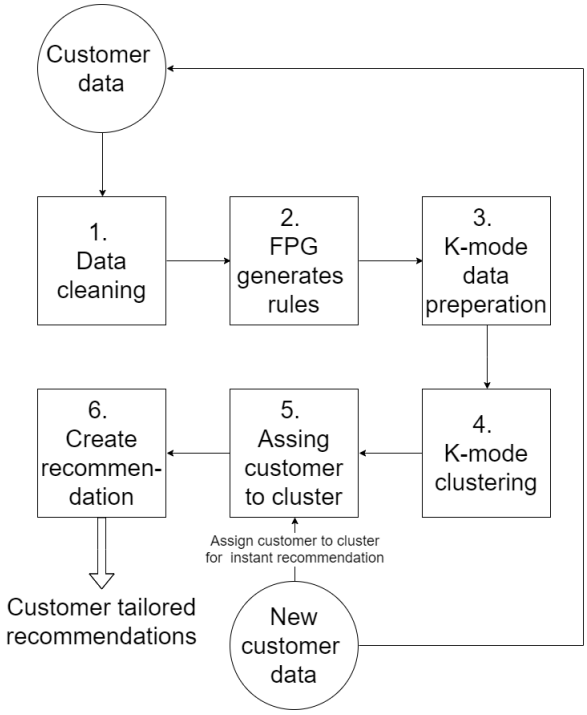

Figure 3.2: The different steps when creating recommendation with the ARM-KC implementation

2. The second step is to transform these numbers into vectors where, if the rule is satisfied, it is marked as a 1, otherwise as a 0.

3. The third step is to create a vector for each customer containing his/her corresponding rules, in order to use it for clustering. If a customer satisfies the rules R_1 and R_3, but not R_2, the customer vector is $(1, 0, 1)$.

Rule name	Association rule
R_1	$1065, 78 \Rightarrow 2748$
R_2	$2748, 1065 \Rightarrow 78, 2153$
R_3	$741 \Rightarrow 2748, 78$

Table 3.2: Assignment of rule codes

With every customer represented as a rule-vector, this procedure creates a matrix as can be seen in Table 3.3. This matrix is in the correct format for the K-mode algorithm to utilize for creating clusters.

3.4.2 K-mode Algorithm

To cluster with K-mode, we used Nico de Vos python library KModes (de Vos, 2020). De Vos implemented the first version of the K-mode algorithm that clusters categorical data (Huang, 1998). Running the algorithm on Table 3.3 leads to two things:

Customer Id	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	...	R_{388}
1	1	0	0	1	0	0	1	0	0	1	...	1
2	1	0	0	0	0	0	1	0	1	1	...	0
3	1	1	0	0	0	0	0	0	0	0	...	0
4	0	0	1	0	0	0	1	0	0	0	...	0
5	0	0	0	1	0	0	1	1	0	0	...	0
...	...	...	...	...	...	...	...	...	...	...	...	...
45228	0	0	0	1	0	0	1	1	0	0	...	0

Table 3.3: Assignment of rule codes

1. The algorithm creates centroid vectors with 0s and 1s creating the centers of the clusters. The number of clusters is decided by the user. An example of a 10 customers placement in three clusters can be seen in Figure 3.3.

2. The algorithm assigns each customer to the closest centroid using Hamming distance.

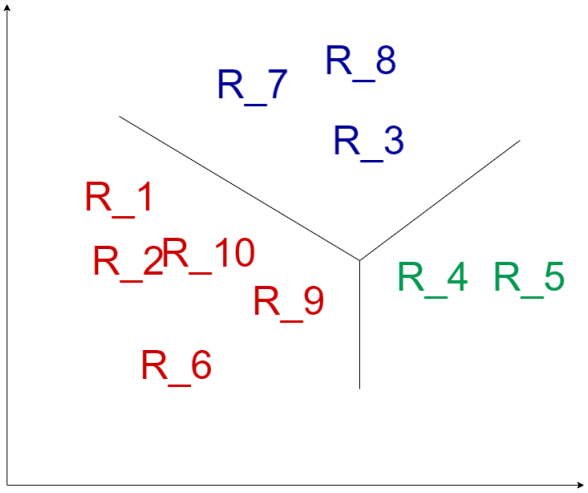

Figure 3.3: Example of three clusters with a total of 10 rules

Our algorithm is highly dependent on what rules the FPG algorithm produces. Therefore the number of rules was strongly believed to influence the accuracy of the K-mode algorithm. Hence, the K-mode algorithm was executed with different number of clusters and FPG rules. The rules experimented with are; *52*, *164* and *388*. The number of clusters experimented with are; *15*, *20* and *25*.

3.4.3 K-mode Recommendations

The final step of the recommendation engine is to output recommendations for specific customers. Since each cluster contains a set of rules and each customer is assigned to a cluster, recommendations can be made by recommending all the other products belonging in the cluster, except for those that the customer has already bought. Given a customer placed in cluster C_1 based on satisfying rule R_1 and $R_1, R_2, R_3 \in C_1$, the algorithm will recommend

the consequent, meaning the right-hand-side of R_2 and R_3. If Table 3.2 is an representation of C_1, the recommendations would be products *15, 19*. In order to assign a customer to a cluster, the customer always needs to satisfy a minimum of one of the clusters rules. Hence, the algorithm is limited to only making predictions for customers with a purchase history.

The products are ranked according to their support, with higher support leading to higher rank.

3.4.4 Filtering customers

As mentioned in the previous section, recommendations were made based on the centroids of the clusters. One of the clusters, no matter how the algorithm was tuned, contained a centroid that was only zeros. A cluster with only zeros means that there are no rules in the cluster and therefore no recommendations can be made from that cluster. Therefore, the application comes with an option to filter away the customers that are assigned to the zero cluster. In the evaluation step of the thesis, ARM-KC is evaluated with both the zero cluster included and without.

3.5 Qlik Associative engine

We found that FPG and ARM-KC had two major limitations. Firstly, the model needed to be trained with every new addendum to the data set. Secondly, it could only create recommendations if the products that the customer bought were present in the trained model. To overcome these limitations, we used the Qlik Associative Engine (QIX) API to implement an associative rule mining algorithm. QIX data handling is based on a associative database model. That makes the engine faster for handling the selections compared to regular query based database structures. The goal was to make a QIX implementation that mimics the association rule mining ability to make customer predictions based on the same receipt data as used by ARM-KC set but without the need of a training phase.

The application was successfully implemented in a way that it can make real time predictions on the entire data set without any previous training. With a product combination, an output recommendation is created with all the data. This means that every time a new recommendation is made, all the calculations will be redone. This also means that even if the data is updated frequently, the calculations will always be made on the latest data. In the application, it is possible to select gender and martial status for the customer recommendation. If for example the attribute male is selected, the recommendation will only be based on purchases done by males.

This also led to limitations on the application in such a way that it is only possible to create recommendations for one product combination at a time. FPG and ARM-KC create a model with the most frequent rules. To create this kind of model with the QIX implementation, the algorithm needs to iterate through every possible product combination, which would be too computationally expensive. Another restriction on the application is that the output recommendations are limited to one product for each input combination. On the contrary, other association rule algorithms allow multiple products as a recommendation.

The implementation in QIX was based on the QIX function *set analysis* (qlik.com, 2020). Set analysis makes it possible to customize data selection in a way that specific unions and

intersections are created based on the current selection. This is used in our algorithm to create the following functionality:

1. A user selects the purchased products that the recommendation should be based on. Let it be P_1 and P_2.
2. All receipts where the product combination P_1, P_2 occurs are selected.
3. On these receipts, all other products are counted. These products are the consequent to the antecedent P_1, P_2 and the count is the support for this rule.
4. Lastly, the confidence is calculated by diving the support with the number of times the combination P_1, P_2 occurs in the receipt in total.

The two expressions used in our QIX implementation can be seen below:
To calculate the support:

```
Count(distinct {<productId=,
receiptId={"=aggr(count(distinct productId),
receiptId) =getselectedcount(productId)"}>} receiptId)
```

To calculate the confidence:

```
Count(distinct {<productId=,
receiptId={"=aggr(count(distinct productId),
receiptId) =getselectedcount(productId)"}>} receiptId)/
(max( Total AGGR(Count(distinct {<productId=,  receiptId =
{"=aggr(count(distinct productId),
receiptId)=getselectedcount(productId)"}>}
receiptId),productId)))
```

Configurations

The QIX application is implemented in Javascript using Qlik's enigma.js library. The program has five different configurations:

- Products: The product combination that the customer has bought.
- Number of rec: How many output recommendations should be made.
- Gender: The gender of the customer in the data.
- Marital status: Whether the customer is married or not.
- Subsets: The number of the smallest subsets. If the subset is selected as 2, with products P_1, P_2 and P_3, the program will output recommendations for subsets P_1, P_2, P_1, P_3 and P_2, P_3.

The output is:

- The recommendations generated by the product combination with its associated support and confidence.
- Recommendations generated by the product sub-combinations with the combinations associated support and confidence.

Enigma.js

The library used to implement QIX is enigma.js. Enigma.js is a javascript library created to ease the communication with the QIX-API:s. The library sets up a websocket connection against QIX-engine and communicates over the websocket using JSON-schemas.

3.6 Train- and Test set

In order to evaluate the performance of the developed recommendation engine, a test set was created. The test set consisted of the same receipts as the train set. However, every receipt was divided. Three ways of division were compared; 50/50, 70/30 and 90/10.

The division was performed by grouping all purchases of the set by customerId. The purchases of the customer that are saved into test- and train set are chosen randomly.

3.7 Evaluation

As mentioned earlier, the test set consisted of 30 percent of every receipt. The larger part, the train set was used to train the algorithm and receive a recommendation. The smaller part, the test set was used to calculate F1, recall and precision.

3.7.1 ARM-KC

The F1 score was calculated for the entire recommendation but also for each cluster. No extra packages were used in order to calculate precision, recall and F1 score.

Instead, a recommendation for every customer was received by training the algorithm with the full set and predicting for the train set. This vector was called y_pred. The products in y_pred were compared with the products of the customer in the test set, which was called y_true.

The evaluation of the algorithm "ARM-KC" was done on three different set of rules; 52 rules, 155 rules and 388 number of rules.

3.7.2 FPG

The first step to evaluate FPG was by using the train set to create the FPG rules. This gave us recommendations for each customer, y_pred. These product recommendations were then compared to the products in the test set to obtain a vector with correct recommendations.

The FPG algorithm was only evaluated using the set of rules that performed the best during the evaluation of the algorithm "ARM-KC", namely 388 number of rules.

Chapter 4
Results

In this chapter, we present the results of our evaluation of ARM-KC. The first sections present F1-scores with different configurations running the ARM-KC algorithm. The following section compares ARM-KC with the baseline we have used, namely Frequent Pattern Growth, which is the industry standard algorithm for making customer recommendations. The last section explicates the result of the implementation based on Qlik Associative Engine.

4.1 ARM-KC Configurations

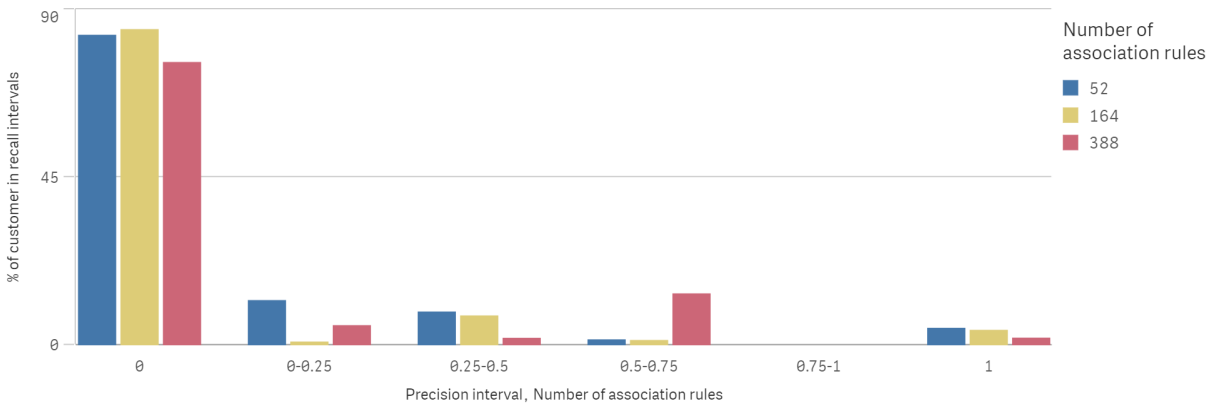

Figure 4.1: Precision for different number of rules

4.1.1 Different Number of Rules

The y-axis of Figure 4.1 shows the percentage of customers whose precision is in the different ranges shown on the x-axis. As can be seen in the Figure, around 80% of the customers get 0

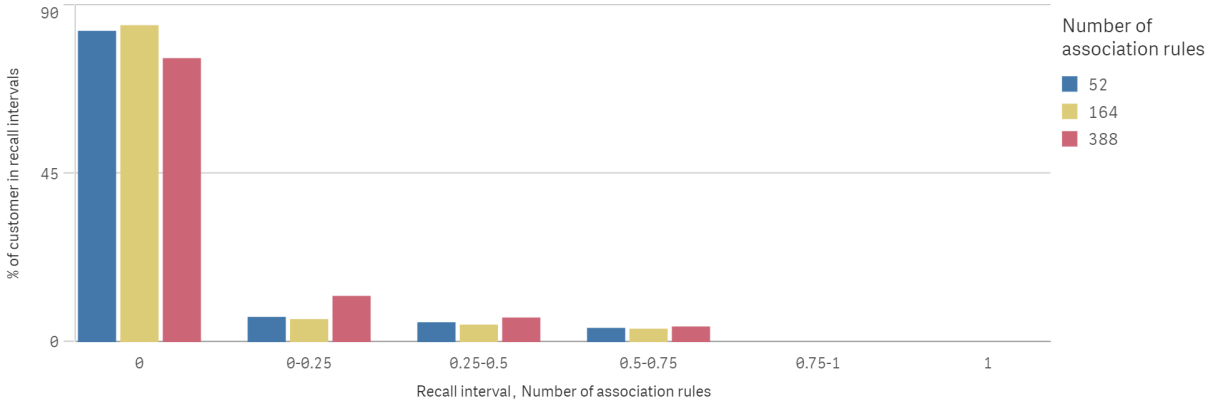

Figure 4.2: Recall for different number of rules

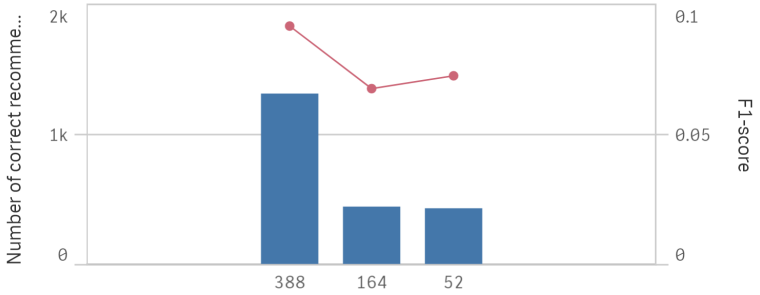

Figure 4.3: ARM-KC with different number of FPG-rules

correct recommendations, whereas 388 rules having 76% and 164 having 85%. Figure 4.2 has the same layout as the precision graph but shows the recall ranges of the evaluations instead of the precision ranges.

Figure 4.3 shows the F1-score and the number of correct recommendations when running the algorithm with different number of FPG-rules as input. The graph shows the ARM-KC algorithm run with 52, 164 and 388 rules. 388 rules provides both the highest number of correct recommendations and the highest F1 score. 164 provides a few more recommendations than 52 rules. However, 52 rules gives a better F1 score than 164 rules and therefore gives a higher precision.

The recommended set of rules should therefore, comparing the three random choices evaluated, always be 388. An interpretation of Figure 4.3 is that the F1-score is not linearly dependent on the number of rules. However, with a higher number of rules the algorithm is able to predict more recommendations correctly.

4.1.2 Different Number of Clusters

Figure 4.4 shows the result when differing the number of clusters in the K-mode algorithm. In the same way as Figure 4.3, it shows the number of correct recommendations and F1-score. This graph describes that the F1-score gets linearly worse with more clusters. This is noticeable since intuitively, more clusters should increase the F1-score. Another result that the graph suggests is that there is no correlation between the number of clusters and the number of correct results.

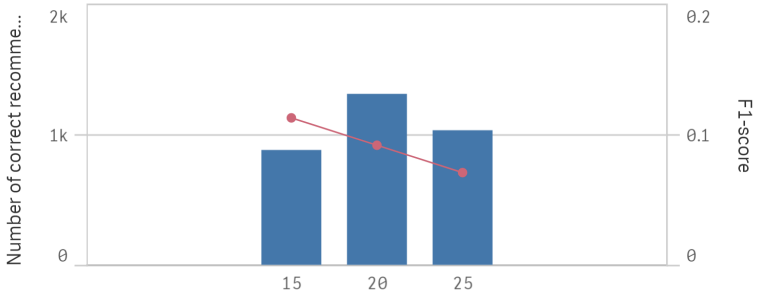

Figure 4.4: ARM-KC with different number of clusters

4.1.3 Different Split Ratios

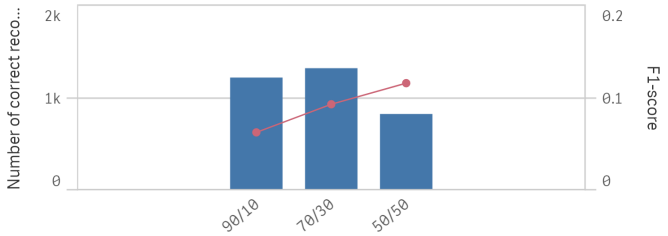

Figure 4.5: Example of three clusters with a total of 10 rules

ARM-KC was evaluated using a train set and a test set, where the train set contained a part of every receipt and the test set the rest. For example, if a receipt consists of products {1, 2, 3, 4, 5, 6} and the split ratio is 50/50, three of the products are placed in the train set and three are placed in the test set. This is done for every receipt in the set.

Figure 4.5 shows the evaluation of the ARM-KC algorithm with different split ratios between the customer product predictions and answer set. In the Figure, a 90/10 split ratio can be seen. The first part describes that 90 percent of the customers purchase history is used to make predictions and 10 percent is used to evaluate the predictions. The two y-axis describe, as the two previous figures, the number of correct recommendations and the F1-score. In this figure the F1-score is increasing linearly when the answer part of the customers purchases grows. However, it can be seen that the number of recommendations drops with 40% when the ratio is changed from 70/30 to 50/50. From the graph it is also clear that 90/10 has no advantage over a 70/30 split as the 70/30 split gives both more recommendations and more accurate recommendations.

4.2 ARM-KC Compared to FPG

Figure 4.6 shows ARM-KC's interval of precision compared to the FPG's. The graph is based on FPG and ARM-KC evaluated with 388 FPG rules and a predictive/answer split ratio of 70/30. ARM-KC was clustered using 20 clusters. The customers are filtered in a way where customers that are placed in the K-mode algorithm's 0-cluster are removed. The figure shows that FPG provides 0 correct predictions with 5% more customers than ARM-KC.

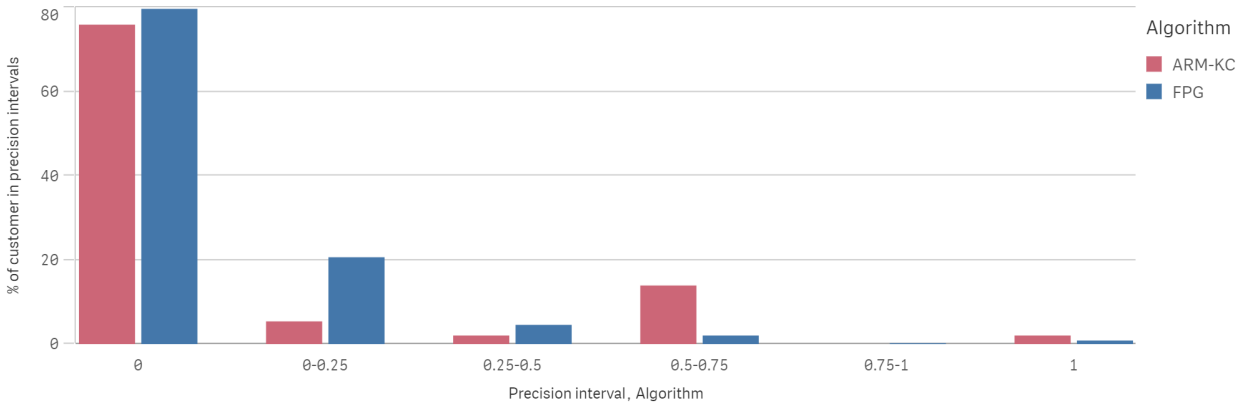

Figure 4.6: FPG's and ARM-KC's precision compared

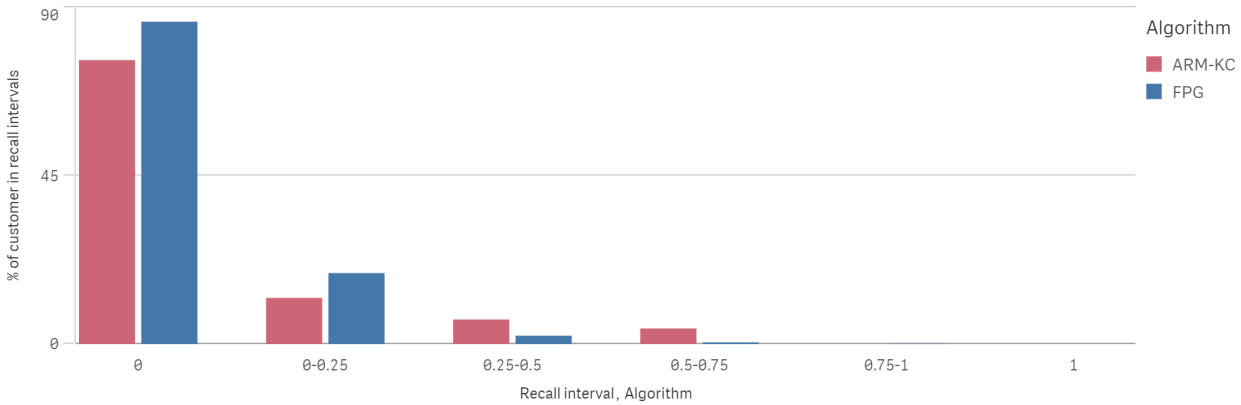

Figure 4.7: FPG's and ARM-KC's recall compared

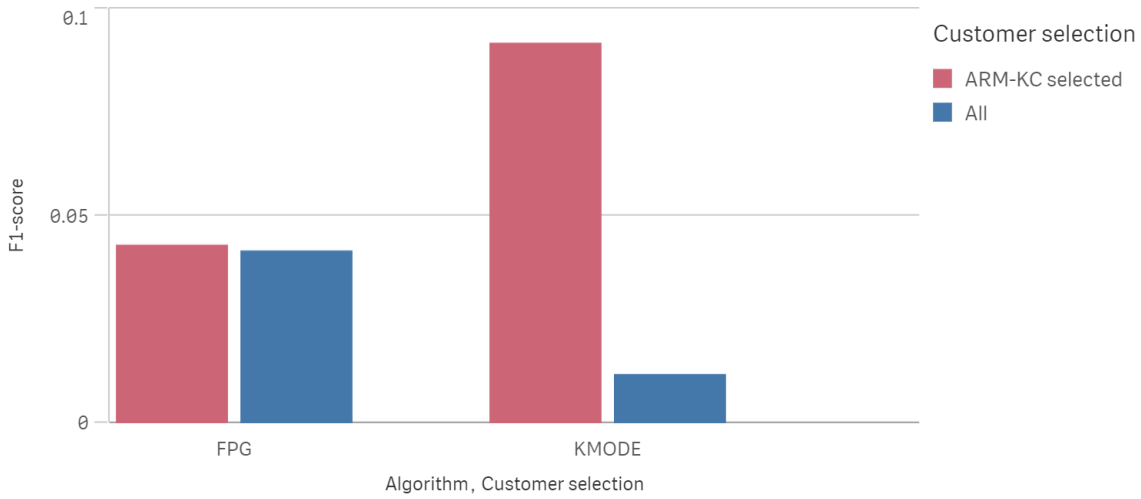

Figure 4.8: FPG's and ARM-KC's F1 scores compared

Figure 4.7 is similar to Figure 4.6 but shows the recall intervals instead of precision intervals. In this figure we see that the 0 prediction bar is 11% higher with FPG than ARM-KC.

Figure 4.8 shows a comparison between FPG's and ARM-KC's evaluated F1-scores. The blue bars describe the algorithms F1-scores when evaluating with all customers that satisfy at

Zero cluster included	Number of customers
No	5,422
Yes	41,540

Table 4.1: Number of customers with different filtration

least one FPG-rule. The red bars on the other hand show the F1-score for only the customers not filtered away by ARM-KC's for being in the zero cluster. The FPG gives roughly the same result as it did previously. The ARM-KC gives a 9 times higher F1-score with the zero cluster filtered away. The number of customers that the different filtration settings can be seen in Table 4.1.

4.3 F1-Score in Different Clusters

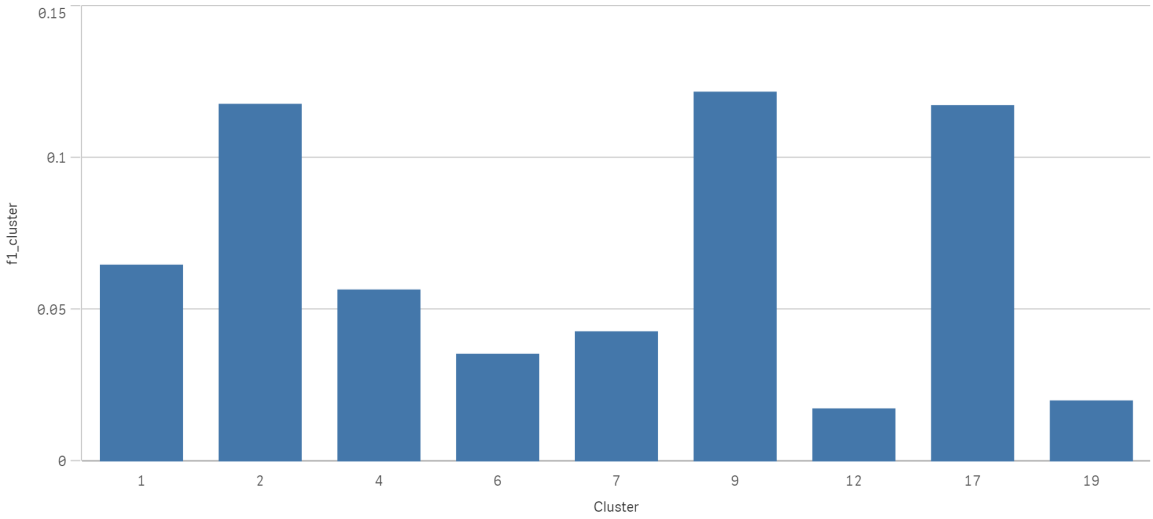

Figure 4.9: F1-scores for different clusters

Figure 4.9 shows the F1-score for the different clusters. This means that the F1-score is evaluated for each K-mode cluster. The algorithm was executed with 20 clusters, 388 FPG-rules and evaluated with a 70/30 split ratio. There is a significant difference in performance between the different clusters. For example cluster 2, 9 and 17 have an approximately 6 times higher F1-score than 12 and 19. Overall, cluster 9 has the highest F1-score with 0.12 and cluster 12 has the lowest with 0.2. This gives a difference of 0.1 between the best and the worst predicted cluster.

This means that for clusters *2*, *9* and *17*, it is easier to make predictions.

4.4 Qlik's Associative Engine Implementation

Since the Qlik's Associative Engine (QIX) implementation is based on FPG, the F1-scores produced by QIX would be the same as for FPG. It performs slightly worse because of the limitation of the algorithm that it can only output single FPG consequent for each rule. The result of the QIX-implementation is more related to benchmarking. Also the use case differs which makes it impracticable to evaluate with our evaluation methods. Therefore, the QIX-implementation will be evaluated in the discussion.

Chapter 5
Discussion

5.1 Evaluation

5.1.1 Anonymized Data Set

Customer Features

We clustered customers based on what they had purchased. One natural extension would be to associate the clusters to demographic or lifestyle labels. An example of a label would be Parents" for a cluster containing customers that had purchased a lot of products specific to parents. Such characteristic products would for example be a cot, a child seat and a baby gym.

Another label would for example be "House renovators". Customers belonging into this cluster would have purchases such as laminate floor coverings, doorknobs and floor decking.

Given data with original description of products, a customer could have received the appropriate labels depending on what cluster they belonged. The recommendation could have consisted of products belonging that label.

The anonymization makes it impossible to add features to the customers. Had it been possible to add features such as "Parent" or "House renovator" the algorithm would have led to a much better understanding of the customers in the future as well. Would IKEA add a new product, it would be easier to know which customers to recommend it to by just checking the information about them. This would save the time that otherwise would go to run through the entire ARM-KC to include one or a few new products.

Cluster Interpretation

Working with data that isn't anonymized gives the possibility to make sense of the clusters. The K-mode algorithm places the customers in the clusters they belong to depending on what

their purchase history tells about them. This means that clusters should contain association rules consisting of products in similar categories. A possibility here would be to calculate which categories are the most common in the cluster and a name for the cluster would be possible.

Spam Reduction

In the case of named clusters, one possible evaluation would be to check which product categories that are easier to recommend correctly. This would be done by evaluating the clusters with appropriate tools and comparing them to each other. A possible usage for this kind of evaluation would be to focus on recommending the categories in the highest ranked clusters. This would be useful since recommendations usually require sending out a lot of information before reaching a customer that is affected by it. Customers can perceive this as spam, which might cause the company to lose them. By focusing on categories with lower risk of wrongful recommendations, spam could be avoided at a higher rate.

5.1.2 ARM-KC

Number of FPG Rules

The number of FPG rules used to develop ARM-KC affects how well the program evaluates. This can be seen in Figure 4.3. We started with 164 rules and decided to try a higher and a lower number. Both cases gave higher F1 scores but 52 rules gave a lower number of correct recommendations. 388 rules was better both in the case of F1 score and on the number of correct recommendations. A fair assumption is that it would lead to even better results by going even higher up regarding the number of rules. However, the evaluation of ARM-KC is done by comparing to the original FPG algorithm. Therefore, trying out higher numbers of FPG rules would not contribute much since the original FPG algorithm would also improve. We chose to maintain our program with 388 FPG rules.

Split Proportion of Test/Train Set

As seen in Figure 4.5, the highest F1 score is obtained by splitting the set 50/50 and the lowest by splitting 90/10. However, we chose to split the set 70/30. The reason for this is that it provides a higher number of recommendations compared to the split 50/50. On the other hand, it provides a lower number of recommendations than the split 90/10 but a higher F1 score. The combination of a reasonably high F1 score and number of recommendations led to us choosing the 70/30 split. The reason that the 70/30 split is the best might be that having fewer products in the basket gives too bad recommendations due to poor predicting and having too many products in the train set leaves too few to recommend.

Number of Clusters

In K-mode, an important aspect is to choose a right number of clusters to divide the customers into. A commonly used method is *the elbow method*.

However, the data set we worked with in this project does not have a natural clustering, meaning that some customers are supposed to be in the same cluster because they only have things in common with each other. In our data set, most customers have purchases in common with most other customers. This leads to errors when using *the elbow method*. Instead we used trial and error to decide the best number of clusters. The starting number was 20. To decide which direction to go after that, we chose to try a higher and a lower one as can be seen in Figure 4.4.

The decisions were made by examining the variables in Table 5.1.

Number of clusters	Correct Number of Recommendations	F1 score
15	879	0.11
20	1308	0.09
25	1030	0.07

Table 5.1: Comparison of different cluster numbers with Correct number of recommendations and F1 score

Since a number of clusters higher than 20 gives a lower number of correct recommendations and also gives a lower F1 score, we decided that 20 clusters was the ceiling. 15 clusters however, require a prioritisation to be made. On the one hand, 15 clusters give a higher F1 score than 25 clusters. On the other hand they give a lower number of correct recommendations since more customers fall into a cluster where no recommendation can be made due to a centroid without FPG rules.

Since the F1 score only differs about 18 percent while the correct number of recommendations differs with around 32.8 percent, we chose 20 clusters as the best number and we did not investigate lower than 15.

Filtering customers

As seen in Figure 4.8, ARM-KC has a lower F1 score than the original FPG algorithm when calculated for all 41,546 customers. The reason for this is that when running K-mode we place customers in clusters based on the FPG rules in the centroids.

ARM-KC provides a higher F1 score than the original FPG algorithm if the F1 score is calculated based only on the customers that get at least one recommendation from ARM-KC. Even though it might seem that 5422 out of 41,546 is a small number to get a recommendation for, it is important to recognize that get a much higher accuracy for a part of the customers might give better sales than getting bad recommendations for all customers.

5.2 Algorithm comparison

The evaluation part gives a great understanding of how the application was tuned and which parameters were significant. However, creating the application was also about deciding which algorithms should be used and how they should be used.

5.2.1 Frequent Pattern Growth

The main goal when choosing the algorithm was to make predictions of the customer's next buy. The data set contained limited information about the customers. There were only three features provided, namely what they had bought, their marital status and their gender. Making customer segments that provided useful insights based on features was therefore limited. The industry standard for finding relationships between products is market basket analysis. Therefore market basket analysis was the natural first step for us based on the sparse data set. The most commonly used technique within market basket analysis is associative rule mining and since our data set was in a proper format, it was chosen as a first step. Within item rule mining there are several different algorithms. The most used is probably the popular Apriori algorithm. However, we decided to use another algorithm, namely frequent pattern growth (FPG), because of its faster computational time.

5.2.2 ARM-KC

Working with our data set it was obvious to create association rules between the products. However, just these rules do not give an understanding of the customer segments since they only explains products relations. The main goal with our ARM-KC was to divide the customers into sub groups and based on these sub-groups create recommendations. The expectations of the customer segmentation was to create more personalized recommendations and therefore increase the precision and recall compared to recommendations given by only running FPG.

There are a lot of different methods for grouping customers together. The intention was to create multiple groups of customers with similar buying patterns. A standard way of creating groups with similar characteristics is clustering. One of the most popular clustering algorithms is the K-means algorithm. However, K-means is not suitable for categorical data. But with the modified K-means algorithm, namely *K-mode*, it is possible to do clustering on categorical data.

In order for K-mode to create the clusters, every customer had to be represented with a vector. The appropriate form of the matrix can be seen in Table 3.3. This matrix structure of the rules leads to some complications which in turn leads to some limitations on the ARM-KC applications. One is that every rule creates a new feature in the matrix, which means that more rules lead to a wider matrix. The training time of the K-mode algorithm is very dependent on the width of the matrix. Because of our limited time and computational power, the matrix could not contain too many rules.

To only train with a few rules, on the other hand, leads to another problem. The structure of the IKEA data was that a few products were bought with a high frequency. Therefore, a lot of the rules contained the same products in different combinations creating a circular dependency. Since our application could not handle too many rules, some of the clusters contained very many rules and others very few. This led to the algorithms providing very precise and many recommendations for some customers while providing no recommendations at all for a lot of customers. As a result, which can be seen in Figure 4.8, ARM-KC created better recommendations then FPG on a smaller subset of customers but performed worse on the entire set.

5.2.3 Qlik Associative Engine

The FPG and ARM-KC algorithms have two major constraints:

- The first is that both algorithms need to be trained. The training process is time consuming. With bigger data sets, more rules and limited computer power the training time increases, in some cases exponentially. Even with our relatively small ARM-KC model with just 388 rules and 20 clusters it took between 4-5 hours to train. Another disadvantage with the training process is that if the data set changes, the rules no longer reflect the data set. Therefore, for precise recommendations that are completely based on the data, the model needs to be trained again.

- The second constraint is that the recommendation of products is limited to only the set of rules that is created in the training processes. All products that are not a part of the trained model will not be related to any recommendation. In our case, because of time limit, the training time needed to be kept low. This means that there were limited rules in our model. In our most extensive training phase, only 109 out of 45,141 products were consequents in the rules. This means that only 0.24% of IKEA's total products could be recommended with that FPG model.

Our QIX implementation was able to overcome both of these limitations. Without these limitations, a system was created that was easy to use and with some real-life advantages compared to the classic FPG algorithm.

One advantage is that the QIX algorithm is not limited to product patterns that occur more frequently than the support threshold. This means that all products will be taken in consideration by the algorithm when creating customer recommendations. FPG and ARM-KC is limited to only give recommendation about popular products, which on the other hand can be argued to be enough since a few products are responsible for most sales. With our QIX implementation however, this is no longer an arguing point since it gives recommendations for both popular and non-popular products. In the end, this leads to more complete and comprehensive customer recommendations.

Another advantage with the QIX implementation is that the algorithm will always be based on the current data. Every time a purchase pattern is selected, the application will be executed all over. This makes it possible for the application to see if the data set has changed since the last time. If the the data is changed, it can be reloaded and the algorithm will always be executed on the latest data set. This functionality is infeasible with both ARM-KC and FPG because of the training phase. Every time the data changes, the model needs to be trained again to represent the updated data as well. This training phase is time consuming so it would be impossible in a scenario where the data set is frequently updated.

Another improvement that the QIX implementation has over FPG and ARM-KC is the possibility to conveniently include other customer features. In our data set, the customers had two features: gender and martial status. These features were unsuitable to use in the FPG's and K-mode's training phase, but were relevant to understand the data set. With the QIX implementation it is possible to select gender and martial status. In this case, the calculations will be made with only data of this selection. If male is selected, only male's purchase history will be taken into the calculations. This is a large advantage in creating more customer specific recommendations. This functionality could be implemented by using the FPG and

ARM-KC algorithm as well. However, this would require a training phase with every different feature combination. In our data set there were 3 different alternatives in both marital status and gender. That would result in 9 (3^2) different combinations, which would require 9 training phases; each of them very time consuming. With 10 features and 10 alternatives it would require 10^{10} training phases while the QIX implementation would calculate recommendations instantaneously.

5.3 Further development

5.3.1 Parameter tuning

The ARM-KC algorithm's *number of clusters* and *number of rules* tuning was discussed in Section 5.1.2. We think that in further versions of ARM-KC, tuning these parameters further could lead to improved overall performance.

Both the F1 score and the amount of recommendations are dependent on the number of rules used as input for the ARM-KC algorithm. The number of rules used in our model was limited due to lack of time and computational power. This had the consequent that only 109 of over 40,000 products could be recommended. Therefore, in further developments of the application, a higher number of FPG rules should be used for ARM-KC in order to create more recommendations.

In our evaluation, we limited ARM-KC to 15-25 clusters, which is a short span. There was no clear correlation between *the number of clusters* and the performance, so further evaluation on the number of cluster should be done. This includes evaluating both less than 15 and more than 25 clusters.

5.3.2 A non-anonymized data set

Because of the anonymized structure of the data set, there was sparse information about the customers that was useful. With a non-anonymized data set, including more features, the ARM-KC could be used to its full potential.

Interpretation of the clusters

To make more sense of clustering, labeling and interpretation of the clusters are important. Within customer segmentation, it is even common practice to name the customer segmentations. A big problem with the current data set is that it is impossible to give the clusters labels. If the data set contained non-anonymized features, it would be easier to create customer statistics for each cluster, which would have made it possible to label the clusters.

Clustering with additional features

The ARM-KC application is built to cluster depending on categorical data. Therefore, including more features than only purchased products might improve the results. In the IKEA data set, there was two additional features, marital status and gender, that could have been utilized in the clustering. However, the downside of adding more features is that it might

remove the purposes of what was discussed in Section 5.3.2, since the cluster would be based on customer characteristics more than buying patterns.

5.3.3 Qlik implementations

In the current version of our QIX implementation, the biggest limitation is that it does not create a set of the most frequent rules like FPG and ARM-KC do. There are many market basket analysis use-cases where a rule set is desirable. An example of such a use-case is physical item placement in a store. To create this kind of rule sets, the QIX-implementation has to iterate through every possible combination of products. Over 45,000 products makes this infeasible. However, in further development on streamline selection, a suffix-based pattern exploration could be utilized. Since FPG is based on suffix-based pattern exploration, we see no reason why our QIX implementation shouldn't be at least as fast as FPG without losing it advantages.

A current flaw with the QIX-implementation is that it can only output one product as consequent, whereas FPG can have a set of output products. With the 388 FPG rules that were used when we compared the algorithms, this limitation was not very noticeable because almost all of the rules only had one consequent. But with more rules, a lot more consequents with multiple products will occur.

Chapter 6
Conclusion

In this thesis, we have implemented two personalized recommendation engines that can be used in different scenarios in *market basket analysis*. One recommendation engine provides customer segmentation where only purchase history is given beforehand as knowledge about the customers. We call this recommendation ARM-KC and it performs three main steps in order to produce recommendations:

- Create association rules of the products with Frequent Pattern Growth (FPG).
- Use the association rules to create new characteristics for the customers.
- Cluster the customers with K-mode.

Another option could be to recommend products to customers by just using the association rules given by running FPG. This would save the time it takes to run K-Mode. As seen in Table 4.8, FPG provides basically the same F1 score for a small part of the customers in the set as it does for the entire set. The reason one might use ARM-KC can also be seen in Table 4.8. There we see that for a small number of customers in the set, the F1-score for the recommendations becomes higher than the FPG due to further personalizing. Although it only applies to a small number of the customers in the set, one might argue that it increases the sales for the company and lowers the risk of spam.

One more vital improvement provided by the ARM-KC compared to classic FPG is that more knowledge about the customers can be extracted. Since the clusters can be evaluated on their own, the most important clusters can be prioritised. For example, if the cluster with the highest F1-score contains mainly products bought by parents and house owners, the company might want to focus their recommendations on these sorts of products and customers. We worked with an anonymized data set and therefore had limited possibilities to explore this further.

The other recommendation engine in this thesis is an association rule mining algorithm implemented by using the Qlik Associative Engine (QIX) API. The need for this solution

comes from the time it takes to train classic FPG and ARM-KC. For certain scenarios, for example when new products and customers are added to the data set frequently, or a meeting with a new customer is being held and a recommendation is needed quickly, a solution that requires no training can be useful.

Of course , this solution also has downsides compared to classic FPG and ARM-KC. The solution takes a long time to produce association rules and recommend for more than a few customers.

When implementing a recommendation engine, one might choose a frequently used algorithm and settle with the good results that is provides. However, a better approach is to modify the algorithms for your specific needs in order to obtain the best outcome. For our specific need of obtaining more personalized recommendations, we managed to double the F1 score for some customers with ARM-KC compared to the results given by the industry standard algorithms. For our specific need of getting quick recommendations on frequently altering data, we managed to do so in insignificant time with the QIX implementation.

Bibliography

Abbas, W., Ahmad, N., and Binti Zaini, N. (2013). Discovering purchasing pattern of sport items using market basket analysis. *2013 International Conference on Advanced Computer Science Applications and Technologies, Advanced Computer Science Applications and Technologies (ACSAT), 2013 International Conference on, International Conference on Advanced Computer Science Applications and Technologies*, pages 120 – 125.

Agrawal, R., Imieli'nski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases.

Agrawal, R., Srikant, R., et al. (1994). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499.

Alberti, M. E. (2016). Trade and weighing systems in the southern aegean from the early bronze age to the iron age: How changing circuits influenced glocal measures. In *Of Odysseys and Oddities: Scales and Modes of Interaction Between Prehistoric Aegean Societies and their Neighbours*. Oxbow, Reading, Massachusetts.

Bellman, R., Corporation, R., and Collection, K. M. R. (1957). *Dynamic Programming*. Rand Corporation research study. Princeton University Press.

Cao, L., Zhang, H., Zhao, Y., Luo, D., and Zhang, C. (2011). Combined mining: Discovering informative knowledge in complex data. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 41(3):699–712.

de Vos, N. (2020). kmodes 0.10.1. `https://pypi.org/project/kmodes/`.

Domingos, P. (2012). A few useful things to know about machine learning. *Communications of the ACM*, 55(10):78 – 87.

Everitt, B. S. (1980). *Cluster analysis*. Heinemann.

Fullerton, R. (2016). Segmentation in practice: An overview of the eighteenth and nineteenth centuries. In *The Routledge Companion to Marketing History*, page 94. Oxon, Routledge.

Garg, R. and Gulia, P. (2015). Comparative study of frequent itemset mining algorithms apriori and FP growth. *International Journal of Computer Applications*, 126:8–12.

Glas, F. (2015). Machine-learning techniques for customer recommendations. Master's Thesis Lund.

Goutte, C. and Gaussier, E. (2005). A probabilistic interpretation of precision, recall and f-score, with implication for evaluation. In Losada, D. E. and Fernández-Luna, J. M., editors, *Advances in Information Retrieval*, pages 345–359, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gutierrez, N. (2006). Demystifying market basket analysis. *DM Review Special Report*.

Hamming, R. W. (1950). Error detecting and error correcting codes. *The Bell System Technical Journal*, 29(2):147–160.

Han, J., Cheng, H., Xin, D., and Yan, X. (2007). Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1):55–86.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29(2):1–12.

Hong-Wei, Y., Zhi-Geng, P., Bing-Xu, and Ling-Min, Z. (2004). Machine learning-based intelligent recommendation in virtual mall. *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826), Machine Learning and Cybernetics, 2004. Proceedings of 2004 International Conference on, Machine learning and cybernetics*, 4:2634.

Huang, J. Z. (2009). Clustering categorical data with k-modes. *Encyclopedia of Data Warehousing and Mining*, page 246.

Huang, Z. (1998). Extensions to the k-means algorithm for clustering large data sets with categorical values. *Data Mining and Knowledge Discovery*, 2(3):283–304.

Johnson, S. C. (1967). Hierarchical clustering schemes. *Psychometrika*, 32(3):241–254.

Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2):129–137.

Mahajan, M., Nimbhorkar, P., and Varadarajan, K. (2009). The planar k-means problem is np-hard.

Minh, N. T. and Wu, Y.-H. (2019). Integrating meta-path similarity with user preference for top-n recommendation. *2019 International Conference on Technologies and Applications of Artificial Intelligence (TAAI), Technologies and Applications of Artificial Intelligence (TAAI), 2019 International Conference on*, pages 1 – 6.

Qlik (2020a). The associative difference. `https://www.qlik.com/us/-/media/files/resource-library/global-us/direct/datasheets/ds-the-associative-difference-en.pdf`.

Qlik (2020b). What is qix and why should you care? `https://developer.qlik.com/knowledge/tutorial/engineTutorial`.

qlik.com (2020). Set analysis and set expressions. `https://help.qlik.com/en-US/sense/November2019/Subsystems/Hub/Content/Sense_Hub/ChartFunctions/SetAnalysis/set-analysis-expressions.htm?l=en-EN`.

Ralambondrainy, H. (1995). A conceptual version of the k-means algorithm. *Pattern Recognition Letters*, 16(11):1147 – 1157.

Santosh, K. (2014). Centroid Based Clustering Algorithms- A Clarion Study. *Computer Science and Information Technologies*, 5(6):7309–7313.

Sibson, R. (1973). SLINK: An optimally efficient algorithm for the single-link cluster method. *The Computer Journal*, 16(1):30–34.

Suits, D. B. (1957). Use of dummy variables in regression equations. *Journal of the American Statistical Association*, 52(280):548–551.

Xu, R. and Wunsch, D. C. (2009). *Clustering*. Wiley.

Yanchang, Z., Huaifeng, Z., Longbing, C., Chengqi, Z., Bohlscheid, H., Wobcke, W., and Mengjie, Z. (2008). Combined pattern mining: from learned rules to actionable knowledge. In *AI 2008: Advances in Artifical Intelligence. 21st Australasian Joint Conference on Artificial Intelligence*, Centre for Quantum Comput. & Intell. Syst., Univ. of Technol., Sydney, NSW, Australia.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2020-03-12

EXAMENSARBETE Personalized Product Recommendations
STUDENTER Alexander Pålsson, Enis Hajzeri
HANDLEDARE Pierre Nugues (LTH), José Díaz López(Qlik)
EXAMINATOR Jacek Malec (LTH)

Skräddarsydda kundrekommendationer

POPULÄRVETENSKAPLIG SAMMANFATTNING **Alexander Pålsson, Enis Hajzeri**

Maskinlärning används idag inom detaljhandeln för att skapa personliga produktrekommendationer för kunder. Detta arbete presenterar två lösningar som kombinerar välkända maskinlärnings algoritmer med ny teknik vilket resulterar i förbättrade rekommendationer.

Att förstå sina kunder för att kunna öka försäljningen är något som företag försökt göra länge. Faktum är att man har hittat bevis som tyder på att de Aeganska samhällena på den grekiska halvön använde sig av kundsegmentering redan under bronsåldern. Förståelsen av kunders beteenden har många olika tillämpningar men den kanske viktigaste för företagen är att omsätta denna till reklam. I och med internet har mängden reklam som når kunder ökat explosionsartat. För konsumenter har detta lett till att man får så mycket reklam att det är nästintill omöjligt att filtrera alla olika erbjudanden. Därför finns det en stor risk för att den ökande mängden reklam inte leder till en ökad försäljning utan ger en helt motsatt effekt, eftersom konsumenter uppfattar reklamen som irriterande. För att stoppa denna negativa trend runt reklam och få kunder mer positivt inställda är en strategi att skräddarsy reklam för varje enskild konsument, d v s att reklamen innehåller produkter som hen kan se sig själv köpa.

Hur gör man för att skräddarsy reklam?

Idag tränar man datorer för att göra gissningar baserat på historiska data, en teknik som brukar kallas maskininlärning. De två kanske vanligaste maskininlärnings algoritmerna för att förutspå kunders beteende är Frequent Pattern Growth (FPG) och klustrering. FPG skapar rekommendationer genom att beräkna sannolikheter av att varor köps tillsammans baserat på köphistorik. Klustrering däremot grupperar kunder med liknade egenskaper i så kallade kluster. Sedan rekommenderar man alla inom samma kluster liknande varor. Det finns några nackdelar med båda dessa tekniker, t ex att algoritmerna måste tränas vilket kan ta dagar. En annan nackdel är att eftersom tekniken bygger på att hitta produkter med hög köpfrekvens gör att produkter som inte köps så ofta får väldigt få rekommendationer. För att lösa några av problemen med dagens direktreklam och algoritmerna FPG och klustrering, har vi inom detta examensarbete vidareutvecklat två olika system. Det första systemet förbättrar noggrannheten för de skräddarsydda kundrekommendationerna. Genom att kombinera FPG och klustrering på ett unikt sätt har antalet korrekta gissningar dubblerats för ett stort antal kunder. Det andra fokuserar på problemet med att algoritmerna normalt måste tränas. Med hjälp av Qliks QIX-API har en applikation tagits fram som rekommenderar produkter till konsumenter med samma precision som FPG, fast utan träning. Qliks snabba datahantering och förmåga att extrahera relationer i data gör det möjligt att ögonblickligen hitta samma associationer i kundkvitton som FPG. En bonus med QIX applikationen är att inga produkter utelämnas oavsett köpfrekvens.