RECOMMENDER SYSTEM CONSTRUCTION USING LATENT SEMANTIC ANALYSIS AND DATA MINING METHODS ON E-COMMERCE DATA

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

ARİF GÖRKEM ÖZER

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
COMPUTER ENGINEERING

AUGUST 2019

Approval of the thesis:

RECOMMENDER SYSTEM CONSTRUCTION USING LATENT SEMANTIC ANALYSIS AND DATA MINING METHODS ON E-COMMERCE DATA

submitted by **ARİF GÖRKEM ÖZER** in partial fulfillment of the requirements for the degree of **Master of Science in Computer Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. Mehmet Halit Oğuztüzün
Head of Department, **Computer Engineering** _____

Prof. Dr. İsmail Hakkı Toroslu
Supervisor, **Computer Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Tolga Can
Computer Engineering, METU _____

Prof. Dr. İsmail Hakkı Toroslu
Computer Engineering, METU _____

Prof. Dr. Ahmet Coşar
Computer Engineering, UTAA _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Arif Görkem Özer

Signature :

ABSTRACT

RECOMMENDER SYSTEM CONSTRUCTION USING LATENT SEMANTIC ANALYSIS AND DATA MINING METHODS ON E-COMMERCE DATA

Özer, Arif Görkem
M.S., Department of Computer Engineering
Supervisor: Prof. Dr. İsmail Hakkı Toroslu

August 2019, 83 pages

Recommender systems are developed to provide better recommendations to users of e-commerce applications. In addition to this goal, e-commerce applications benefit from their recommender systems to show advertisements to users, apply discounts on specific items. The task of recommending an item to a user is always a challenge; luckily, there are many methods developed to complete this task such as collaborative filtering, association rule mining etc. These methods mainly look at the co-occurrence of items; however, we think that user behavior on different items should be extracted by doing latent semantic analysis on the data. Latent semantic analysis is used for understanding the context of a text, we think that it can be used for providing recommendations by processing transactional data. The data used throughout this thesis work consists of transactions made in various e-commerce companies. In this thesis work, existing methods and proposed recommendation methods are examined and recommendation results on this data are shown.

Keywords: Latent Semantic Analysis, Singular Value Decomposition, Association Rule Mining, Sequential Pattern Mining, Collaborative Filtering

ÖZ

E-TİCARET VERİSİ ÜZERİNDE GİZLİ ANLAMSAL ANALİZ VE VERİ MADENCİLİĞİ YÖNTEMLERİ KULLANILARAK ÖNERİ SİSTEMİ GELİŞTİRİLMESİ

Özer, Arif Görkem
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. İsmail Hakkı Toroslu

Ağustos 2019 , 83 sayfa

Öneri sistemleri, e-ticaret uygulamalarının kullanıcılarına daha iyi önerilerde bulunmak adına geliştirilmektedir. Bu amaca ek olarak, e-ticaret uygulamaları, kullanıcılarına reklam göstermek ve belirli ürünlere indirim uygulamak adına öneri sistemlerinden yararlanmaktadır. Herhangi bir kullanıcıya bir ürün önermek her zaman için bir zorluk olmuştur, neyse ki bu zorluğu aşmak adına işbirlikçi filtreleme, birliktelik kuralı madenciliği gibi yöntemler ortaya atılmıştır. Bu yöntemler, esas olarak ürünlerin birlikte bulunmalarını incelemektedir. Bizim düşüncemize göre, farklı ürünler üzerindeki kullanıcı davranışları gizli anlamsal analiz yapılarak ortaya çıkarılmalıdır. Gizli anlamsal analiz bir metnin içeriğini anlamak için kullanılan bir yöntemdir. Bu yöntemin alışveriş verileri üzerinde önerilerde bulunurken kullanılabileceğini düşünüyoruz. Bu tez çalışmasında kullanılan veriler çeşitli e-ticaret sitelerinde yapılan alışveriş işlemlerini içermektedir. Bu veriler kullanılarak farklı yöntemler ve ortaya atılan yöntem incelenmiş ve sonuçları da bu teze dahil edilmiştir.

Anahtar Kelimeler: Gizli Anlamsal Analiz, Tekil Değer Ayrışımı, Birliktelik Kuralı Madenciliği, Ardışık Zamanlı Örüntü Madenciliği, İşbirlikçi Filtreleme

To my family and friends

ACKNOWLEDGMENTS

Firstly, I would like to thank my advisor Prof. Dr. İsmail Hakkı Toroslu for his valuable advice and knowledge. He is a very supportive, tolerating, encouraging person and he kindly supported me throughout the development of this thesis. It was always educatory to work with him.

This thesis work is done by using the transactional data provided by StockMount company, therefore, I would like to thank them for providing this data through their e-commerce integration software. Without their data support, it was not possible to develop this thesis.

I would like to thank Melike, she always supported me until the end. I also want to thank my friends Emre, Semih, Yusuf and Şeyma. There were times that I talked about and discussed my work with them. They were always patient and supportive.

Finally, I would like to thank my parents and my brother Furkan. They always believed in me, I am grateful to them for their unconditional love. Without their efforts, I would not be able to succeed in my life.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xix

LIST OF ABBREVIATIONS xxi

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 Proposed Methods and Models 1

1.3 Contributions and Novelties 2

1.4 The Outline of the Thesis 2

2 RELATED WORK 3

2.1 Recommendation with Collaborative Filtering 3

2.2 Recommendation with Content-Based Filtering 4

2.3 Recommendation with Knowledge Discovery in Databases 5

2.3.1 Association Rule Mining 5

2.3.2 Sequential Pattern Mining 6
2.4 Recommendation with Latent Semantic Analysis 7
3 TECHNICAL BACKGROUND 9
3.1 Latent Semantic Analysis with Singular Value Decomposition 9
3.2 Association Rule Mining with Apriori Algorithm 11
3.3 Sequential Pattern Mining with PrefixSpan Algorithm 12
4 DATA STATISTICS & PREPROCESSING 15
4.1 General Information 15
4.2 Data Preprocessing Steps 19
5 SINGULAR VALUE DECOMPOSITION (SVD) ON THE DATA 23
5.1 Introduction 23
5.2 Difficulties About Applying SVD on Data 23
5.2.1 Data Size 23
5.3 Id Virtualization & Translation 26
5.4 Constructing Feature Matrices & Normalization 28
5.5 Application of SVD 29
5.5.1 Choosing Number of Singular Values 30
5.6 Recommendation Model After SVD 31
5.6.1 Product-to-Product Recommendation 31
5.6.1.1 Recommendation 31
5.6.1.2 Evaluation - Single Fold 35
5.6.1.3 Evaluation - 10-Fold Cross-Validation 36
5.6.2 Customer-to-Customer Recommendation 39

5.6.2.1 Recommendation 39
5.6.2.2 Evaluation 41
5.7 SVD Recommendation Results 42
6 ASSOCIATION RULE MINING (ARM) ON THE DATA 55
6.1 Introduction 55
6.2 Data Preprocessing for ARM 55
6.3 Application of Apriori Algorithm 58
6.4 Mining Association Rules 59
6.5 Association Rule Evaluation 60
6.6 ARM Results 61
7 SEQUENCE PATTERN MINING (SPM) ON THE DATA 65
7.1 Introduction 65
7.2 Data Preprocessing for SPM 65
7.3 Application of PrefixSpan Algorithm 68
7.4 Evaluation of Frequent Sequence Patterns 69
7.5 SPM Results 70
8 CONCLUSIONS 77
REFERENCES 79

LIST OF TABLES

TABLES

Table 5.1 Singular values vs Accuracy (product-to-product recommendation) . 30

Table 5.2 Singular values vs Accuracy (customer-to-customer recommendation) 30

Table 5.3 SVD P2P - single, singulars=5, significance=0.2, data size = 200K . 42

Table 5.4 SVD P2P - single, singulars=10, significance=0.2, data size = 200K 43

Table 5.5 SVD P2P - single, singulars=50, significance=0.2, data size = 200K 43

Table 5.6 SVD P2P - single, singulars=100, significance=0.2, data size = 200K 44

Table 5.7 SVD P2P - single, singulars=250, significance=0.2, data size = 200K 44

Table 5.8 SVD P2P - single, singulars=500, significance=0.2, data size = 200K 45

Table 5.9 SVD P2P - single, singulars=750, significance=0.2, data size = 200K 45

Table 5.10 SVD P2P - single, singulars=1000, significance=0.2, data size = 200K 46

Table 5.11 SVD P2P - single, singulars=1500, significance=0.2, data size = 200K 46

Table 5.12 SVD P2P - 10-fold cross, singulars=50, exist >= 2, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5 47

Table 5.13 SVD P2P - 10-fold cross, singulars=50, exist >= 3, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5 47

Table 5.14 SVD P2P - 10-fold cross, singulars=50, exist >= 5, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5 47

Table 5.15 SVD P2P - 10-fold cross, singulars=100, exist >= 2, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5 48

Table 5.16 SVD P2P - 10-fold cross, singulars=100, exist >= 3, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5 48

Table 5.17 SVD P2P - 10-fold cross, singulars=100, exist >= 5, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5 48

Table 5.18 SVD P2P - 10-fold cross, singulars = 50, exist >= 3, low = 5, significance=0.2, data size = 2.5M, cust. thres. = 10, prod. thres. = 50 49

Table 5.19 SVD P2P - 10-fold cross, singulars = 50, exist >= 3, low = 5, significance=0.2, data size = 2.5M, cust. thres. = 20, prod. thres. = 50 49

Table 5.20 SVD P2P - 10-fold cross, singulars = 50, exist >= 3, low = 5, significance=0.2, data size = 2.5M, cust. thres. = 10, prod. thres. = 60 49

Table 5.21 SVD C2C - single, singulars=50, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5 50

Table 5.22 SVD C2C - single, singulars=100, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5 50

Table 5.23 SVD C2C - single, singulars=250, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5 50

Table 5.24 SVD C2C - single, singulars=500, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5 51

Table 5.25 SVD C2C - single, singulars=1000, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5 51

Table 5.26 SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 20, prod. thres. = 50, recommend **at least** 5 cust. for each alter. cust. 51

Table 5.27 SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 20, prod. thres. = 50, recommend **exactly** 5 cust. for each alter. cust. 52

Table 5.28 SVD C2C - 10-fold cross, singulars = 50, low = 5, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 50, recommend **at least** 5 cust. for each alter. cust. 52

Table 5.29 SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 50, recommend **exactly** 5 cust. for each alter. cust. 52

Table 5.30 SVD C2C - 10-fold cross, singulars = 50, **significance=0.2**, data size = 2.5M, cust. thres. = 10, prod. thres. = 60, recommend **at least** 5 cust. for each alter. cust. 53

Table 5.31 SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 60, recommend **at least** 5 cust. for each alter. cust. 53

Table 5.32 SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 60, recommend **exactly** 5 cust. for each alter. cust. 53

Table 6.1 ARM - Shopping carts and frequent product sets per partition (weekly - 200K transactions) 58

Table 6.2 ARM - Shopping carts and frequent product sets per partition (monthly, sup=0.0007, 200K transactions) 59

Table 6.3 ARM - Shopping carts and frequent product sets per partition (monthly, sup=0.001, 2.5M transactions) 59

Table 6.4 ARM Results - 10-fold cross, **momently carts**, min confidence=25%, **sup=5/num_of_carts**, data size=200K 62

Table 6.5 ARM Results - 10-fold cross, **weekly carts**, min confidence=25%, **sup=5/num_of_carts**, data size=200K 62

Table 6.6 ARM Results - 10-fold cross, **monthly carts**, min confidence=25%, **sup=5/num_of_carts**, data size=200K 62

Table 6.7 ARM Results - 10-fold cross, **momently carts**, min confidence=25%, **sup=7/num_of_carts**, data size=200K 63

Table 6.8 ARM Results - 10-fold cross, **weekly carts**, min confidence=25%, **sup=7/num_of_carts**, data size=200K 63

Table 6.9 ARM Results - 10-fold cross, **monthly carts**, min confidence=25%, **sup=7/num_of_carts**, data size=200K 63

Table 6.10 ARM Results - 10-fold cross, **monthly carts**, min confidence=20%, **sup=0.001**, data size=2.5M 64

Table 6.11 ARM Results - 10-fold cross, **monthly carts**, min confidence=20%, **sup=0.001**, data size=2.5M, **platform="epttavm"** 64

Table 6.12 ARM Results - 10-fold cross, **monthly carts**, min confidence=20%, **sup=0.001**, data size=2.5M, **platform="n11.com"** 64

Table 7.1 SPM - single fold, daily, 200K transactions 68

Table 7.2 SPM - single fold, weekly, 200K transactions 68

Table 7.3 SPM - single fold, monthly, 200K transactions 69

Table 7.4 SPM - single fold, momently, data size=200K 71

Table 7.5 SPM - single fold, daily, data size=200K 71

Table 7.6 SPM - single fold, weekly, data size=200K 71

Table 7.7 SPM - single fold, monthly, data size=200K 71

Table 7.8 SPM - 10-fold cross, monthly, yearly sequences, data size=2.5M . . 71

Table 7.9 SPM - 10-fold cross, monthly, yearly sequences, data size=2.5M, **platform="n11.com"** 72

Table 7.10 SPM - 10-fold cross, monthly, yearly sequences, data size=2.5M, **platform="epttavm"** 72

Table 7.11 SPM - 10-fold cross, weekly, yearly sequences, data size=2.5M . . . 72

Table 7.12 SPM - 10-fold cross, weekly, monthly sequences, data size=2.5M . . 73

Table 7.13 SPM - Example sequences, 10-fold cross, monthly, data size=2.5M, support=0.001 74

Table 7.14 SPM - Example sequences, 10-fold cross, monthly, data size=2.5M, support=0.001 75

Table 7.15 SPM - 10-fold cross, monthly, data size=2.5M, support=0.001 . . . 76

Table 7.16 Examples of monthly frequent sequences with support=0.001 76

LIST OF FIGURES

FIGURES

Figure 3.1 Visual explanation of SVD 10

Figure 4.1 Shopping-Product count histogram (pie) - 200K transactions . . 17

Figure 4.2 Shopping-Product count histogram (bar) - 200K transactions . . 17

Figure 4.3 Shopping-Product count histogram (pie) - 2.5M transactions . . . 18

Figure 4.4 Shopping-Product count histogram (bar) - 2.5M transactions . . 18

Figure 5.1 Target customer-target product set sizes (200K transactions) . . . 25

Figure 5.2 Target customer-target product set sizes (2.5M transactions) . . . 25

Figure 5.3 Customer and product id virtualization 27

Figure 5.4 Customer and product id translation 28

Figure 5.5 Cross-validation of product-to-product recommendation with 10 folds 37

Figure 5.6 Existing Product Count vs. Accuracy 38

Figure 5.7 Existing Product Count vs. Recommended Customer Count . . . 38

Figure 6.1 Shopping-Product count histogram (pie) - 200K transactions . . 56

Figure 6.2 Shopping-Product count histogram (pie) - 2.5M transactions . . . 56

Figure 6.3 Visual explanation for ARM procedure with 10-partitions 58

Figure 7.1 SPM Cross-Validation 67

Figure 7.2 Histogram of various sequence lengths - 200K transactions . . . 67

Figure 7.3 Visual explanation for sequence evaluation 70

LIST OF ABBREVIATIONS

ARM	Association Rule Mining
CB	Content-Based Filtering
CF	Collaborative Filtering
CFM	Category Feature Matrix
GSP	Generalized Sequence Pattern
KDD	Knowledge Discovery in Databases
LHS	Left-hand side
LSA	Latent Semantic Analysis
PFM	Product Feature Matrix
SPM	Sequential Pattern Mining
SVD	Singular Value Decomposition

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

Existing recommender systems mostly consider co-occurence of items when it comes to recommending an item to a user. This keeps recommendations shallow and recommendations become insufficient most of the time. For example, if there are 10 items recommended to a user with existing recommender systems, only 2 or 3 of them are related to the products that the user purchased before. This issue makes users to lose their interests with the e-commerce application and they manually search for related items. Besides, e-commerce application loses the chance of making users buy things, which means less income for a single shopping experience.

1.2 Proposed Methods and Models

Latent semantic analysis (LSA) is a way of extracting context and hidden insights from a group of texts. It is usually used when extracting general idea mentioned in a paragraph with sentences, in a document with paragraphs or in a library with documents. We believe that LSA can be useful for also transactional data, in which there are customers purchasing products. In this thesis work, by applying LSA, we tried to find out whether customers' transaction behavior can be extracted or not to develop a recommender system. In addition to extracting "customers' behavior", association rules are mined to construct a better recommender system. With extensions to these methods, more meaningful recommendations can be done.

1.3 Contributions and Novelties

LSA is usually done by applying a method called *singular value decomposition (SVD)*, which is explained later in the following chapters. This mathematical method shrinks the representation of the data, in our case, which is a customer-product purchase matrix. When SVD is applied to the data and collaborative filtering (CF) is done, results were not satisfying at all. Then *association support factors* are introduced to boost the performance of the model. These factors are multiplied with plain cosine similarities and helped the model while doing recommendations.

A group of methods is inspected for recommendation systems. Association rule mining (ARM) and sequential pattern mining (SPM) methods can be used for recommending products too. We believe that by finding relationships between products with ARM and SPM, we can design a better recommender system.

1.4 The Outline of the Thesis

Organization of the thesis as follows: Chapter 2 discusses recommendation methods and they are discussed with related works about them. Chapter 3 dives into technical, mathematical details of methods used in thesis work. In chapter 4, preprocessing operations on the data are explained. In chapter 5, application of LSA method called *Singular Value Decomposition* on transactional data is discussed. Chapter 6 is about association rule mining on transactional data, with the help of classical *Apriori* algorithm. Sequential pattern mining is done on transactional data with *PrefixSpan* algorithm and this is explained in chapter 7. Chapter 8 contains conclusions about the thesis.

CHAPTER 2

RELATED WORK

There are various types of recommender systems for e-commerce applications. Many alternative methods are proposed for recommending items to users in e-commerce applications. These methods are mainly grouped into 4 categories: recommendation with *collaborative filtering*, recommendation with *content-based filtering*, recommendation with *knowledge discovery in databases* (KDD) and recommendation with *latent semantic analysis* (LSA). This chapter mentions related works about these 4 categories of recommendation studies:

2.1 Recommendation with Collaborative Filtering

Collaborative filtering (CF) examines the similarity of users' preferences [1]. Recommendations by using CF can be done in two ways: **user-based** and **item-based**.

In user-based recommendations, users' item preferences are inspected. If two users have purchased common items or they have similar interests, then these users are called **neighbor users** of each other [1]. Items of closest N neighbor users are suggested to the user that the recommender system tries to make recommendations to.

In item-based recommendation, *cosine similarities* [2] between items are inspected [3]. In most of the recommendation systems, items are represented as a row of users' ratings. If cosine similarity of two items' representations is higher than cosine similarities with other items, then these two items can be recommended to users who buy one of them.

As an example of CF, Linden et. al. (Amazon.com) [4] did item-to-item CF for

their customers. Before recommending an item that is similar to another item, the item similarity matrix is constructed by calculating cosine similarities among items offline. Another work by Sarwar et. al. (GroupLens) [3] focused on item-based CF. They try to find a user's predicted rating for an item by using the weighted sum of user ratings and regression. Jacobi et. al. (Amazon.com) [5] did personal, user-to-user recommendations by processing electronic carts of users. Wu et. al. (LinkedIn.com) [6] did CF horizontally on user browsing history data in a large-scale recommender system. These are some of CF study examples.

Although CF is a widely-used recommendation method, there is a critical issue about it. The issue arises when a new user is joined to the system. Since the user hasn't purchased/rated any items yet, the recommendation system has no idea about the user, it can not find similar users to the user. No recommendation can be done because of that. This problem is called **cold-start problem** [7].

2.2 Recommendation with Content-Based Filtering

Content-based filtering (CB) constructs a profile of a user by inspecting properties of items that the user has purchased/rated and tries to recommend items according to the user's constructed profile [8]. As an example study, Pazzani [8] applied CB on restaurant descriptions and user ratings for recommending restaurants to users. He extracted features from restaurant descriptions and adjusted weights of these features by applying *Winnow* [9] algorithm. After these, he used adjusted weights for creating user profiles and making recommendations.

CB method seems useful in theory; however, some issues arise while dealing with the data in real life. Since CB tries to recommend items that are similar to preferences of the user itself, recommendations keep the user's interest region tight. Recommendations become more repetitive and shallower. This is called **over-specialization problem** [10]. Another issue arises when a new item is introduced to the system. Each new item needs to be defined as a set of tags (properties). The decision of defining new item with already existing tags or generating new tags causes trouble and makes working with CB method quite troublesome.

To overcome these problems in both CF and CB methods; Pazzani [8], Basilico et. al. [11], Claypool et. al. [12], Balabanovic et. al. [13] and many other people defined better recommender systems by combining CF and CB approaches in their studies. Pazzani [8] combined both methods by merging Pearson correlation [14] values of users (user similarity) and weights. Basilico et. al. [11] combined by defining different kernel and similarity functions. Claypool et. al. [12] used these methods separately and combined their predictions in a weighted manner. Balabanovic et. al. [13] generated user profiles with CB and used these profiles while doing recommendation CF.

With the combination, they generally achieved better results than plain CF or plain CB.

2.3 Recommendation with Knowledge Discovery in Databases

After the creation of Web, it has been seen that as the number of users, web sites and web companies were increased, the amount of data on the Web was also increased [15] [16]. Since doing inspection on the data itself is troublesome in databases, it is essential to eliminate unnecessary parts of the data and develop methods for finding useful information [17]. Also, data generally contain hidden knowledge within itself and this hidden knowledge can not be revealed by applying pure statistics.

Knowledge discovery in databases (KDD) means extracting useful and mostly hidden, unknown information from a collection of data [18]. Before revealing insights about the data, it is cleaned, processed and selected. There are many methods to discover insights within the data. These methods are quite different than the CF and CB methods.

2.3.1 Association Rule Mining

One common method for knowledge discovery is *association rule mining (ARM)* [19]. Agrawal et. al. proposed this method back in 1993 and the method examines the associativity among items by looking at the transaction history. If items are frequently

purchased together, they are likely to be associated. When there are items a and b with positive associativity and a user buys the item a, then item b is recommended to the user by a recommender system that does ARM. For instance, ARM asks this question: **If the user buys a gaming keyboard, is he/she likely to buy a gaming mouse?**

There are lots of studies that dive into ARM. For instance, Agrawal et. al. [20] presented *Apriori* and *Apriori-tid* algorithms for mining association rules. These algorithms contain candidate itemset generation and pruning them for finding frequent itemsets. The algorithms they presented can be considered as a milestone on ARM. Another work by Vaidya et. al. [21] presented an algorithm for applying ARM on distributed data partitions, without compromising the privacy of any partition. They found association rules with a protocol they presented in their study. As an extension to works of Agrawal et. al., Tao et. al. [22] viewed this problem from another perspective. They considered not only frequency of candidate itemsets, but also consider the "profit" gained by candidate itemsets. They ignored the **downward closure property** [19] of Agrawal et. al. and modified it with significant-weighted support calculation, where weights denote the profit (price) earned from an item/an itemset/a transaction. Liu et. al. [23] suggested using different a approach for selecting minimum support while eliminating candidate itemsets. They introduced **minimum item support**, which is the minimum of item support values in an itemset. This change allowed them to have higher support values for itemsets with frequent items and lower support values for itemsets with less frequent items. Just like the work of Tao et. al., they intended to change downward closure property and kept potentially useful itemsets that are not frequent itemsets (which are eliminated by the Apriori algorithm in earlier iterations) for future iterations.

2.3.2 Sequential Pattern Mining

Another method for knowledge discovery is *sequential pattern mining (SPM)* [24]. In ARM, the main focus is on "which items are purchased together" or "which items are related to each other". However, in SPM, the main focus is on "which item will be purchased after one specific item is purchased". SPM is done not only for itemset mining but also used for string mining in bioinformatics [25].

SPM is interested in transaction purchase time, besides association between items. A recommendation is done in a time-independent perspective with ARM. An item b recommended to a person that purchased item a at any time. On the other hand, SPM takes care of basket sequences. It tries to spot a sequence of items within various baskets and recommendation is done according to how the sequence proceeds. Baskets in a sequence don't need to be consecutive while discovering sequential patterns. For instance, SPM asks this question: **Will the user purchase a gaming headset in 3 months when he/she buys a gaming keyboard and a mouse?**

Just like in ARM, Agrawal et. al. [24] are pioneers of SPM studies. They proposed the algorithms called *AprioriAll*, *AprioriSome* which are similar to Apriori algorithm for ARM. Another work is done by Ayres et. al. [26] which introduced an algorithm called *SPAM* and represented sequences with bitmaps. SPAM algorithm is based on depth-first search traversal [27] and does pruning on constructed itemset tree. Pei et. al. [28] introduced a new algorithm called *PrefixSpan*. They addressed few difficulties of Apriori-like solutions, such as generation of large number of candidate sequences, multiple scans of databases, mining long sequential patterns. To solve these problems, instead of looking at all possible occurrences of frequent sequences, they do projection based on frequent prefix subsequences. Algorithm details are explained in chapter **Technical Background**.

2.4 Recommendation with Latent Semantic Analysis

Latent semantic analysis (LSA) is a method for extracting knowledge and contextual information from text corpora [29]. Extracted core knowledge is simply the projection of words and passages in the text to a baseline called **semantic space** [29]. This core knowledge is a representation of how the human brain understands words and passages. With LSA, relationships between words and passages are found within the text. The result of LSA is an approximation of cognitive activities that happen when a human reads words and passages. LSA is usually done with a method called *singular value decomposition* (SVD) [30].

LSA can be applied to data from different origins. For example, if data contains

sentences within a paragraph, LSA can reveal the context of the paragraph. When the data is about users, ratings and movies, doing LSA can reveal the relationship between users and movie genres. If LSA is done to transactional data, it can reveal customer behavior about shopping and the relationship between shopping behavior-users and shopping behavior-products. LSA provides a good approximation of the data and a great method for finding out insights within the data. Unlike neural networks and deep architectures, LSA is simpler since it is only a mathematical matrix decomposition method. LSA doesn't need any prior knowledge to extract context from the data, only the existence of groups of textual information is enough.

As a study, Deerwester et. al. [31] used LSA as an indexing method in their work. They stated that term-based information retrieval systems are not enough to understand semantic information within documents. They applied SVD method to the data and saw that LSA handles synonym words very well and it partially handles polysemous words. Hoffman [32] introduced a new model called *probabilistic LSA (PLSA)*. The difference between PLSA and LSA is the objective function utilized to determine the optimal decomposition/approximation. LSA uses *Frobenius norm* [33] whereas PLSA uses the likelihood function of multinominal sampling. This change helped to find out the optimal number of latent space dimensions, which is harder with SVD, since it depends on heuristics. Landauer et. al. [34] tested SVD with data containing 60000 words and 30000 text passages and concluded that optimizing dimensions in SVD can improve information extraction performance. Also, they found that SVD is a great tool for extracting relationships between words when syntax and vocabulary of the language are unknown. Behrens et. al. [35] used LSA for recommending products to the users. After applying SVD on the data containing abstracts for the products, they looked at the cosine similarities for finding similar abstracts of products. When similar abstracts are found, then these products are recommended to users.

After recommendation methods are presented, the next chapter explains the technical details behind these methods to understand them better.

CHAPTER 3

TECHNICAL BACKGROUND

Technical details about models, paradigms, algorithms that are used in this thesis work are explained with formulas and examples in this section. This thesis work is based on the technical details below.

3.1 Latent Semantic Analysis with Singular Value Decomposition

Singular value decomposition (SVD) [30] is a factorization method for matrices [36]. It decomposes a matrix into 3 smaller-sized matrices. Multiplying these 3 matrices will result in regeneration of the original matrix OM:

$$OM = U * S * V^T \tag{3.1}$$

Let OM be the data matrix containing data about m documents with possibly n terms. The aim is to reveal **hidden concepts** about the data [37]. These hidden concepts give insights about our data and they can not be understood directly in the first place. When SVD is applied to OM with size (m, n):

- Matrix U will be in size (m, r). U is called **left singular vectors**, which is a relationship matrix between documents and concepts.
- Matrix S will be in size (r, r). S is a special matrix, only values on the diagonal are non-zero. Values on the diagonal are called **singular values**. Singular values give information about strength of a concept.

- Matrix V will be in size (n, r). V is called **right singular vectors**, which is a relationship matrix between terms and concepts.

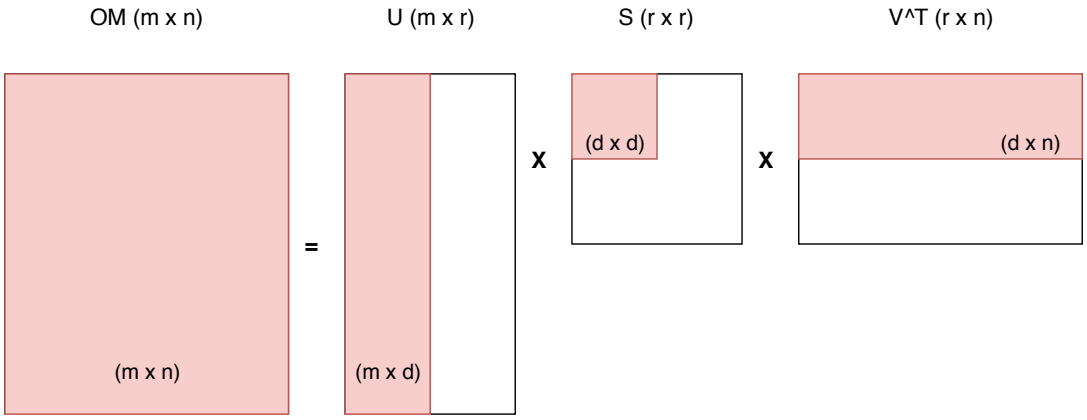


Figure 3.1: Visual explanation of SVD

SVD can be considered as dimension reduction [1] and compression method [38], since r is much less than m and n. Choosing the number of singular values is the main challenge for SVD [34]. It is a trade-off between size, performance and information:

- If r (number of singular values) is small, OM is shrunk more, resulting in a smaller-sized representation of OM. If representation is smaller-sized, processing left and right singular values will be faster. However, more shrinking means losing more information. The regeneration matrix will be quite different than OM.
- If r is large, OM is shrunk less, resulting in a large-sized representation of OM. Information loss is less than the previous case, the regeneration matrix will be quite similar to OM. However, processing left and right singular values will take more execution time.

In this thesis work, the data matrix will be a customer-product matrix or a customer-category matrix. To do customer-to-customer recommendation, rows of matrix U will be examined. To do product-to-product recommendation, rows of matrix VS will be examined. The similarity of two rows of U means customer similarity, the similarity of two rows of VS means product similarity (or category similarity) with respect to relationships with hidden concepts.

3.2 Association Rule Mining with Apriori Algorithm

Most commonly used algorithm for ARM is *Apriori* algorithm [19]. Firstly, items that are frequently purchased are found, by looking at how many baskets in the data contain those items. If the ratio between purchase count and the number of baskets is above *support threshold*, then these items form a frequent itemset. For the next iteration, frequent itemsets are unified to have larger frequent itemsets. This procedure is done until no new frequent itemset is found.

$$Support(A) = \frac{\text{number of times itemset A is purhcased}}{\text{number of baskets}} \tag{3.2}$$

After frequent itemsets are found, **association rules** are constructed. Assocation rules are found by calculating *confidence* and *lift* factors of each of sub-itemset pairs:

- *Confidence* factor is the ratio between the number of times that items are purchased together and the number of times one of the items is purchased. A higher confidence factor means higher associativity.

  Frequent itemset: [A,B]

  Tested rule: A $\rightarrow$ B (If itemset A is purchased, itemset B is likely to be purchased)

$$Confidence(A, B) = \frac{Support(A, B)}{Support(A)} \tag{3.3}$$

- *Lift* factor is the ratio between support value of two items and the multiplied support values of each item.

  Tested rule: A $\rightarrow$ B (If itemset A is purchased, itemset B is likely to be purchased)

$$Lift(A, B) = \frac{Support(A, B)}{Support(A) * Support(B)} = \frac{Confidence(A, B)}{Support(B)} \tag{3.4}$$

  - If $lift > 1$, items A and B are **dependent** to each other. When item A is purchased, item B will probably be purchased.

- If $lift = 1$, items A and B are **independent** from each other. It is just a coincidence that these items are frequently purchased together.
- If $lift < 1$, items A and B are **contrary** to each other. If item A is purchased, item B is unlikely to be purchased.

3.3 Sequential Pattern Mining with PrefixSpan Algorithm

PrefixSpan algorithm [28] is one of the common SPM algorithms. It works in divide & conquer logic. Sequential patterns are found by declaring prefixes and their projected databases. Consider following sequences of baskets:

Sequence 1: [a (abc) (ac) d (cf)] (5 baskets)
Sequence 2: [(ad) c (bc) (ae)] (4 baskets)
Sequence 3: [(ef) (ab) (df) c b] (5 baskets)
Sequence 4: [e g (af) c b c] (6 baskets)

Firstly, length-1 sequential patterns are found. After that, projected databases are found according to prefixes in length-1 sequential patterns. If the prefix is <a>, we have projected database as [(abc) (ac) d (cf)], [(_d) c (bc) (ae)], [(_b) (df) c b] and [(_f) c b c]. This is done for all length-1 sequential patterns (<a>, <b>, . . . , <f>).

Prefix: <a>
Sequence 1: [(abc) (ac) d (cf)]
Sequence 2: [(_d) c (bc) (ae)]
Sequence 3: [(_b) (df) c b]
Sequence 4: [(_f) c b c]

After this step, mining is done for each projected database. Prefix is extended to one of <aa>, <ab>, . . . , <af> and with these length-2 sequential patterns, newly projected databases are found. For example, projected database for the prefix is <aa> is [(_bc) (ac) d (cf)].

Prefix: <aa>
Sequence 1: [(_bc) (ac) d (cf)]

This process is repeatedly done with longer sequential patterns until the projected database consists of only 1 sequence. If such a case happens, the execution of the current branch is finished.

When all possible longest prefixes are found, the algorithm finishes.

CHAPTER 4

DATA STATISTICS & PREPROCESSING

4.1 General Information

Data consists of 200000 transactions made from e-commerce websites such as **Akakçe**, **EpttAvm**, **GittiGidiyor**, **Hepsiburada**, **SanalPazar**, **Trendyol**. These websites are the most popular e-commerce websites visited in Turkey. Besides this data, there is another data which contains 2.5 million transactions and results will be shown with some of the recommendation methods discussed in this thesis work.

Data is a single table containing transaction information. Any transaction information in the data contains:

- platform name
- store name
- order information
- customer information
- delivery and invoice address information
- product name, category, price, quantity information

The time period for transactions is from 7 Nov 2014 to 29 Aug 2018. As a side note, it is seen that the number of transactions increase in weeks of important days such as Valentine's Day, Mother's Day, Black Friday, etc.

Data A and data B statistics are like the following. Below statistics are obtained after a considerable amount of preprocessing work:

Data with 200000 transactions:

- 6 distinct platforms (websites mentioned above)
- 560 distinct stores
- 142501 distinct customers
- 73482 distinct products
- 16048 distinct product categories

Data with 2.5 million transactions:

- 10 distinct platforms
- 825 distinct stores
- 1457436 distinct customers
- 335882 distinct products
- 39419 distinct product categories

By looking at the pure statistics, it can be seen that this amount of data requires too much time to process it. Also, if the amount of transactions increases, the number of distinct assets such as customers, products, product categories increase rapidly.

Each transaction of the data examined in this thesis work contains only one product. For instance, if a customer bought 3 products in a single shopping, it is submitted as 3 different transactions to the database. When transactions of the data is grouped by their transaction time, it is seen that most of the shopping experiences contain only one product:

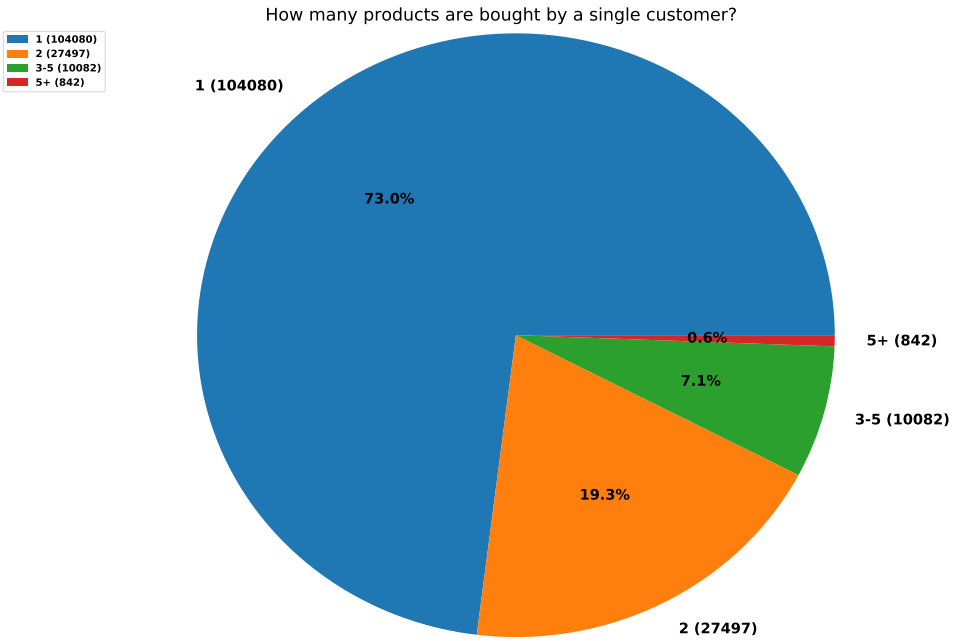


Figure 4.1: Shopping-Product count histogram (pie) - 200K transactions

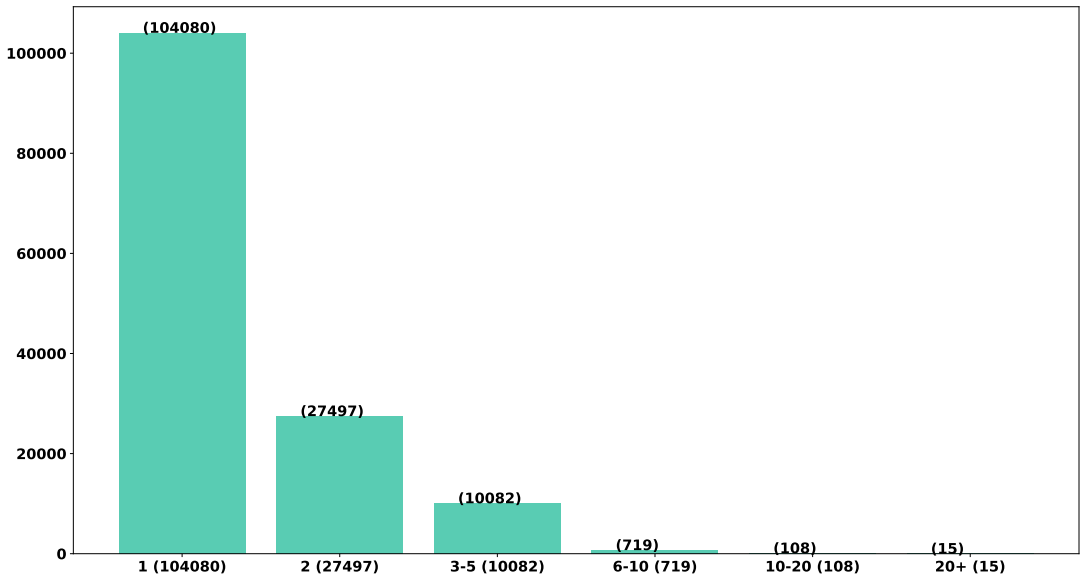


Figure 4.2: Shopping-Product count histogram (bar) - 200K transactions

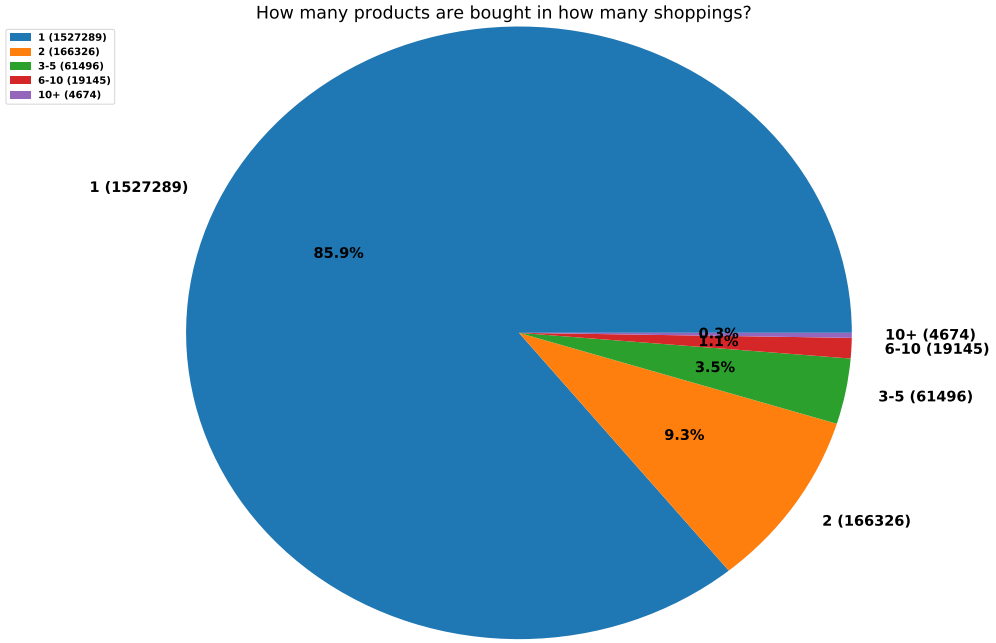


Figure 4.3: Shopping-Product count histogram (pie) - 2.5M transactions

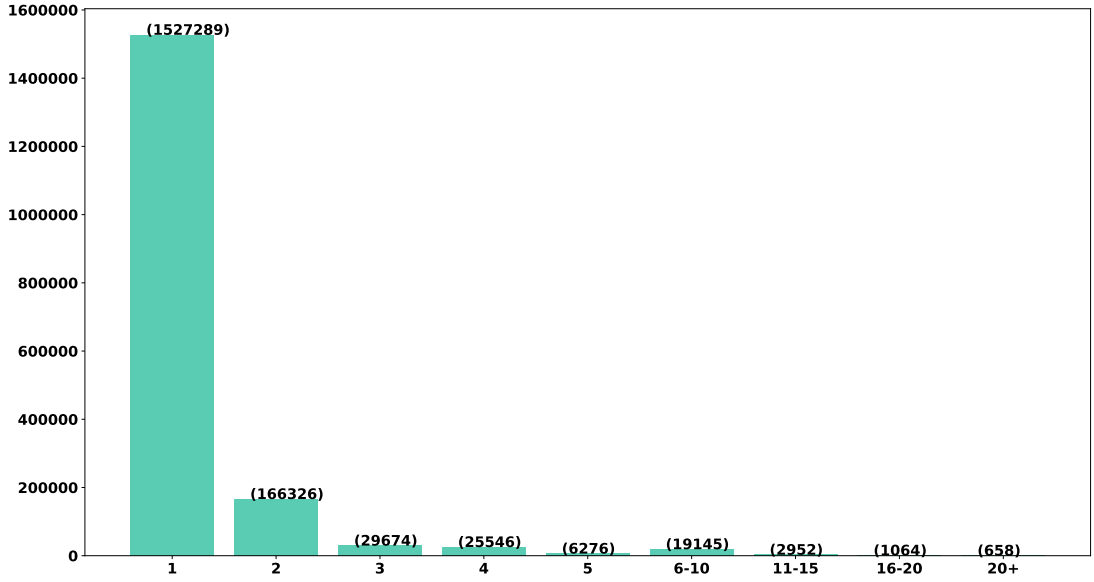


Figure 4.4: Shopping-Product count histogram (bar) - 2.5M transactions

Since 1-product shoppings are the majority of the transactions, they can not be used while doing recommendations, mining association rules or finding frequent product sequences. These transactions are excluded most of the time.

Data is processed for anonymous usage. No personal information is compromised. Throughout this work, anonymized data is used and processed.

4.2 Data Preprocessing Steps

Preprocessing took long since transactions are from different platforms with different categorization and there are non-processable characters in columns of a transaction. Below operations are done as a part of preprocessing work:

- For all columns of the main table, all letters are converted to lowercase.
- For all columns of the main table, all Turkish letters are converted to English letters:
  - Letter 'ı' converted to letter 'i'
  - Letter 'ö' converted to letter 'o'
  - Letter 'ü' converted to letter 'u'
  - Letter 'ç' converted to letter 'c'
  - Letter 'ğ' converted to letter 'g'
  - Letter 'ş' converted to letter 's'
- For all columns of the main table, all words are trimmed. Multiple spaces are removed. Empty columns are converted to *NULL*. Irrelevant special characters are removed.
- For "Product Category" column of the main table, product category was separated with various characters for hierarchical categorization such as:
  - ',' (comma)
  - '-' (dash)
  - '<<' (double lower than signs)

- '/' (slash)
- '//' (double slashes)

etc. To have a common standard structure for product categories, all special characters between category sections are converted to **'|' (pipe)** character.

- A distinct customer is defined as a group of **platform name, customer nickname, customer full name, customer city, customer phone** columns. There are also customer delivery address and customer invoice address columns. However, since multiple customers can choose the same address for delivery and invoice address, these columns are not used while identifying distinct customers.

- After identifying distinct customers, it is seen that some of the customers have the same information except phone numbers. This may be due to the fact that some of the customers are corporate identities. Therefore, these customers with the same information except phone number are merged and considered as one single customer.

- After identifying distinct products, it is seen that some of the products have different categories. This is due to the fact that the category of the product is defined variously in different platforms/stores. Therefore, to overcome this issue, the number of transactions are considered. **If product p_1 with category cat_1 is more popular (has purchased in more transactions) than same product p_1 with category cat_2, then category cat_1 is accepted as product p_1's category for all transactions.** This method reduced the number of distinct product categories, naturally.

- There are 16048 distinct product categories (39419 categories for large data) found in the data. Categories are defined as branches of category trees from different platforms. These category branches are from different category trees. There may be multiple branches from different platforms which are similar to each other, but considered as different branches (categories):

```
mc  => main category
sbc => subcategory

category a from platform a => mc1 | sbc1 | sbc2 | sbc3
```

```
category b from platform b => sbc3 | sbc2 | sbc1 | mc1
category c from platform c => mc1 | sbc1 | sbc3
category d from platform d => sbc2 | mc1
...
```

As seen in above example, categories a, b, c, d are similar to each other when considered in subcategory perspective. Also, categories a and c can be considered as the same category. **Different platforms have different ways of constructing subcategory hierarchy on their data; unfortunately, there is no common standard for categorizing the products.** Grouping similar categories is another big problem to solve. For simplicity, grouping categories is skipped and category branches are used as-is. **Since there are 16048 distinct category branches as a whole, distinct category set cardinality is considered to be 16048 in this work.**

- A distinct customer delivery information is defined as a group of **customer delivery title, customer delivery address, customer delivery postal code, customer delivery district, customer delivery city, customer delivery phone**.
- A distinct customer invoice information is defined as a group of **customer invoice title, customer invoice address, customer invoice postal code, customer invoice district, customer invoice city, customer invoice phone, customer invoice tax authority, customer invoice tax number**.
- For simplifying the main table; customers, products, product categories, delivery information, invoice information, stores, platforms, order statuses are defined with IDs by the above rules. A distinct id number is given for each of these columns. The number of columns in the main table is reduced and the main table becomes processable by the software since no textual information is left on the processed table.

CHAPTER 5

SINGULAR VALUE DECOMPOSITION (SVD) ON THE DATA

5.1 Introduction

Collaborative filtering (CF) method is a good way of providing recommendations for customers, where there is a *"customer purchased a product"* relationship. [3]
CF provides recommendations for customers by finding customers with similar product choices. When a customer c_2 with similar choices to customer c_1 is found, products purchased by customer c_2 but not purchased by customer c_1 is recommended to customer c_1.
CF is also applied for recommendations to customers by finding similar products to products that they purchased. If product p_2 is similar to product p_1 and the customer has purchased product p_1, then product p_2 is recommended to the customer.
It is important to find out relationships among customers and among products before CF is done. At this point, *Singular Value Decomposition (SVD)* [30] is applied for revealing such relationships. With the help of SVD, the aim is to discover **"concepts"** within the data and to find out customer-concept, product-concept relationships.

5.2 Difficulties About Applying SVD on Data

5.2.1 Data Size

As stated in chapter **Data Statistics & Preprocessing**, there are 142.501 distinct customers and 73.482 distinct products in the data with 200K transactions. Generating a feature matrix from this data requires too much memory and it takes too much time

to process it using SVD:

$$142501 * 73482 =\sim 10.5 \text{ billion integers} =\sim 41 \text{ billion bytes} =\sim 39 \text{ GB}$$

Since memory requirement is high and data is too sparse, filtering customers and products becomes an essential task to do [40]. Filtering is applied according to the following procedure:

- *prod_trx_threshold* and *cust_trx_threshold* are defined as thresholds.
- Products that are purchased less than *prod_trx_threshold* times are filtered out or ignored. If a product is purchased more than *prod_trx_threshold* times, it has a significant effect on transactions made by all customers. Such products are called *target products*.
- Customers who purchased less than *cust_trx_threshold* distinct products among target products are also filtered out or discarded. Remaining customers are called *target customers*. If a customer has not purchased any target product or has purchased less than *cust_trx_threshold* distinct target products, transactions of this customer have no significant effect on transactions.

Choosing *prod_trx_threshold* and *cust_trx_threshold* values actually depends on the data. In **Figure 5.1**, **red and blue** lines represent target customer set size when *cust_trx_threshold* is 2. **Orange and light-blue** lines represent target customer set size when *cust_trx_threshold* is 3. *prod_trx_threshold* range is between 4-10.
After histograms with various thresholds are examined, *cust_trx_threshold* is selected as 2 and *prod_trx_threshold* is selected as 5 for the data with 200K transactions. When a method that similar to the *elbow method* explained in [41] is applied, these values seem logical to use them.
In **Figure 5.2**, the histograms retrieved from the larger data with 2.5M transactions are shown. *Prod_trx_threshold* value is chosen as 50 and *cust_trx_threshold* is chosen as 10. When data gets larger, boundaries should get larger too, in order to give faster results while doing experiments. Experiment results with this data is also shown among results.

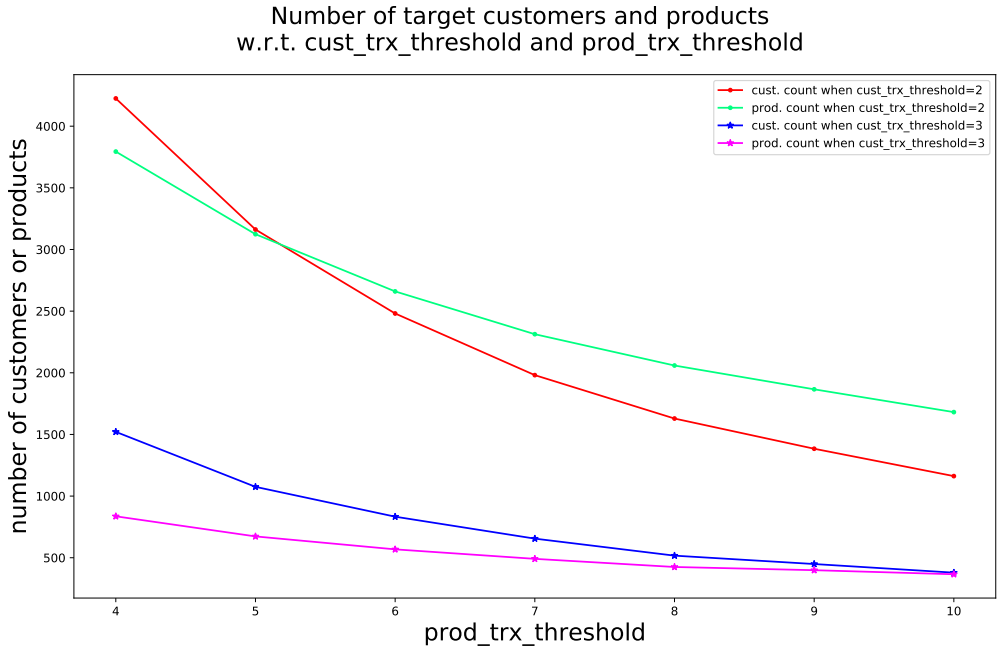


Figure 5.1: Target customer-target product set sizes (200K transactions)

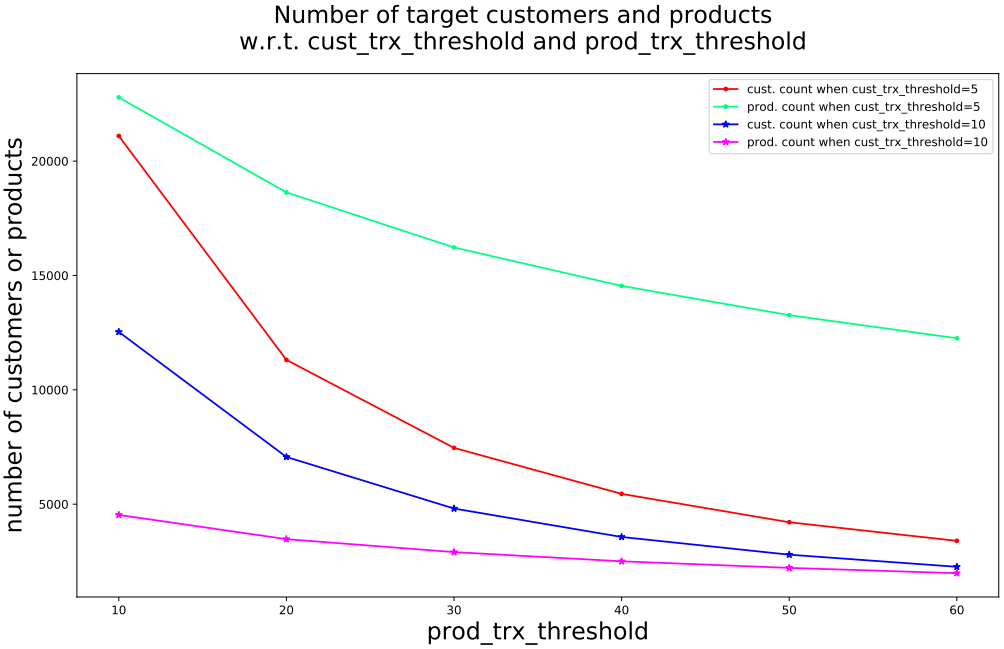


Figure 5.2: Target customer-target product set sizes (2.5M transactions)

5.3 Id Virtualization & Translation

The actual number of customers and products is much bigger than the number of target customers and products. If we constructed the feature matrix from all customers and products, we could use the distinct ids as the indices of the rows and columns of the feature matrix as-is. Since we shrunk the number of customers and products in the feature matrix as target customers and target products, we need to define virtual ids for target customers and target products, to manage row and column operations in the feature matrix.

It will be better understood with this example: Assume that we have a customer with 1457832 as customer id and we have a product with 54368 as product id. Let us also assume that customer 1457832 has purchased product 54368. If we didn't shrink the number of customers and products and we wanted to read/change the value in the feature matrix, we needed to do:

```
FM[1457832][54368] = 7
```

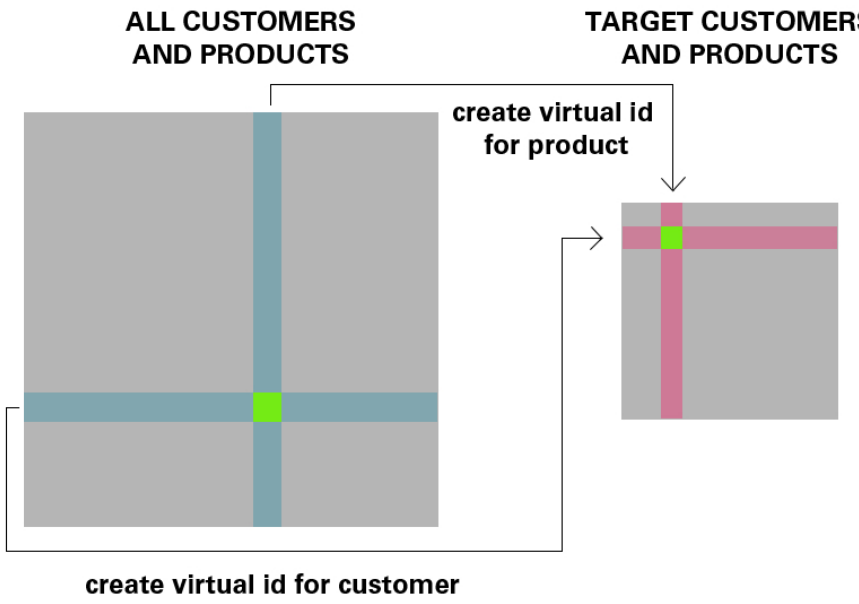
In contrast, we have target customers and target products concepts. We can not use 1457832 and 54368 as indices; therefore, we need to define virtual ids for customer 1457832 and product 54368 to use virtual ids as indices. After this, we can use these virtual ids to read/modify values in the feature matrix:

```
virtual_id_customer = get_virtual_id_of_customer(1457832)
virtual_id_product = get_virtual_id_of_product(54368)
FM[virtual_id_customer][virtual_id_product] = 7
```

Id virtualization and translation steps are done with the help of real id-virtual id mappings. These mappings are created before the model applies SVD to the data. After these mappings are created, similar customers or similar products are found and these sets contain virtual ids of customers and products. Later on, these virtual ids are translated back to the real customer and real product ids:

```
virtual_id_customer = get_virtual_id_of_customer(1457832)
similar_customers = get_similar_customer_to_customer(
                                    virtual_id_customer)
real_ids_similar_customers = get_real_ids_of_customers(
                                    similar_customers)
```

Whole procedure can be also seen in **Figure 5.3** and **Figure 5.4**. Figure 5.3 shows the mapping between real customer/product ids and virtual customer/product ids. Figure 5.4 shows the translation between virtual ids and real ids, when the model looks for similar customers to a customer in customer-to-customer recommendation.




```
map_real_id_to_virtual_id_customer[1457832] = 26
map_real_id_to_virtual_id_customer[26] = 1457832

map_real_id_to_virtual_id_product[54368] = 12
map_real_id_to_virtual_id_product[12] = 54368

feat.matrix[26][12] = 7
```

Figure 5.3: Customer and product id virtualization

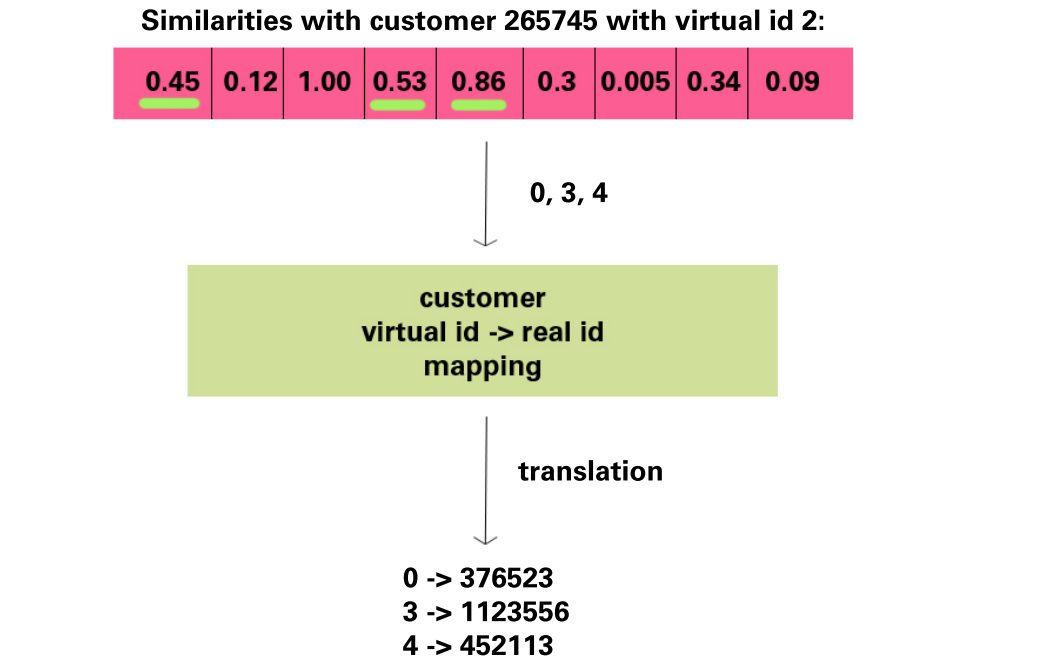


Figure 5.4: Customer and product id translation

5.4 Constructing Feature Matrices & Normalization

Having above background information in mind, customer-product purchase and customer-category purchase matrices are constructed. After that, all rows of customer-product matrix are normalized over the maximum number of transactions made by customer for any product:

```
feat. matrix row = [1, 0, 2, 0, 1, 3, 0, 2]
max. number of purchases = 3
operation = divide by 3
after norm. = [0.33, 0, 0.66, 0, 0.33, 1.0, 0, 0.66]
```

Similar method is applied on customer-category matrix. After normalization process, resulting matrices are called *"product feature matrix" (PFM)* and *"category feature matrix" (CFM)*.

5.5 Application of SVD

Customer-concept, product-concept relationships are not seen obviously by looking at the raw data. Therefore, there should be a way to find out these relationships [1]. When SVD is applied to PFM and CFM, relationships are revealed. The PFM is disassembled by SVD and as a result, 3 different matrices are found:

$$PFM = U * S * V^T \tag{5.1}$$

Where U is *customer-concept matrix*, S is a square matrix with *singular values* only in diagonal and V is *product-concept matrix*. Matrix S (with singular values) represents the strength of these concepts.

After applying SVD, CF is done for product-to-product recommendation by using matrix VS. To find similar products to a particular product, *cosine similarity* of two rows of matrix VS is found. If value is high, similarity is high too. [1] [2] [3]

On the other hand, CF is done for customer-to-customer recommendation in longer way that is explained in [39]:

- (a) - Queried customer row is multiplied by matrix V
- (b) - Result from (a) is multiplied by inverse of matrix S
- (c) - Cosine similarities are considered between rows of matrix U and result from (b).

Besides disassembling of PFM, the CFM is also disassembled by SVD and as a result, 3 different matrices are found:

$$CFM = U_2 * S_2 * V_2^T \tag{5.2}$$

Where U_2 is *customer-concept* matrix, S_2 is a square matrix with singular values only in diagonal and V_2 is *category-concept matrix*. Matrix S_2 (singular values) represents

the strength of these concepts related with categories.

After applying SVD, to find similar categories to a particular category, matrix V_2S_2 is used. Cosine similarity of two rows of matrix V_2S_2 is found. If value is high, similarity is high too.

5.5.1 Choosing Number of Singular Values

When *cust_trx_threshold* is selected as 2 and *prod_trx_threshold* is selected as 5 (for data with 200K transactions), there are 3124 target customers and 3162 target products. SVD is applied to product feature and category feature matrices. To decide optimal the number of singular values, various tests are done. Product-to-product recommendation results according to various number of singular values can be found in **Table 5.1**:

Table 5.1: Singular values vs Accuracy (product-to-product recommendation)

Sing. values	Accuracy	SVD Fit Time (sec)
5	0,4867	17,8165
10	0,5316	18,2998
50	**0,6701**	**17,7358**
100	**0,7014**	**18,3158**
250	0,7634	18,3801
500	0,7774	18,1763
750	0,7746	20,3402
1000	0,7623	18,9738
1500	0,5358	24,3609

Customer-to-customer recommendation results according to various number of singular values can be found in **Table 5.2**:

Table 5.2: Singular values vs Accuracy (customer-to-customer recommendation)

Sing. values	Accuracy	SVD Fit Time (sec)
10	0,0995	17,6409
50	0,2291	17,3380
100	**0,3117**	**18,7882**
250	0,4055	18,9985
500	0,4135	18,0745
1000	0,3730	20,9154

Results in both tables are the results of SVD single-fold experiments. That means the data is fed to the model as one single block. From the above results, **number of singular values is chosen to be 50 or 100**. Criteria for these decisions are:

- **Having a small number of singular values to overcome complexity.** Since aim of SVD to find a representation of the original matrix and also shrink it to a smaller size, there should be fewer number of singular values.
- **Having shorter fit time for SVD.** Running time for applying SVD to feature matrices should be decreased to wait less for the results.
- **Having high evaluation accuracy.** More accuracy means more reliability.

5.6 Recommendation Model After SVD

5.6.1 Product-to-Product Recommendation

5.6.1.1 Recommendation

The product-to-product recommendation aims to recommend products that are similar to the products of a customer. As explained in **Application of SVD** section, SVD decomposes PFM and CFM to product of 3 different matrices. Before making recommendations to customer c_1, following steps are done:

1. SVD applied to PFM and U, S, V matrices are obtained.
2. SVD applied to CFM and U_2, S_2, V_2 matrices are obtained.
3. Matrix V represents the relationship between products and concepts. To find a similarity between two products, row pairs of matrix VS are used.
4. Matrix V_2 represents relationship between categories and concepts. To find similarity between two categories, row pairs of matrix V_2S_2 are used.
5. By calculating *cosine-similarity* [2] of pairs of rows of matrix VS, **"product similarity matrix"** is constructed. Now, one row of product similarity matrix

represents similarities with other products. Similarities range from 0 to 1. The similarity value is 1 for the product itself.

6. By calculating *cosine-similarity* of pairs of rows of matrix V_2S_2, **"category similarity matrix"** is constructed. Now, one row of category similarity matrix represents similarities with other categories. Similarities range from 0 to 1. The similarity value is 1 for the category itself.

7. Let p_1 is the product examined. For finding similar products to p_1, similarity values from the corresponding row of product similarity matrix are sorted in descending order.

8. Let cat_1 is the category of the product p_1. For finding similar categories to cat_1, similarity values from the corresponding row of category similarity matrix are sorted in descending order.

9. After cosine similarities are found with other products or categories, **"association support factors"** are considered for each product pair. Association support factor is higher when products are bought together in a large number of transactions made by different customers, lower otherwise. The calculation for association support factors is done in the following way:

```
C = Num. of distinct customers that purchased both products
TRX_C = Total num. of transactions made by customers in set C.
TRX_ALL = Total num. of transactions in target scope
(target customers, target products)
INT = Num. of distinct customers who purchased both products
UN = Num. of distinct customers who purchased at least one of
the products

Assoc. Sup. Factor(p1, p2) = C * (TRX_C/TRX_ALL) * (INT/UN)
```

Using only cosine similarities had one big disadvantage: the similarity of two products is 1 when two products are purchased only in one transaction or these products are purchased by only one customer. Since this is the case for some of the target products, association support factors are introduced and multiplied with cosine similarities. **With this way, the similarity of products has**

more consistency, since "behavior" of various customers are included.

After the calculation of association support factors, these factors are multiplied with product cosine similarities found before. Multiplication results are sorted by descending order again, to eliminate product pairs that are purchased only in one transaction or are purchased by only one customer.

10. Now it is time for filtering less similar products out. A threshold that is called **"significance"** is applied to the results. For instance, if *significance* is 0.2, products that have similarity to product p_1 higher or above 20% are filtered and below 20% are discarded.

11. In addition to *significance* threshold, **"lowerbound"** and **"upperbound"** thresholds are applied. *Lowerbound* is used for recommending at least l similar products and *upperbound* is used for recommending at most u similar products at the end. Number of similar products that are recommended changes according to boundaries. Whole procedure of finding similar products to product p_1 is summarized below:

    $l \leftarrow$ lowerbound value

    $u \leftarrow$ upperbound value

    $\rightarrow$ Find corresponding row of p_1 in product similarity matrix.

    $\rightarrow$ Find association support factors for p_1 and each other product.

    $\rightarrow$ Multiply cosine similarities and association support factors. Sort results in descending order.

    $\rightarrow$ Check out how many products are above the 'significance' threshold considering similarity results found before.

    $\rightarrow$ Find most similar product category to product p_1's category. Get <*l*> products from that category.

    $n \leftarrow$ num. of sim. products that have sim. above 'significance'

    if $n \leq l$ **then**

    $\rightarrow$ Select products with top l similarity values

    else

$\rightarrow$ Select products with top n similarity values

end if

if $n > u$ **then**

$\rightarrow$ Select products with top u similarity values

end if

if $include\_categories = True$ **then**

while $n \leq u$ **do**

$\rightarrow$ Select one product from the most similar category and increment n

end while

end if

5.6.1.2 Evaluation - Single Fold

In this case, customers are selected among target customers and their transaction information is also included in the model while doing SVD. For evaluation of success in the product-to-product recommendation, the following method is applied to all customers. Let's assume that we want to make recommendations to customer c_1 with purchased products p_1, p_2, p_3:

1. **"Alternative customers"** from customer c_1 is generated. These customers are basically clones of the customer c_1, except one of the products is removed from c_1's purchased product list. To understand better, examine below explanation:

   $alter\_cust\_1 \leftarrow [p2, p3]$
   $alter\_cust\_2 \leftarrow [p1, p3]$
   $alter\_cust\_3 \leftarrow [p1, p2]$

2. After products are recommended for each product of each alternative customer, if recommended products contain the missing product, then it can be thought that recommendation is successful. Otherwise, the recommendation fails. When all alternative customers are examined, total accuracy is calculated. Below example explains this more:

   - $alter\_cust\_1 = [p2, p3]$
     $sim\_products\_to\_p2 \leftarrow [p4, p5, p6, \dots]$
     $sim\_products\_to\_p3 \leftarrow [p4, p1, p7, \dots]$
     $union(sim\_products\_to\_p2, sim\_products\_to\_p3) = [p1, p4, p5, p6, p7, \dots]$

     p_1 is in union set, recommendation is **SUCCESSFUL**.

   - $alter\_cust\_2 = [p1, p3]$
     $sim\_products\_to\_p1 \leftarrow [p3, p4, p5, \dots]$
     $sim\_products\_to\_p3 \leftarrow [p4, p1, p7, \dots]$
     $union(sim\_products\_to\_p1, sim\_products\_to\_p3) = [p1, p3, p4, p5, p7, \dots]$

     p_2 is **NOT** in union set, recommendation is **FAILED**.

   - $alter\_cust\_3 = [p1, p2]$

$sim\_products\_to\_p1 \leftarrow [p3, p4, p5, \dots]$

$sim\_products\_to\_p2 \leftarrow [p4, p5, p6, \dots]$

$union(sim\_products\_to\_p1, sim\_products\_to\_p2) = [p3, p4, p5, p6]$

p_3 is in union set, recommendation is **SUCCESSFUL**.

- **Recommendation accuracy = success / (success + failed) = 2 out of 3 are guessed = 0.66 (66%)**

3. After this is done for all customers in data, average of size of recommended product sets and average of accuracies are calculated. These two values are the results of the product-to-product recommendation. Results can be examined in **SVD Recommendation Results** section.

5.6.1.3 Evaluation - 10-Fold Cross-Validation

In this case, customers are selected among target customers and their transaction information stayed outside of the model while doing SVD. After results are obtained and SVD is fed with whole data, 10-fold cross-validation is done. Partitioning operation is done as explained below:

- **Transactions are split customer by customer and transactions that belong to 10% of the all target customers are constituting each fold.** Since there are ∼3000 target customers when *prod_trx_threshold* is 5 and *cust_trx_threshold* is 2, each fold of the data contains transactions that belong to ∼300 customers. Please keep in mind that these cardinalities are found from the data with 200K transactions.

- For each fold of 10 folds, transactions that belong to customers in the other 9 folds are processed in SVD. After product similarity and category similarity matrices are constructed, transactions of the current fold are used in the evaluation.

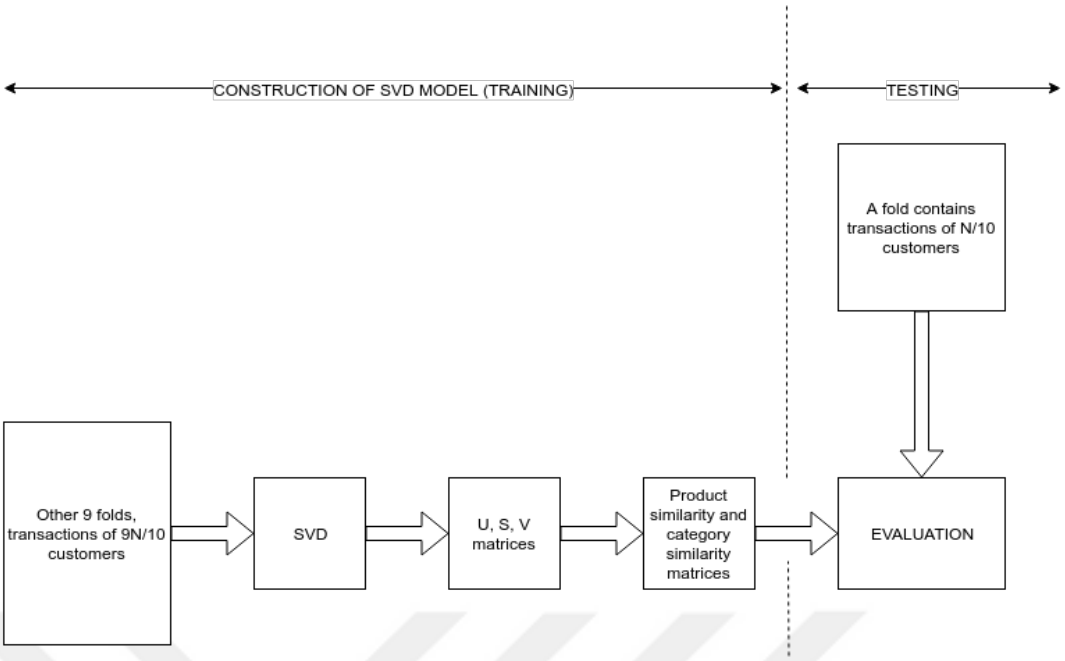


Figure 5.5: Cross-validation of product-to-product recommendation with 10 folds

- Evaluation method in **Figure 5.5** is actually same as the method explained in **Evaluation - Single Fold** section. This time, instead of all customers, evaluation is done for only customers in the current fold, and transaction information of customers of the current fold is **excluded** from the model.

- While obtaining results, this issue has arisen: **There are products that are purchased by customers in current fold, but not purchased by customers in other 9 folds.** These products are not among products that the model knows and these products have no representation in product-concept matrix V. **Therefore, it is meaningful to discard recommendation for these products.**

  At this point, *existing product count* threshold is applied. If number of products of the current customer is less than *existing product count* threshold, then no recommendation is done to the customer. **Figure 5.6** and **Figure 5.7** show the relationship between *existing product count* threshold, accuracy and number of customers that have received a recommendation for one of the partitions. Detailed results are shown **Table 5.11** for 50 singular values.

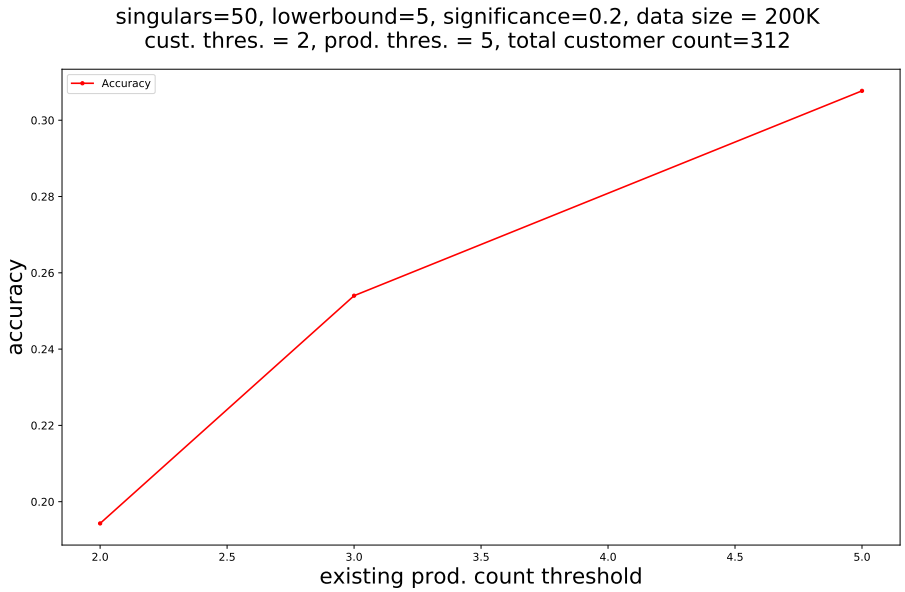


Figure 5.6: Existing Product Count vs. Accuracy

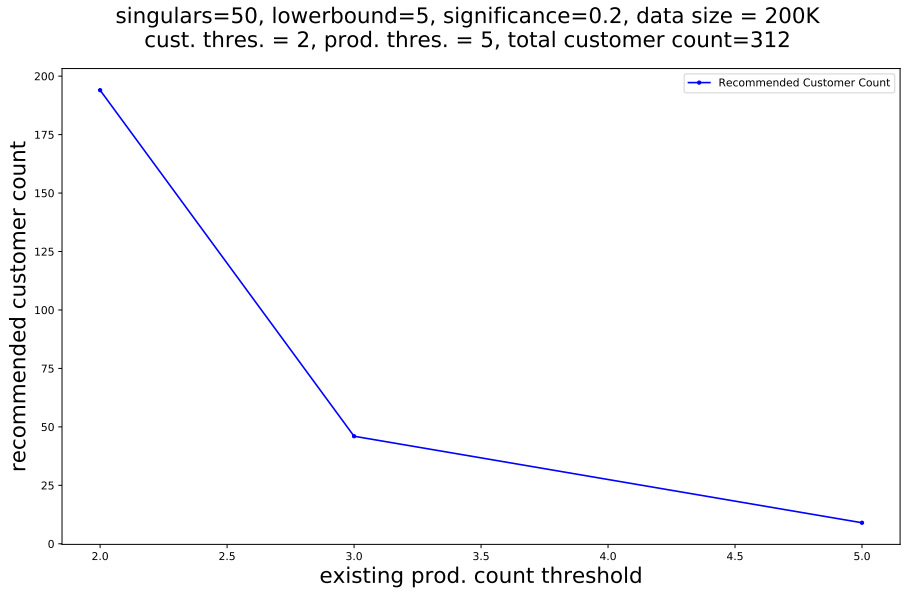


Figure 5.7: Existing Product Count vs. Recommended Customer Count

- After this is done for all customers in a fold, average of size of recommended product sets and average of accuracies are calculated. These two values are the results of the product-to-product recommendation for a certain fold. Results can be examined in **SVD Recommendation Results** section.

5.6.2 Customer-to-Customer Recommendation

5.6.2.1 Recommendation

The customer-to-customer recommendation aims to recommend products that are purchased by customers that are similar to the customer. As explained in **Application of SVD** section, SVD decomposes PFM and CFM to product of 3 different matrices. Before making recommendations to customer c_1, following steps are done:

1. SVD applied to PFM and U, S, V matrices are obtained.
2. SVD applied to CFM and U_2, S_2, V_2 matrices are obtained.
3. Matrix V represents the relationship between products and concepts. Matrix S is a square matrix, which consists of singular values in diagonal and remaining values are 0 (zero). Matrix U is a customer-concept relationship matrix.
4. Customer c_1's PFM row query is multiplied by V. The result is then multiplied by S^{-1} (inverse of matrix S). Now, the customer PFM row query is projected to customer-concept space.
5. Matrix V_2 represents the relationship between categories and concepts. To find similarity between two categories, row pairs of matrix V_2 are used. Matrix S_2 is a square matrix, which consists of singular values in diagonal and the remaining values are 0 (zero). Matrix U_2 is a customer-concept relationship matrix.
6. Customer c_1's CFM row query is multiplied by V_2. Result is then multiplied by S_2^{-1}. Now, customer CFM row query is projected to customer-concept space.
7. Similarities with other customers **by products** are found by finding cosine similarity between customer c_1's projected PFM row query and rows of matrix U. Similarities range from 0 to 1. The similarity value is 1 for the customer itself.
8. **S**imilarities with other customers **by product categories** are found by finding cosine similarity between customer c_1's projected CFM row query and rows of matrix U_2. Similarities range from 0 to 1. The similarity value is 1 for the customer itself.

9. A threshold that is called **"significance"** is applied to the results, just like it is done in **Product-to-Product Recommendation** section.

10. In addition to *significance* threshold, **"lowerbound"** and **"upperbound"** thresholds are applied just like it is done in **Product-to-Product Recommendation** section.

11. Number of similar customers whose products are recommended changes according to boundaries. Whole procedure of finding similar customers to customer c_1 is summarized below:

    $l \leftarrow$ lowerbound value

    $u \leftarrow$ upperbound value

    $\rightarrow$ Find cosine similarities with rows of matrix U. Sort results in descending order.

    $\rightarrow$ Check out how many customers are above the 'significance' threshold considering similarity results found before.

    $\rightarrow$ Do same operations for finding similar customers by product categories. This time, use matrix U_2.

    $n \leftarrow$ num. of sim. products that have sim. above 'significance'

    if $n \leq l$ **then**

      $\rightarrow$ Select customers with top l similarity values

    else

      $\rightarrow$ Select customers with top n similarity values

    end if

    if $n > u$ **then**

      $\rightarrow$ Select customers with top u similarity values

    end if

    if $include\_categories = True$ **then**

      while $n \leq u$ **do**

        $\rightarrow$ Add one customer from similar customers found by product category similarity and increment n

      end while

    end if

5.6.2.2 Evaluation

For evaluation of success in customer-to-customer recommendation, following method is applied to all customers. Let's assume that we want to make recommendations to customer c_1 with products p_1, p_2, p_3 and categories cat_1, cat_2, cat_3:

1. p_1, p_2, p_3 represents a PFM row query and cat_1, cat_2, cat_3 represents CFM row query.

2. **"Alternative customers"** from c_1 is generated considering c_1's PFM row query. These customers are basically clones of the customer c_1, except one of the products is removed from c_1's PFM row query. In addition to these, alternative customers from c_1 is generated considering c_1's CFM row query. Similarly, one of the categories is removed from c_1's CFM row query and considered as an alternative customer.

   To understand better, examine below explanation:

   $$alter\_cust\_1\_products \leftarrow [p2, p3]$$
   $$alter\_cust\_2\_products \leftarrow [p1, p3]$$
   $$alter\_cust\_3\_products \leftarrow [p1, p2]$$
   $$alter\_cust\_4\_categories \leftarrow [cat2, cat3]$$
   $$alter\_cust\_5\_categories \leftarrow [cat1, cat3]$$
   $$alter\_cust\_6\_categories \leftarrow [cat1, cat2]$$

3. After similar customers are found for each alternative customer, if union of products of similar customers contain the missing product, then it can be thought that recommendation is successful. Otherwise, recommendation fails. When all alternative customers are examined, total accuracy is calculated. Below example explains this more:

   $$alter\_cust\_1\_products = [p2, p3]$$
   $$sim\_cust\_by\_products \leftarrow [c2, c4]$$
   $$sim\_cust\_by\_categories \leftarrow [c3, c4]$$
   $$union(sim\_cust\_by\_products, sim\_cust\_by\_categories) = [c2, c3, c4]$$

   - If p1 is in **recommended product set** (products of customer union set), recommendation is **SUCCESSFUL**.

- If p1 is **NOT** in recommended product set, recommendation is **FAILED**.
- **Recommendation accuracy for customer c_1 => success / (success + failed)**

4. Above process is done as single fold evaluation and 10-fold cross-validation. In single fold evaluation, transactions of all customers are fed into the model while training. On the other hand, information of the 10% of the target customers is excluded from the model in 10-fold cross-validation.

5. After this is done for all customers in data, average of size of recommended product sets and average of accuracies are calculated. These two values are the results of the customer-to-customer recommendation. Results can be examined in **SVD Recommendation Results** section.

5.7 SVD Recommendation Results

Table 5.3: SVD P2P - single, singulars=5, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
5	10	2	11,7154	0,4863	160,4027
5	10	5	12,5781	0,4867	160,4206
5	10	8	13,4405	0,4871	160,4310
5	20	2	11,7154	0,4863	160,2721
5	20	5	12,5781	0,4867	160,2419
5	20	8	13,4405	0,4871	160,3385
5	30	2	11,7154	0,4863	160,2317
5	30	5	12,5781	0,4867	160,6559
5	30	8	13,4405	0,4871	160,4523
5	40	2	11,7145	0,4863	160,0826
5	40	5	12,5771	0,4867	160,0918
5	40	8	13,4405	0,4871	160,0860
5	50	2	11,7135	0,4863	159,9936
5	50	5	12,5771	0,4867	160,0694
5	50	8	13,4405	0,4871	160,0269
5	60	2	11,7119	0,4863	159,9294
5	60	5	12,5762	0,4867	159,9546
5	60	8	13,4405	0,4871	160,2816
5	70	2	11,7090	0,4863	159,9049
5	70	5	12,5733	0,4867	159,8507
5	70	8	13,4405	0,4871	159,8294
5	80	2	11,6786	0,4858	159,7446
5	80	5	12,5519	0,4862	159,8153
5	80	8	13,4405	0,4871	159,9190

Table 5.4: SVD P2P - single, singulars=10, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
10	10	2	11,8784	0,5328	180,7916
10	10	5	12,7414	0,5332	180,8203
10	10	8	13,6037	0,5336	180,8216
10	20	2	11,8784	0,5328	180,6480
10	20	5	12,7414	0,5332	180,8028
10	20	8	13,6037	0,5336	181,4573
10	30	2	11,8784	0,5328	180,8150
10	30	5	12,7414	0,5332	183,3487
10	30	8	13,6037	0,5336	180,4902
10	40	2	11,8771	0,5328	180,5663
10	40	5	12,7404	0,5332	180,4594
10	40	8	13,6037	0,5336	180,8795
10	50	2	11,8745	0,5328	180,2705
10	50	5	12,7385	0,5332	180,3640
10	50	8	13,6037	0,5336	180,2663
10	60	2	11,8012	0,5320	180,1840
10	60	5	12,6693	0,5325	180,4497
10	60	8	13,6037	0,5336	180,3236
10	70	2	11,4427	0,5278	180,2623
10	70	5	12,3985	0,5299	180,2802
10	70	8	13,6037	0,5336	180,6579
10	80	2	10,6168	0,5135	180,1408
10	80	5	11,9533	0,5233	183,3522
10	80	8	13,6037	0,5336	182,0333

Table 5.5: SVD P2P - single, singulars=50, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
50	10	2	13,6348	0,6816	144,9066
50	10	5	13,9875	0,6817	141,5885
50	10	8	14,3390	0,6822	141,6640
50	20	2	13,4609	0,6797	141,9210
50	20	5	13,8665	0,6804	141,4821
50	20	8	14,3390	0,6822	141,4671
50	30	2	13,2423	0,6781	141,4170
50	30	5	13,6815	0,6790	141,3952
50	30	8	14,3390	0,6822	141,4384
50	40	2	12,5848	0,6709	141,7022
50	40	5	13,2138	0,6739	141,6564
50	40	8	14,3390	0,6822	141,3804
50	50	2	11,7010	0,6583	141,5841
50	50	5	12,7702	0,6684	141,7034
50	50	8	14,3390	0,6822	141,4738
50	60	2	11,0403	0,6445	141,5487
50	60	5	12,4494	0,6622	141,3296
50	60	8	14,3390	0,6822	141,5485
50	70	2	10,4779	0,6301	141,4259
50	70	5	12,2807	0,6587	141,3004
50	70	8	14,3390	0,6822	141,3848
50	80	2	10,0739	0,6193	141,3846
50	80	5	12,1802	0,6576	141,2692
50	80	8	14,3390	0,6822	141,2981

Table 5.6: SVD P2P - single, singulars=100, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
100	10	2	15,4289	0,7198	129,7477
100	10	5	15,4651	0,7198	129,4681
100	10	8	15,5688	0,7208	129,5588
100	20	2	14,9408	0,7127	129,3348
100	20	5	15,0992	0,7161	129,6520
100	20	8	15,5688	0,7208	129,6693
100	30	2	14,0666	0,7037	130,1446
100	30	5	14,5605	0,7122	130,2077
100	30	8	15,5688	0,7208	130,8613
100	40	2	12,9494	0,6871	131,2972
100	40	5	13,9786	0,7030	129,3571
100	40	8	15,5688	0,7208	129,2669
100	50	2	12,2414	0,6702	129,2831
100	50	5	13,6405	0,6961	129,2577
100	50	8	15,5688	0,7208	129,2502
100	60	2	11,7404	0,6583	129,1657
100	60	5	13,5106	0,6946	129,2400
100	60	8	15,5688	0,7208	129,3218
100	70	2	11,2852	0,6512	129,2606
100	70	5	13,3134	0,6933	129,1775
100	70	8	15,5688	0,7208	129,3063
100	80	2	10,7628	0,6392	129,2460
100	80	5	13,0707	0,6896	129,1674
100	80	8	15,5688	0,7208	129,3275

Table 5.7: SVD P2P - single, singulars=250, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
250	10	2	14,8249	0,7874	130,0746
250	10	5	14,9478	0,7892	130,1167
250	10	8	15,3908	0,7957	129,8438
250	20	2	13,5070	0,7679	130,0174
250	20	5	14,1197	0,7786	129,8716
250	20	8	15,3908	0,7957	129,7710
250	30	2	12,1482	0,7467	130,0079
250	30	5	13,4677	0,7695	129,7007
250	30	8	15,3908	0,7957	129,7249
250	40	2	11,1735	0,7302	129,7090
250	40	5	12,9712	0,7643	129,7003
250	40	8	15,3908	0,7957	129,7215
250	50	2	10,5675	0,7197	129,7405
250	50	5	12,6825	0,7617	129,7077
250	50	8	15,3908	0,7957	129,6695
250	60	2	9,9302	0,7077	129,7276
250	60	5	12,3793	0,7591	129,7255
250	60	8	15,3908	0,7957	129,7853
250	70	2	9,2532	0,6937	129,6240
250	70	5	12,0317	0,7557	129,7927
250	70	8	15,3908	0,7957	129,7980
250	80	2	8,5506	0,6737	129,7823
250	80	5	11,7161	0,7514	129,7366
250	80	8	15,3908	0,7957	129,7665

Table 5.8: SVD P2P - single, singulars=500, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
500	10	2	13,5387	0,7922	128,2258
500	10	5	14,2650	0,8065	128,1530
500	10	8	15,7570	0,8283	128,1084
500	20	2	11,2875	0,7570	128,0727
500	20	5	12,9802	0,7934	128,0171
500	20	8	15,7570	0,8283	128,1072
500	30	2	9,7423	0,7375	128,0509
500	30	5	12,1444	0,7884	128,0128
500	30	8	15,7570	0,8283	128,0653
500	40	2	8,6284	0,7205	128,0363
500	40	5	11,5768	0,7844	128,1512
500	40	8	15,7570	0,8283	128,0628
500	50	2	7,7551	0,7047	128,0108
500	50	5	11,1764	0,7810	128,0549
500	50	8	15,7570	0,8283	128,0587
500	60	2	7,1021	0,6896	128,0175
500	60	5	10,9334	0,7794	128,0223
500	60	8	15,7570	0,8283	128,8045
500	70	2	6,5829	0,6789	130,1323
500	70	5	10,7359	0,7774	129,8904
500	70	8	15,7570	0,8283	128,1168
500	80	2	6,1399	0,6641	128,3258
500	80	5	10,6044	0,7761	129,4422
500	80	8	15,7570	0,8283	128,6328

Table 5.9: SVD P2P - single, singulars=750, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
750	10	2	11,3204	0,7642	131,2300
750	10	5	13,0096	0,8039	131,2829
750	10	8	15,8303	0,8407	131,0602
750	20	2	9,0560	0,7317	131,0842
750	20	5	11,7743	0,7952	131,0453
750	20	8	15,8303	0,8407	131,0658
750	30	2	7,6639	0,7117	131,0718
750	30	5	11,1277	0,7915	130,9781
750	30	8	15,8303	0,8407	131,0052
750	40	2	6,7513	0,6921	130,9894
750	40	5	10,7823	0,7886	130,9751
750	40	8	15,8303	0,8407	131,4870
750	50	2	6,1498	0,6774	131,2631
750	50	5	10,5919	0,7871	131,2600
750	50	8	15,8303	0,8407	131,3154
750	60	2	5,7327	0,6657	131,2558
750	60	5	10,4664	0,7862	134,7240
750	60	8	15,8303	0,8407	156,9054
750	70	2	5,3835	0,6544	159,3580
750	70	5	10,3739	0,7854	157,3093
750	70	8	15,8303	0,8407	157,7907
750	80	2	5,1536	0,6459	150,7697
750	80	5	10,3319	0,7849	136,1938
750	80	8	15,8303	0,8407	132,5682

Table 5.10: SVD P2P - single, singulars=1000, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
1000	10	2	9,3976	0,7397	134,6320
1000	10	5	11,9453	0,7918	133,4242
1000	10	8	15,7625	0,8326	136,1518
1000	20	2	7,5301	0,7061	136,4642
1000	20	5	11,1210	0,7853	134,5905
1000	20	8	15,7625	0,8326	134,9084
1000	30	2	6,6463	0,6846	132,6561
1000	30	5	10,8121	0,7824	130,9702
1000	30	8	15,7625	0,8326	131,2225
1000	40	2	6,0653	0,6680	130,8697
1000	40	5	10,6277	0,7805	130,9330
1000	40	8	15,7625	0,8326	131,4631
1000	50	2	5,6031	0,6561	131,2963
1000	50	5	10,4901	0,7796	131,3958
1000	50	8	15,7625	0,8326	131,4279
1000	60	2	5,2823	0,6484	131,4951
1000	60	5	10,3822	0,7790	131,5744
1000	60	8	15,7625	0,8326	131,9860
1000	70	2	5,0170	0,6410	132,4721
1000	70	5	10,3031	0,7785	132,8670
1000	70	8	15,7625	0,8326	132,7731
1000	80	2	4,8585	0,6357	132,6534
1000	80	5	10,2737	0,7782	133,3197
1000	80	8	15,7625	0,8326	133,2872

Table 5.11: SVD P2P - single, singulars=1500, significance=0.2, data size = 200K

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
1500	10	2	6,7887	0,4958	137,8461
1500	10	5	10,7964	0,5552	138,0718
1500	10	8	15,8188	0,5917	137,7414
1500	20	2	5,5230	0,4749	137,7791
1500	20	5	10,3956	0,5527	137,8417
1500	20	8	15,8188	0,5917	138,7766
1500	30	2	5,0567	0,4647	141,2149
1500	30	5	10,2878	0,5516	146,0559
1500	30	8	15,8188	0,5917	142,2191
1500	40	2	4,8399	0,4603	141,0715
1500	40	5	10,2487	0,5514	142,0461
1500	40	8	15,8188	0,5917	144,3924
1500	50	2	4,7029	0,4567	145,8517
1500	50	5	10,2321	0,5511	142,7950
1500	50	8	15,8188	0,5917	144,8857
1500	60	2	4,6172	0,4544	141,4535
1500	60	5	10,2138	0,5511	143,1852
1500	60	8	15,8188	0,5917	144,9033
1500	70	2	4,5365	0,4523	143,8953
1500	70	5	10,2074	0,5511	143,5057
1500	70	8	15,8188	0,5917	144,2396
1500	80	2	4,4773	0,4507	150,8798
1500	80	5	10,2007	0,5511	142,8283
1500	80	8	15,8188	0,5917	143,1212

Table 5.12: SVD P2P - 10-fold cross, singulars=50, exist >= 2, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Recomm. Cust.
0	14,5722	0,1943	313	194
1	14,2865	0,1805	313	178
2	14,3446	0,1727	313	177
3	14,4108	0,1618	313	185
4	13,6354	0,1445	312	181
5	15,1061	0,1602	312	198
6	15,3918	0,1995	312	171
7	15,3015	0,1770	312	199
8	15,0966	0,1890	312	176
9	14,6978	0,1945	312	182

Table 5.13: SVD P2P - 10-fold cross, singulars=50, exist >= 3, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Recomm. Cust.
0	22,6087	0,2540	313	46
1	21,8333	0,2295	313	42
2	21,1731	0,2513	313	52
3	20,6271	0,2192	313	59
4	20,2162	0,2338	312	37
5	23,3396	0,1738	312	53
6	22,4600	0,2321	312	50
7	22,8167	0,2619	312	60
8	22,4694	0,2300	312	49
9	22,1087	0,1936	312	46

Table 5.14: SVD P2P - 10-fold cross, singulars=50, exist >= 5, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Recomm. Cust.
0	38,8889	0,3077	313	9
1	30,8889	0,5246	313	9
2	33,0000	0,2615	313	13
3	31,5556	0,3725	313	9
4	31,1667	0,3167	312	6
5	34,0714	0,2831	312	14
6	34,8000	0,2690	312	10
7	36,5455	0,3452	312	11
8	32,5385	0,2515	312	13
9	34,3000	0,2157	312	10

Table 5.15: SVD P2P - 10-fold cross, singulars=100, exist >= 2, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Recomm. Cust.
0	13,7423	0,1937	313	194
1	13,9775	0,1594	313	178
2	14,5989	0,1858	313	177
3	14,4270	0,1479	313	185
4	12,9558	0,1301	312	181
5	14,3990	0,1377	312	198
6	14,8889	0,1712	312	171
7	14,9749	0,1683	312	199
8	14,7102	0,1724	312	176
9	14,4121	0,1931	312	182

Table 5.16: SVD P2P - 10-fold cross, singulars=100, exist >= 3, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Recomm. Cust.
0	21,3261	0,2519	313	46
1	20,4762	0,2111	313	42
2	20,9808	0,2093	313	52
3	19,4237	0,2096	313	59
4	18,9189	0,1905	312	37
5	21,3019	0,0898	312	53
6	20,6400	0,1953	312	50
7	21,0167	0,2249	312	60
8	20,7551	0,2008	312	49
9	21,0435	0,2096	312	46

Table 5.17: SVD P2P - 10-fold cross, singulars=100, exist >= 5, lowerbound=5, significance=0.2, data size = 200K, cust. thres. = 2, prod. thres. = 5

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Recomm. Cust.
0	34,8889	0,2595	313	9
1	29,3333	0,4665	313	9
2	30,6154	0,1833	313	13
3	26,6667	0,3095	313	9
4	29,5000	0,2306	312	6
5	30,3571	0,1375	312	14
6	31,2000	0,2183	312	10
7	30,6364	0,2345	312	11
8	28,3077	0,1799	312	13
9	31,5000	0,1973	312	10

Table 5.18: SVD P2P - 10-fold cross, singulars = 50, exist >= 3, low = 5, significance=0.2, data size = 2.5M, cust. thres. = 10, prod. thres. = 50

Fold #	Rec. Prod. Count	Accuracy	Accuracy without A.S.F.	Num. of Cust.	Exist Prod. Count
0	50,8288	0,3926	0,2035	222	13,9595
1	49,7568	0,4152	0,2253	222	14,0270
2	48,0631	0,4288	0,2406	222	13,4414
3	49,9910	0,3915	0,2107	222	13,9234
4	50,6516	0,4003	0,2036	221	13,7964
5	49,0181	0,3853	0,2007	221	13,2262
6	52,1041	0,4014	0,2208	221	14,3846
7	53,1176	0,3918	0,2153	221	14,6109
8	51,1810	0,4036	0,2225	221	14,1086
9	49,5520	0,3973	0,2085	221	13,6606

Table 5.19: SVD P2P - 10-fold cross, singulars = 50, exist >= 3, low = 5, significance=0.2, data size = 2.5M, cust. thres. = 20, prod. thres. = 50

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Exist Prod. Count
0	97,8148	0,3653	27	26,7778
1	98,4444	0,3662	27	26,7037
2	87,0741	0,3336	27	22,7037
3	83,2593	0,3659	27	22,4074
4	97,2222	0,3817	27	26,5185
5	88,2963	0,4074	27	24,4815
6	82,3462	0,4550	26	25,0385
7	96,6296	0,3833	27	25,8519
8	87,9259	0,4404	27	24,0000
9	91,7037	0,3683	27	24,2222

Table 5.20: SVD P2P - 10-fold cross, singulars = 50, exist >= 3, low = 5, significance=0.2, data size = 2.5M, cust. thres. = 10, prod. thres. = 60

Fold #	Rec. Prod. Count	Accuracy	Num. of Cust.	Exist Prod. Count
0	50,8090	0,3950	199	14,1910
1	52,5202	0,4106	198	14,6818
2	50,1212	0,3966	198	13,9394
3	48,9697	0,3765	198	13,4141
4	48,8535	0,3978	198	13,4192
5	49,8081	0,4084	198	14,0404
6	47,1869	0,4072	198	12,9495
7	49,6667	0,4105	198	13,7778
8	49,4495	0,4001	198	13,7172
9	50,2626	0,4132	198	14,0354

Table 5.21: SVD C2C - single, singulars=50, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
50	20	2	18,4536	0,2312	4702,5027
50	20	5	19,1661	0,2312	4672,1858
50	20	8	19,8787	0,2312	4684,6581
50	50	2	18,1965	0,2310	4687,8889
50	50	5	19,0333	0,2312	4692,9597
50	50	8	19,8787	0,2312	4686,2308
50	80	2	17,1328	0,2188	4683,7405
50	80	5	18,4603	0,2250	4674,6399
50	80	8	19,8787	0,2312	4688,1245

Table 5.22: SVD C2C - single, singulars=100, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
100	20	2	19,1303	0,3165	4702,9823
100	20	5	19,2033	0,3165	4689,1078
100	20	8	19,2775	0,3165	4699,1564
100	50	2	18,5647	0,3150	4708,1858
100	50	5	18,8995	0,3160	4720,9611
100	50	8	19,2775	0,3165	4698,1465
100	80	2	16,4408	0,2902	4717,6381
100	80	5	17,6328	0,3021	4704,9448
100	80	8	19,2775	0,3165	4717,0770

Table 5.23: SVD C2C - single, singulars=250, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
250	20	2	16,4385	0,4179	4785,1616
250	20	5	16,4782	0,4179	4780,5992
250	20	8	16,6021	0,4183	4787,7474
250	50	2	14,6431	0,4105	4781,4325
250	50	5	15,3646	0,4124	4788,5946
250	50	8	16,6021	0,4183	4798,9668
250	80	2	11,1722	0,3547	4799,5110
250	80	5	13,3643	0,3812	4800,0721
250	80	8	16,6021	0,4183	4808,1097

Table 5.24: SVD C2C - single, singulars=500, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
500	20	2	14,8006	0,4356	4958,0358
500	20	5	15,3640	0,4385	5009,8288
500	20	8	17,0128	0,4464	5121,1852
500	50	2	11,1319	0,3888	5022,7590
500	50	5	13,2753	0,4140	4946,8331
500	50	8	17,0128	0,4464	4931,8107
500	80	2	7,4885	0,3183	4941,6503
500	80	5	11,6761	0,3871	4943,8480
500	80	8	17,0128	0,4464	4952,1467

Table 5.25: SVD C2C - single, singulars=1000, significance=0.2, data size=200K, cust. thres. = 2, prod. thres. 5

Singular Values	Significance	Lower Bound	Rec. Prod. Count	Accuracy	Total Time (sec)
1000	20	2	11,7465	0,3767	5435,5399
1000	20	5	14,4974	0,3989	5432,5109
1000	20	8	18,4850	0,4130	5441,4779
1000	50	2	9,1517	0,3207	5433,6882
1000	50	5	13,1316	0,3817	5463,6312
1000	50	8	18,4850	0,4130	5450,3495
1000	80	2	6,2177	0,2733	5557,1359
1000	80	5	12,1421	0,3666	5599,0893
1000	80	8	18,4850	0,4130	5717,2521

Table 5.26: SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 20, prod. thres. = 50, recommend **at least** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
29,0741	26,7778	10,8889	0,3922
29,6296	26,7037	11,1481	0,4080
25,2963	22,7037	9,0000	0,4206
24,1481	22,4074	10,0741	0,4413
28,2222	26,5185	12,6296	0,4607
26,8148	24,4815	12,9259	0,5196
26,5926	24,1852	13,6667	0,5388
27,7778	25,8519	11,2963	0,4661
27,0000	24,0000	10,5556	0,4489
26,7407	24,2222	11,0000	0,4621

Table 5.27: SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 20, prod. thres. = 50, recommend **exactly** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
29,0741	26,7778	9,5926	0,3564
29,6296	26,7037	9,6667	0,3550
25,2963	22,7037	8,4444	0,3938
24,1481	22,4074	8,9259	0,3944
28,2222	26,5185	10,5185	0,3931
26,8148	24,4815	10,1481	0,4098
26,5926	24,1852	11,0370	0,4520
27,7778	25,8519	8,6296	0,3559
27,0000	24,0000	9,0741	0,3889
26,7407	24,2222	8,9259	0,3758

Table 5.28: SVD C2C - 10-fold cross, singulars = 50, low = 5, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 50, recommend **at least** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
14,2928	13,9595	8,4730	0,6135
14,3874	14,0270	7,9955	0,5717
13,6486	13,4414	7,8784	0,5997
14,2117	13,9234	8,1667	0,5934
14,1312	13,7964	7,8959	0,5873
13,5973	13,2262	7,7285	0,6005
14,8507	14,3846	8,3167	0,5745
14,9593	14,6109	8,9095	0,6039
14,5656	14,1086	8,5973	0,6098
14,0860	13,6606	7,9005	0,5831

Table 5.29: SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 50, recommend **exactly** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
14,2928	13,9595	3,3153	0,2442
14,3874	14,0270	3,4414	0,2453
13,6486	13,4414	3,6532	0,2837
14,2117	13,9234	3,5225	0,2466
14,1312	13,7964	3,1719	0,2404
13,5973	13,2262	3,0271	0,2338
14,8507	14,3846	3,1719	0,2253
14,9593	14,6109	3,3937	0,2374
14,5656	14,1086	3,4977	0,2513
14,0860	13,6606	3,1176	0,2286

Table 5.30: SVD C2C - 10-fold cross, singulars = 50, **significance=0.2**, data size = 2.5M, cust. thres. = 10, prod. thres. = 60, recommend **at least** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
14,4523	14,1910	13,1960	0,9275
14,9397	14,6080	13,5729	0,9254
14,1465	13,9394	12,9899	0,9318
13,6515	13,4141	12,5051	0,9314
13,8687	13,4192	12,4343	0,9202
14,3586	14,0404	13,1869	0,9340
13,2525	12,9495	12,1111	0,9316
14,0303	13,7778	12,8485	0,9318
14,1313	13,7172	12,6616	0,9229
14,2626	14,0354	13,0000	0,9222

Table 5.31: SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 60, recommend **at least** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
14,4523	14,1910	7,4874	0,5284
14,9397	14,6080	7,4874	0,5296
14,1465	13,9394	6,9949	0,5121
13,6515	13,4141	7,2121	0,5473
13,8687	13,4192	7,1263	0,5360
14,3586	14,0404	7,6364	0,5649
13,2525	12,9495	7,8838	0,6082
14,0303	13,7778	7,1768	0,5383
14,1313	13,7172	7,7879	0,5687
14,2626	14,0354	7,3182	0,5294

Table 5.32: SVD C2C - 10-fold cross, singulars = 50, **significance=0.5**, data size = 2.5M, cust. thres. = 10, prod. thres. = 60, recommend **exactly** 5 cust. for each alter. cust.

Actual Prod. Count	Exist. Prod. Count	Match. Prod. Count	Accuracy
14,4523	14,1910	3,2312	0,2306
14,9397	14,6080	3,3568	0,2339
14,1465	13,9394	3,1768	0,2304
13,6515	13,4141	3,2525	0,2498
13,8687	13,4192	3,2172	0,2415
14,3586	14,0404	3,5707	0,2637
13,2525	12,9495	3,2222	0,2564
14,0303	13,7778	3,1869	0,2427
14,1313	13,7172	3,4242	0,2538
14,2626	14,0354	3,3485	0,2439

CHAPTER 6

ASSOCIATION RULE MINING (ARM) ON THE DATA

6.1 Introduction

In addition to recommendation methods mentioned in chapter **Singular Value Decomposition (SVD) on the Data**, association rules about within data are mined and evaluated to come up with another recommendation method.

Firstly, data is pre-processed and divided into 10 partitions. After that, frequent product sets within 9/10 of partitions are found with *Apriori* algorithm in a way that is explained in [19]. Frequent product sets are found among all products and transactions, instead of following target products concept mentioned **previously**.

After frequent product sets are found, association rules between items are mined. If there is a product set with products [p_1, p_2], rules can be $p_1 \rightarrow p_2$ (those who buy p_1 is likely to buy p_2) or $p_2 \rightarrow p_1$. Whether any of these rules are consistent or not depends on their *support*, *confidence* and *lift* values. How these values affect the consistency is explained in chapter **Technical Background**.

Finally, association rules are evaluated with 10-fold cross-validation method by using shopping carts of the current (test) partition.

6.2 Data Preprocessing for ARM

In *Apriori* algorithm, the problem defined as "finding frequent itemsets among transactions that contain multiple products in them". In real life, transactions with multiple products are called "shopping carts". These shopping carts can contain related, same

or completely independent multiple products. On the other hand, most of the transactions (shoppings) in the data used for this thesis work contains only one product, which is stated previously in chapter **Data Preprocessing**.

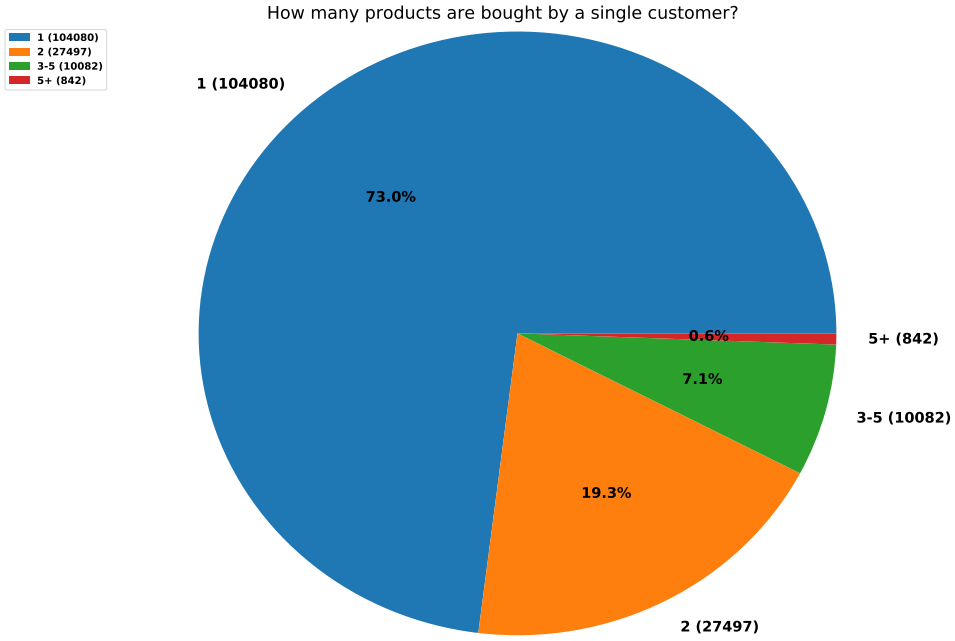


Figure 6.1: Shopping-Product count histogram (pie) - 200K transactions

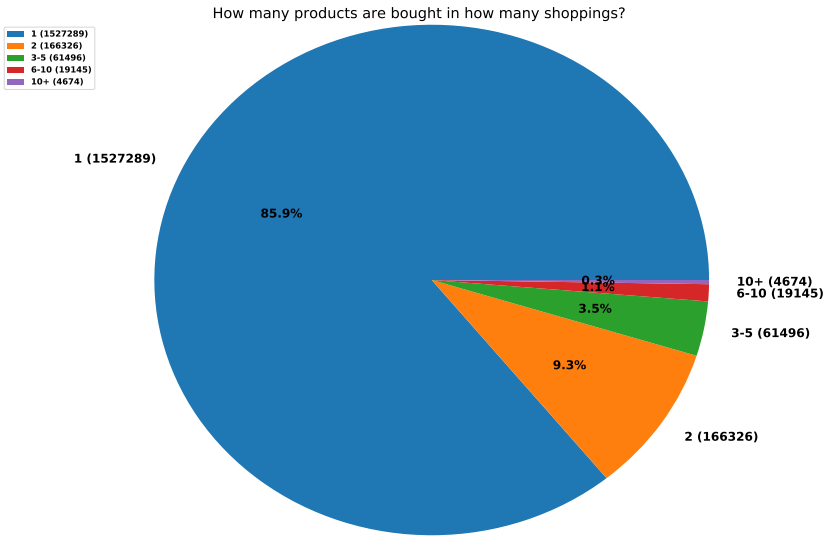


Figure 6.2: Shopping-Product count histogram (pie) - 2.5M transactions

With this information in mind, in addition to preprocessing work mentioned in chapter **Data Statistics & Preprocessing**, following work is done:

- Since most of the transactions are single purchases, it is difficult to consider these transactions as shopping carts. To overcome this problem, transactions that are done in same **time period** are considered as a shopping cart.

  For better understanding, lets consider below simple transactions:

  ```
  t1: p1 on 24 May 2019 by customer c1
  t2: p2 on 25 May 2019 by c1
  t3: p2 on 26 May 2019 by c2
  t4: p3 on 28 May 2019 by c2
  t5: p4 on 29 May 2019 by c2
  ```

- Time period-transaction mapping is constructed. Time period can be **weekly** or **monthly** period. Here is an example for weekly mapping:

  ```
  week 1 (20 May-26 May 2019) => t1, t2, t3
  week 2 (27 May-2 Jun 2019) => t4, t5
  ```

- Using time period-transaction mapping, for all distinct weeks/months, transactions are separated into smaller groups according to the customer id of the transaction. These final groupings are called shopping carts.

  ```
  cart 1: [p1, p2] by c1
  cart 2: [p2] by c2
  cart 3: [p3, p4] by c2
  ```

- When grouping by time period is done, there are still shopping carts with only 1 product exists. These 1-product shopping carts are **excluded** and remaining shopping carts are used for exploring frequent product sets.

  ```
  cart 1: [p1, p2] by c1
  *cart 2: [p2] by c2     <= EXCLUDED
  cart 3: [p3, p4] by c2
  ```

- All shopping carts are shuffled and partitioned into 10 parts to do 10-fold cross-validation. For each partition, association rules are mined from shopping carts

in other 9 partitions and evaluated with shopping carts in the current (test) partition. Whole procedure can be seen in **Figure 6.2**.

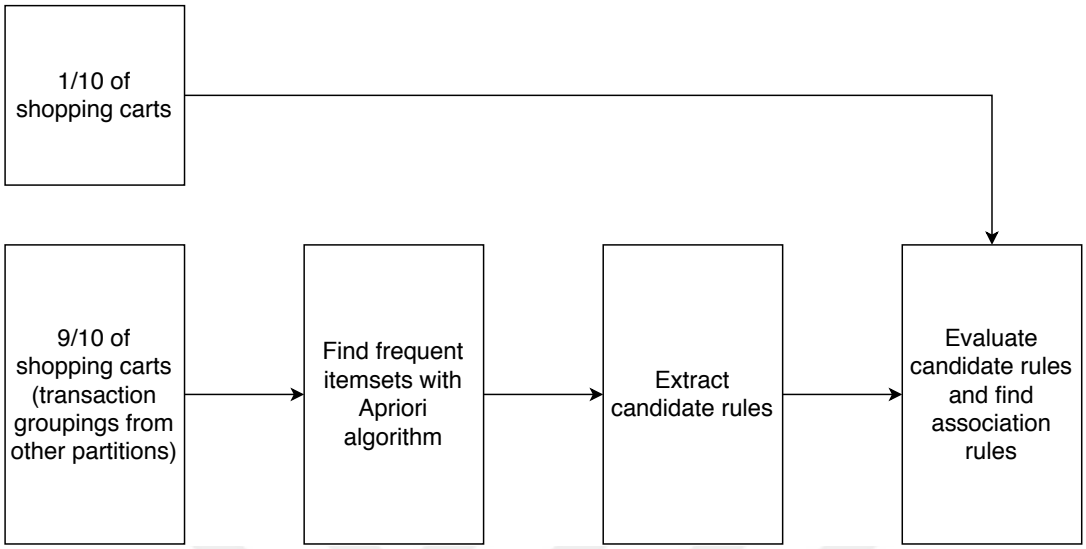


Figure 6.3: Visual explanation for ARM procedure with 10-partitions

6.3 Application of Apriori Algorithm

Frequent product sets are found by applying Apriori algorithm. When transactions are grouped by weekly and support threshold = 5, the following results are found for each partition:

Table 6.1: ARM - Shopping carts and frequent product sets per partition (weekly - 200K transactions)

Part	Shop. carts	Freq. Prod. Sets
0	6313	238
1	6313	246
2	6313	242
3	6313	238
4	6313	245
5	6314	239
6	6314	242
7	6314	243
8	6314	248
9	6314	248

When transactions are grouped by monthly and support threshold = 5, following results are found for each partition:

Table 6.2: ARM - Shopping carts and frequent product sets per partition (monthly, sup=0.0007, 200K transactions)

Part	Shop. carts	Freq. Prod. Sets
0	6798	274
1	6798	271
2	6798	257
3	6798	266
4	6799	260
5	6799	262
6	6799	266
7	6799	255
8	6799	262
9	6799	264

Table 6.3: ARM - Shopping carts and frequent product sets per partition (monthly, sup=0.001, 2.5M transactions)

Part	Shop. carts	Freq. Prod. Sets
0	189865	19
1	189866	20
2	189865	20

6.4 Mining Association Rules

Various candidate rules that are mined from frequent product sets are tested. If a candidate rule has higher support value than *support threshold* and higher confidence value than *min_confidence*, this candidate rule is an association rule mined from the other 9 partitions. Assuming that there are 2-product sets, 3-product sets and 4-product sets:

- If frequent product set is in format [a,b] with length=2, following candidate rules and their inverse are tested:
  - a $\rightarrow$ b

- If frequent product set is in format [a,b,c] with length=3, following candidate rules and their inverse are tested:
  - a $\rightarrow$ bc
  - ab $\rightarrow$ c
  - ac $\rightarrow$ b
- If frequent product set is in format [a,b,c,d] with length=4, following candidate rules and their inverse are tested:
  - a $\rightarrow$ bcd
  - ab $\rightarrow$ cd
  - ac $\rightarrow$ bd
  - ad $\rightarrow$ bc
  - abc $\rightarrow$ d
  - acd $\rightarrow$ b
  - abd $\rightarrow$ c

6.5 Association Rule Evaluation

After association rules are found from other 9 partitions, these rules are evaluated using shopping carts of test partition. For some of the association rules, products on the left-hand side (LHS) of a rule don't exist in shopping carts of test partition. While doing the evaluation, such rules are **ignored**. The ratio between "rules with products on LHS found in shopping carts of test partition" and the total number of rules is given in results below.

Rule accuracy is simply the precision of the rule in the test partition. The ratio between "number of shopping carts that contain products on both LHS and RHS of the rule" and "number of shopping carts that contain products on LHS of the rule" is considered as accuracy. This is also explained below:

```
Rule mined from 9/10 partitions: P1 => P2
Number of shopping carts containing P1 in test partition = M  (LHS
```

```
Found in test partition)
Number of shopping carts containing P1 and P2 in test partition = N
Accuracy = M/N
```

When transactions are grouped by weekly, support threshold = 5 and min_confidence = 25%, following results are found for each partition:

Part	Freq. Prod. Sets	Total Rules	LHS Found	LHS Found Ratio	Accuracy
0	238	11	10	0,9091	0,2750
1	246	10	7	0,7000	0,0833
2	242	7	7	1,0000	0,5000
3	238	10	6	0,6000	0,0000
4	245	10	7	0,7000	0,5952
5	239	10	6	0,6000	0,1944
6	242	11	9	0,8182	0,2037
7	243	8	6	0,7500	0,4583
8	248	11	10	0,9091	0,2833
9	248	14	10	0,7143	0,3000
Avg.	242,9	10,2	7,8	0,7701	0,2893

When transactions are grouped by monthly, support threshold = 5 and min_confidence = 25%, following results are found for each partition:

Part	Freq. Prod. Sets	Total Rules	LHS Found	LHS Found Ratio	Accuracy
0	274	16	11	0,6875	0,2121
1	271	13	12	0,9231	0,3611
2	257	15	12	0,8000	0,3056
3	266	13	11	0,8462	0,2955
4	260	9	9	1,0000	0,3000
5	262	9	6	0,6667	0,6000
6	266	12	6	0,5000	0,1667
7	255	13	11	0,8462	0,2727
8	262	14	8	0,5714	0,1771
9	264	10	8	0,8000	0,0500
Avg.	263,7	12,4	9,4	0,7641	0,2741

6.6 ARM Results

You can find ARM results for data with 200K transactions and 2.5M transactions in the tables below. Configurations are stated as captions of the tables.

Table 6.4: ARM Results - 10-fold cross, **momently carts**, min confidence=25%, **sup=5/num_of_carts**, data size=200K

Part	Period	Carts	Support	Itemsets	Rules	LHS Found Rules	LHS Found Rules Avg. Accuracy
0	moment	5570	0,00089767	216	6	4	0,1875
1	moment	5570	0,00089767	214	10	5	0,1667
2	moment	5570	0,00089767	219	8	5	0,0400
3	moment	5570	0,00089767	217	7	4	0,2083
4	moment	5570	0,00089767	221	6	5	0,1500
5	moment	5570	0,00089767	223	3	2	0,0000
6	moment	5570	0,00089767	219	9	7	0,0714
7	moment	5570	0,00089767	214	7	5	0,1000
8	moment	5570	0,00089767	218	11	8	0,1250
9	moment	5571	0,00089750	220	8	5	0,1000

Table 6.5: ARM Results - 10-fold cross, **weekly carts**, min confidence=25%, **sup=5/num_of_carts**, data size=200K

Part	Period	Carts	Support	Itemsets	Rules	LHS Found Rules	LHS Found Rules Avg. Accuracy
0	week	6313	0,00079202	238	11	10	0,2750
1	week	6313	0,00079202	246	10	7	0,0833
2	week	6313	0,00079202	242	7	7	0,5000
3	week	6313	0,00079202	238	10	6	0,0000
4	week	6313	0,00079202	245	10	7	0,5952
5	week	6314	0,00079189	239	10	6	0,1944
6	week	6314	0,00079189	242	11	9	0,2037
7	week	6314	0,00079189	243	8	6	0,4583
8	week	6314	0,00079189	248	11	10	0,2833
9	week	6314	0,00079189	248	14	10	0,3000

Table 6.6: ARM Results - 10-fold cross, **monthly carts**, min confidence=25%, **sup=5/num_of_carts**, data size=200K

Part	Period	Carts	Support	Itemsets	Rules	LHS Found Rules	LHS Found Rules Avg. Accuracy
0	month	6798	0,00073551	274	16	11	0,2121
1	month	6798	0,00073551	271	13	12	0,3611
2	month	6798	0,00073551	257	15	12	0,3056
3	month	6798	0,00073551	266	13	11	0,2955
4	month	6799	0,00073540	260	9	9	0,3000
5	month	6799	0,00073540	262	9	6	0,6000
6	month	6799	0,00073540	266	12	6	0,1667
7	month	6799	0,00073540	255	13	11	0,2727
8	month	6799	0,00073540	262	14	8	0,1771
9	month	6799	0,00073540	264	10	8	0,0500

Table 6.7: ARM Results - 10-fold cross, **momently carts**, min confidence=25%, **sup=7/num_of_carts**, data size=200K

Part	Period	Carts	Support	Itemsets	Rules	LHS Found Rules	LHS Found Rules Avg. Accuracy
0	moment	5570	0,00125673	121	1	1	0,5000
1	moment	5570	0,00125673	119	2	2	0,1667
2	moment	5570	0,00125673	122	2	2	0,1000
3	moment	5570	0,00125673	119	1	1	0,3333
4	moment	5570	0,00125673	120	1	0	0,0000
5	moment	5570	0,00125673	126	2	2	0,0000
6	moment	5570	0,00125673	124	2	2	0,2500
7	moment	5570	0,00125673	119	2	1	0,5000
8	moment	5570	0,00125673	120	2	2	0,2500
9	moment	5571	0,00125651	123	2	1	0,5000

Table 6.8: ARM Results - 10-fold cross, **weekly carts**, min confidence=25%, **sup=7/num_of_carts**, data size=200K

Part	Period	Carts	Support	Itemsets	Rules	LHS Found Rules	LHS Found Rules Avg. Accuracy
0	week	6313	0,00110882	134	3	3	0,0833
1	week	6313	0,00110882	136	2	2	0,1667
2	week	6313	0,00110882	136	3	3	0,3333
3	week	6313	0,00110882	136	5	4	0,0000
4	week	6313	0,00110882	132	2	1	0,6667
5	week	6314	0,00110865	136	4	4	0,1250
6	week	6314	0,00110865	133	2	2	0,2500
7	week	6314	0,00110865	133	2	1	0,0000
8	week	6314	0,00110865	133	4	4	0,0833
9	week	6314	0,00110865	137	4	4	0,0000

Table 6.9: ARM Results - 10-fold cross, **monthly carts**, min confidence=25%, **sup=7/num_of_carts**, data size=200K

Part	Period	Carts	Support	Itemsets	Rules	LHS Found Rules	LHS Found Rules Avg. Accuracy
0	month	6798	0,00102971	143	3	3	0,1111
1	month	6798	0,00102971	146	3	3	0,6667
2	month	6798	0,00102971	142	4	4	0,0833
3	month	6798	0,00102971	141	4	4	0,3125
4	month	6799	0,00102956	144	4	4	0,3000
5	month	6799	0,00102956	150	2	2	0,5000
6	month	6799	0,00102956	146	4	3	0,3333
7	month	6799	0,00102956	142	4	3	0,3333
8	month	6799	0,00102956	141	3	3	0,2778
9	month	6799	0,00102956	146	4	4	0,1000

Table 6.10: ARM Results - 10-fold cross, **monthly carts**, min confidence=20%, **sup=0.001**, data size=2.5M

Part	Period	LHS Found Carts	LHS Found Cust	LHS-RHS Found Carts	LHS-RHS Found Cust	LHS Not Found Carts	LHS Not Found Cust	Precision (Accuracy)	Rules	Itemsets
0	month	98,9688	98,7500	34,2188	34,2188	20998,0313	20630,2500	0,3939	32	19
1	month	96,6786	96,1071	30,5357	30,2857	21000,3214	20599,8929	0,3426	56	20
2	month	91,8571	91,6964	27,5893	27,5357	21004,1429	20631,3036	0,3306	56	20
3	month	88,3214	88,2321	29,4286	29,3929	21007,6786	20635,7679	0,3694	56	20
4	month	86,3929	85,5893	28,0536	27,8214	21009,6071	20649,4107	0,3588	56	20
5	month	99,6563	99,3750	33,7188	33,6250	20996,3438	20622,6250	0,3822	32	19
6	month	101,2222	100,6111	31,9167	31,8611	20994,7778	20623,3889	0,3551	36	19
7	month	105,6944	104,8333	31,8611	31,8056	20990,3056	20664,1667	0,3437	36	19
8	month	83,7857	83,4643	25,5714	25,5714	21012,2143	20667,5357	0,3340	56	20
9	month	92,1964	91,8214	27,1429	27,0714	21003,8036	20631,1786	0,3212	56	20

Table 6.11: ARM Results - 10-fold cross, **monthly carts**, min confidence=20%, **sup=0.001**, data size=2.5M, **platform="epttavm"**

Part	Period	LHS Found Carts	LHS Found Cust	LHS-RHS Found Carts	LHS-RHS Found Cust	LHS Not Found Carts	LHS Not Found Cust	Precision (Accuracy)	Rules	Itemsets
0	month	2,6923	2,6805	1,0828	1,0769	654,3077	640,3195	0,4861	201	112
1	month	2,2928	2,2762	0,7459	0,7459	654,7072	642,7238	0,3608	261	116
2	month	2,3614	2,1928	0,9036	0,8675	654,6386	635,8072	0,5263	196	108
3	month	2,8774	2,8774	1,2000	1,2000	654,1226	643,1226	0,4954	181	104
4	month	2,1047	2,0756	0,8605	0,8547	654,8953	640,9244	0,5241	223	119
5	month	2,4074	2,4074	0,8889	0,8889	654,5926	640,5926	0,4200	252	123
6	month	2,2312	2,2151	0,8602	0,8495	654,7688	638,7849	0,3945	259	126
7	month	2,2343	2,2286	0,7429	0,7429	654,7657	644,7714	0,3503	271	124
8	month	2,4388	2,4245	0,7626	0,7626	654,5612	643,5755	0,3175	266	127

Table 6.12: ARM Results - 10-fold cross, **monthly carts**, min confidence=20%, **sup=0.001**, data size=2.5M, **platform="n11.com"**

Part	Period	LHS Found Carts	LHS Found Cust	LHS-RHS Found Carts	LHS-RHS Found Cust	LHS Not Found Carts	LHS Not Found Cust	Precision (Accuracy)	Rules	Itemsets
0	month	15,0000	15,0000	7,7500	7,7500	7124,0000	6965,0000	0,6305	8	4
1	month	16,8750	16,8750	7,2500	7,2500	7122,1250	6994,1250	0,4965	8	4
2	month	17,2500	17,2500	7,7500	7,7500	7121,7500	6976,7500	0,4888	8	4
3	month	14,7000	14,6000	6,6000	6,6000	7124,3000	7006,4000	0,5303	10	5
4	month	22,3333	22,3333	11,3333	11,3333	7116,6667	6989,6667	0,6120	6	3
5	month	17,5000	17,5000	8,3333	8,3333	7121,5000	6963,5000	0,5810	6	3
6	month	18,5000	18,5000	8,3333	8,3333	7119,5000	6989,5000	0,5418	6	3
7	month	13,3750	13,3750	6,2500	6,2500	7124,6250	6984,6250	0,5997	8	4
8	month	17,3333	17,1667	8,6667	8,6667	7120,6667	6994,8333	0,6488	6	3
9	month	16,6000	16,6000	7,8000	7,8000	7121,4000	6990,4000	0,5584	10	5

CHAPTER 7

SEQUENCE PATTERN MINING (SPM) ON THE DATA

7.1 Introduction

In chapter **Association Rule Mining on the Data**, associativity of any two products are found according to their *support*, *confidence*, *lift* values. These metrics are actually **time-independent**. This means that if products p_1 and p_2 are found to be related, they will be recommended to customers who buy one of them at any time.

On the other hand, another way of making recommendations is finding frequent sequences on the data and **including time factor to the recommendation system.** With sequential pattern mining (SPM), common customer choices over time can be found. SPM methods not only discover the relationship between products p_1 and p_2, but also they discover the time order for purchasing p_1 and p_2.

Frequent sequences within data are found with *PrefixSpan* algorithm in a way that is explained in [28]. Finally, frequent sequence patterns are evaluated.

7.2 Data Preprocessing for SPM

In *PrefixSpan* algorithm, the problem is defined as "finding frequent transaction sequences from transactions that contain multiple products in them".

In real life, transactions with multiple products are called shopping carts and they contain related, same or completely independent multiple products. On the other hand, most of the shoppings in the data contain only one product, which is previously stated **here**.

With this information in mind, in addition to preprocessing work mentioned in chapter **Data Statistics & Preprocessing**, following work is done:

- Because of the fact that most of the transactions are single purchases, it is difficult to consider these transactions as shopping carts. To overcome this problem, transactions done in the same **time period** are considered as a shopping cart.

  For better understanding, lets consider below simple transactions:

```
t1: p1 on 24 May 2019 by customer c1
t2: p2 on 25 May 2019 by c1
t3: p2 on 26 May 2019 by c2
t4: p3 on 28 May 2019 by c2
t5: p4 on 29 May 2019 by c2
```

- Time period-transaction mapping is constructed. Time period can be **daily**, **weekly** or **monthly** period. Here is an example for weekly mapping:

```
week 1 (20 May-26 May 2019) => t1, t2, t3
week 2 (27 May-2 Jun 2019) => t4, t5
```

- Using time period-transaction mapping, for all distinct days/weeks/months, transactions are separated into smaller groups according to the customer id of the transaction. These final groupings are called shopping carts.

```
cart 1: [p1, p2] by c1
cart 2: [p2] by c2
cart 3: [p3, p4] by c2
```

- When grouping by time period is done, **sequences** are constructed. A sequence consists of shopping carts that have transactions made in the **same year**. Another issue is that even a sequence covers a period of 1 year, there are still sequences with 1-shopping cart. Since these sequences are not suitable for discovering frequent sequence patterns, they are **excluded**:

```
*sequence 1:  [cart 1]         => [<p1,p2>]        <= EXCLUDED
sequence 2:   [cart 2, cart 3] => [<p2>, <p3,p4>]
```

- All sequences are shuffled and partitioned into 10 parts for doing 10-fold cross-validation. For each partition, frequent sequence patterns are searched from the other 9 partitions and evaluated with sequences in the current test partition. Whole procedure can be seen in **Figure 7.1**.

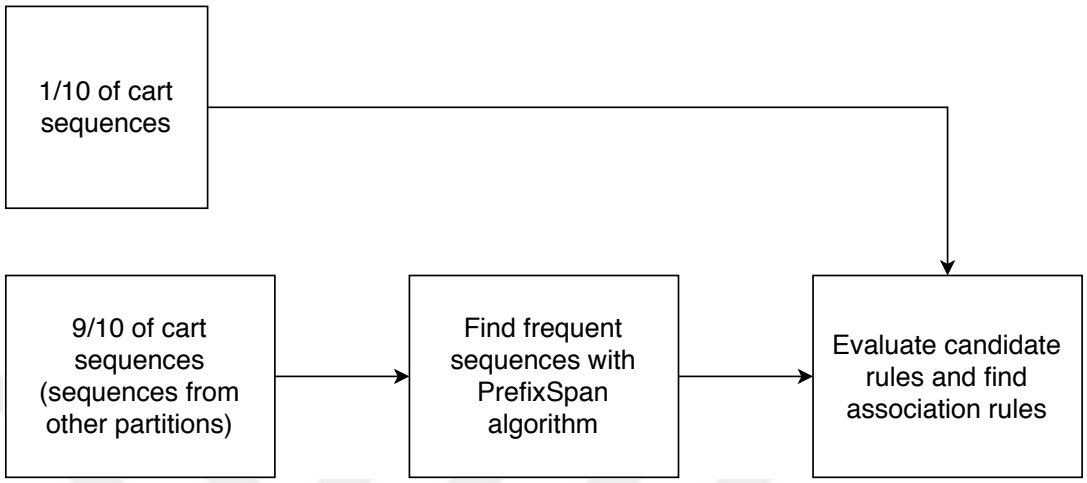


Figure 7.1: SPM Cross-Validation

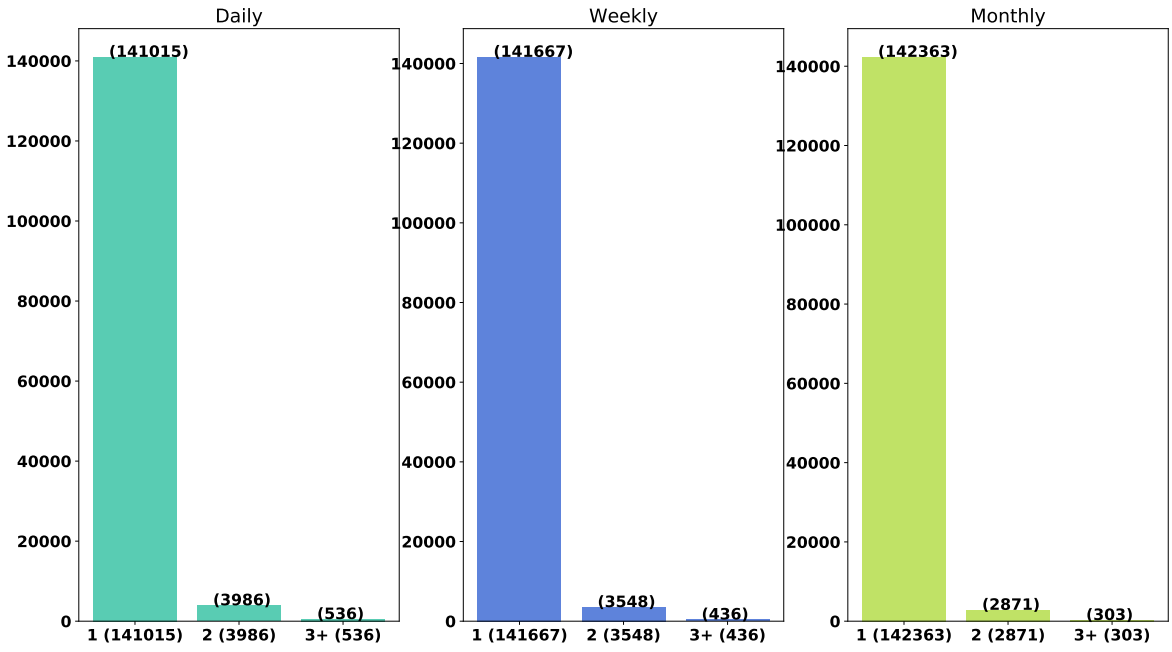


Figure 7.2: Histogram of various sequence lengths - 200K transactions

Figure 7.2 shows the histogram of various sequence lengths 1, 2, 3 or more. For instance, 141015 / 145537 of sequences are excluded from daily sequences and remaining sequences are used for discovering frequent sequences.

7.3 Application of PrefixSpan Algorithm

Frequent sequences are found by applying PrefixSpan algorithm. It starts from 1-cart subsequences, finds sequences that start with a particular 1-cart subsequence and occur more than *support threshold*. After that, it divides search space and expands current sequence length to 2. The same filtering procedure is done again and length is incremented. This goes until no new frequent subsequence is found.

When transactions are grouped by daily, following results are found for different sequence support thresholds:

Table 7.1: SPM - single fold, daily, 200K transactions

Support	Frequent Sequences	Total Sequences
3/4522	63	4522
4/4522	12	4522
5/4522	12	4522
6/4522	6	4522
10/4522	0	4522

When transactions are grouped by weekly, following results are found for different sequence support thresholds:

Table 7.2: SPM - single fold, weekly, 200K transactions

Support	Frequent Sequences	Total Sequences
3/3984	53	3984
4/3984	7	3984
5/3984	7	3984
6/3984	3	3984
10/3984	0	3984

When transactions are grouped by monthly, following results are found for different sequence support thresholds:

Table 7.3: SPM - single fold, monthly, 200K transactions

Support	Frequent Sequences	Total Sequences
3/3174	37	3174
4/3174	15	3174
5/3174	6	3174
6/3174	6	3174
10/3174	0	3174

These values show the number of frequent sequences with respect to various support threshold values. For cross-validation results and more detailed results **SPM Results** section.

7.4 Evaluation of Frequent Sequence Patterns

The evaluation method is different from evaluation methods done for SVD and ARM. Sequences consist of shopping carts and the aim is to find out frequent, successive shopping carts.

When a frequent sequence is found, it is disassembled into subsequences. The possibility of having a specific subsequence is calculated by dividing “the number of times sequence until previous subsequence occurs” to “the number of times sequence until current subsequence occurs”. In **Figure 7.3**, this procedure is explained visually:

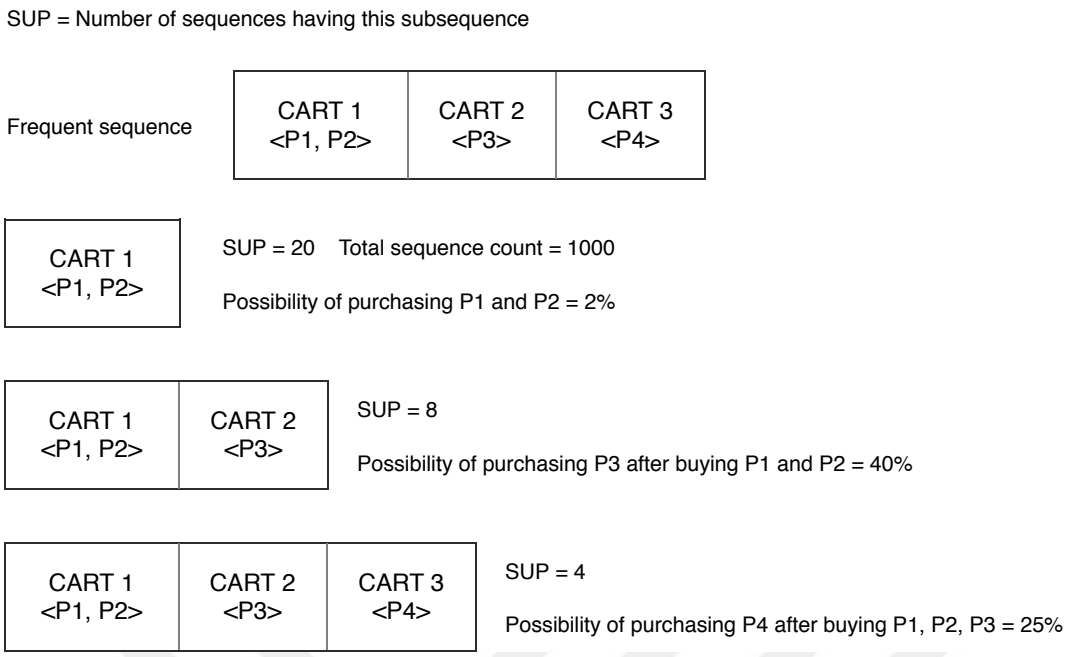


Figure 7.3: Visual explanation for sequence evaluation

With this procedure, the possibility of purchasing a product in a year after a group of other products are purchased can be found. After these possibilities are obtained, they are evaluated with sequences from the test data in the same way. This method can be used in real life to:

- show correct, meaningful advertisements to the customers and appeal them for more shopping [42]
- apply discounts, arrange wholesales
- manage stocks efficiently, increase or decrease the number of a product in the stock

according to frequent sequence patterns discovered from the data.

7.5 SPM Results

Table 7.4: SPM - single fold, momently, data size=200K

Sequence Period Type	Cart Period Type	Prod. Trx. threshold	Num. of Sequences	Support	Num. of Sequence Patterns
Yearly	Momently	3	5419	0,0005536076767	23
Yearly	Momently	4	5419	0,0007381435689	23
Yearly	Momently	5	5419	0,0009226794612	8
Yearly	Momently	6	5419	0,001107215353	8
Yearly	Momently	10	5419	0,001845358922	0

Table 7.5: SPM - single fold, daily, data size=200K

Sequence Period Type	Cart Period Type	Prod. Trx. threshold	Num. of Sequences	Support	Num. of Sequence Patterns
Yearly	Daily	3	4522	0,00066342326404	63
Yearly	Daily	4	4522	0,00088456435206	12
Yearly	Daily	5	4522	0,00110570544007	12
Yearly	Daily	6	4522	0,00132684652808	6
Yearly	Daily	10	4522	0,00221141088014	0

Table 7.6: SPM - single fold, weekly, data size=200K

Sequence Period Type	Cart Period Type	Prod. Trx. threshold	Num. of Sequences	Support	Num. of Sequence Patterns
Yearly	Weekly	3	3984	0,00075301204819	53
Yearly	Weekly	4	3984	0,00100401606426	7
Yearly	Weekly	5	3984	0,00125502008032	7
Yearly	Weekly	6	3984	0,00150602409639	3
Yearly	Weekly	10	3984	0,00251004016064	0

Table 7.7: SPM - single fold, monthly, data size=200K

Sequence Period Type	Cart Period Type	Prod. Trx. threshold	Num. of Sequences	Support	Num. of Sequence Patterns
Yearly	Monthly	3	3174	0,00094517958412	37
Yearly	Monthly	4	3174	0,00126023944549	15
Yearly	Monthly	5	3174	0,00157529930687	6
Yearly	Monthly	6	3174	0,00189035916824	6
Yearly	Monthly	10	3174	0,00315059861374	0

Table 7.8: SPM - 10-fold cross, monthly, yearly sequences, data size=2.5M

Part	Sup=0.001 Accuracy	Sup=0.001 Rec. Success Cust	Sup=0.001 Rec. Cust	Sup=0.001 Sequence Count	Sup=0.0003 Accuracy	Sup=0.0003 Rec. Success Cust	Sup=0.0003 Rec. Cust	Sup=0.0003 Sequence Count
0	0,1732	21,7778	129,4444	9	0,1314	6,6136	68,3750	176
1	0,1336	13,6923	108,4615	13	0,1311	6,3636	66,4830	176
2	0,1287	15,5000	124,3000	10	0,1154	5,5440	69,7306	193
3	0,1461	16,7000	114,3000	10	0,1753	8,2295	67,3552	183
4	0,1481	16,8889	117,8889	9	0,1587	7,1902	65,8043	184
5	0,1275	13,5000	111,5000	10	0,1428	6,8519	68,2804	189
6	0,1429	17,1111	124,2222	9	0,1327	6,6738	70,2086	187
7	0,1410	16,0000	116,3000	10	0,1403	8,6793	69,2500	184
8	0,1646	22,2222	140,8889	9	0,1426	6,8571	71,7143	175
9	0,1300	15,4000	122,2000	10	0,1189	5,9011	70,5440	182

Table 7.9: SPM - 10-fold cross, monthly, yearly sequences, data size=2.5M, **platform="n11.com"**

Part	Sup=0.001 Accuracy	Sup=0.001 Rec. Success Cust	Sup=0.001 Rec. Cust	Sup=0.001 Sequence Count	Sup=0.0003 Accuracy	Sup=0.0003 Rec. Success Cust	Sup=0.0003 Rec. Cust	Sup=0.0003 Sequence Count
0	0,3768	7,7500	21,0000	4	0,3131	3,6429	13,0714	28
1	0,3652	7,0000	21,0000	4	0,2776	2,9000	11,7000	30
2	0,4135	7,0000	17,4000	5	0,2615	2,9394	12,3939	33
3	0,3471	6,4000	20,0000	5	0,3235	3,0333	11,4000	30
4	0,3529	7,6667	22,3333	3	0,3149	3,6207	12,3103	29
5	0,3357	6,8000	21,8000	5	0,3246	3,2143	11,9286	28
6	0,3334	6,8000	20,6000	5	0,2930	3,4839	12,4839	31
7	0,3490	7,4000	24,4000	5	0,3090	3,3591	12,3043	30
8	0,4460	8,5000	21,2500	4	0,3568	3,9286	13,1071	28
9	0,3714	6,6000	19,0000	5	0,3229	3,4688	12,3438	32

Table 7.10: SPM - 10-fold cross, monthly, yearly sequences, data size=2.5M, **platform="epttavm"**

Part	Sup=0.001 Accuracy	Sup=0.001 Rec. Success Cust	Sup=0.001 Rec. Cust	Sup=0.001 Sequence Count
0	0,5612	1,5116	3,4186	43
1	0,6077	1,4118	3,1961	51
2	0,5168	1,3000	3,5000	60
3	0,5372	1,1923	3,2692	52
4	0,6265	1,4068	2,9492	59
5	0,5135	1,5000	4,3148	54
6	0,5669	1,5122	3,4634	41
7	0,5338	1,3793	3,3793	58
8	0,5333	1,4138	3,5517	58
9	0,6740	1,2857	2,5952	42

Table 7.11: SPM - 10-fold cross, weekly, yearly sequences, data size=2.5M

Part	Sup=0.001 Accuracy	Sup=0.001 Rec. Success Cust	Sup=0.001 Rec. Cust	Sup=0.001 Sequence Count	Sup=0.0003 Accuracy	Sup=0.0003 Rec. Success Cust	Sup=0.0003 Rec. Cust	Sup=0.0003 Sequence Count
0	0,1488	18,3333	124,8889	9	0,1458	6,5920	68,7471	174
1	0,1342	18,3750	142,0000	8	0,1322	6,1966	70,2135	180
2	0,1438	21,0000	149,0000	8	0,1255	6,6201	74,8324	180
3	0,1530	19,1250	130,0000	8	0,1280	6,5086	71,9429	176
4	0,1196	14,8000	124,9000	10	0,1217	6,5824	74,6000	171
5	0,1359	19,0000	142,5000	8	0,1338	6,3663	73,3314	172
6	0,1774	25,0000	141,8889	9	0,1447	7,9405	75,8036	168
7	0,1642	23,6667	147,1111	9	0,1432	7,9941	78,9053	172
8	0,1448	18,5556	134,4444	9	0,1248	6,5941	73,8059	170
9	0,1769	23,6667	135,2222	9	0,1359	7,0240	74,7246	174

Table 7.12: SPM - 10-fold cross, weekly, monthly sequences, data size=2.5M

Part	**Sup=0.001 Accuracy**	**Sup=0.001 Rec. Success Cust**	**Sup=0.001 Rec. Cust**	**Sup=0.001 Sequence Count**	**Sup=0.0003 Accuracy**	**Sup=0.0003 Rec. Success Cust**	**Sup=0.0003 Rec. Cust**	**Sup=0.0003 Sequence Count**
0	0,0000	0	0	0	0,3304	1,3000	8,0500	40
1	0,1428	3	21	1	0,2079	1,5370	11,5926	54
2	0,0000	0	0	0	0,2641	1,8939	9,4394	66
3	0,0000	0	0	0	0,2872	1,2791	8,0930	43
4	0,0952	2	21	1	0,2226	1,6441	12,7797	61
5	0,1153	3	26	1	0,2401	1,8305	10,5932	59
6	0,0000	0	0	0	0,2527	2,0270	11,3514	74
7	0,1875	3	16	1	0,2397	1,4912	9,7719	57
8	0,1200	3	25	1	0,2942	2,0615	11,4000	65
9	0,0769	1	13	2	0,2786	1,3265	8,0612	49

Table 7.13: SPM - Example sequences, 10-fold cross, monthly, data size=2.5M, support=0.001

Part	Frequent Sequence	Accuracy	Status
0	((49583, 49583), (49583,))	0,2566	29/113
0	((49583, 49583), (49583, 49583))	0,1947	22/113
0	((49583,), (49583,))	0,1973	29/147
0	((49583,), (49583, 49583))	0,1497	22/147
0	((315947, 315947), (315947,))	0,1680	21/125
0	((315947, 315947), (315947, 315947))	0,1520	19/125
0	((315947,), (315947,))	0,1304	21/161
0	((315947,), (315947, 315947))	0,1180	19/161
0	((108771,), (108771,))	0,1918	14/73
1	((49583, 49583), (49583,))	0,1780	21/118
1	((49583, 49583), (49583, 49583))	0,1610	19/118
1	((49583,), (315947,))	0,0338	5/148
1	((49583,), (49583,))	0,1419	21/148
1	((49583,), (49583, 49583))	0,1284	19/148
1	((315947, 315947), (315947,))	0,1635	17/104
1	((315947, 315947), (315947, 315947))	0,1442	15/104
1	((315947,), (315947,))	0,1189	17/143
1	((315947,), (315947, 315947))	0,1049	15/143
1	((72025, 72025), (72025,))	0,0645	4/62
1	((72025,), (72025,))	0,0633	5/79
1	((108771, 108771), (108771,))	0,2564	10/39
1	((108771,), (108771,))	0,1786	10/56
2	((49583, 49583), (49583,))	0,1983	23/116
2	((49583, 49583), (49583, 49583))	0,1810	21/116
2	((49583,), (315947,))	0,0387	6/155
2	((49583,), (49583,))	0,1484	23/155
2	((49583,), (49583, 49583))	0,1355	21/155
2	((315947, 315947), (315947,))	0,1188	12/101
2	((315947, 315947), (315947, 315947))	0,1188	12/101
2	((315947,), (315947,))	0,0902	12/133
2	((315947,), (315947, 315947))	0,0902	12/133
2	((108771,), (108771,))	0,1667	13/78
3	((49583, 49583), (49583,))	0,1923	20/104
3	((49583, 49583), (49583, 49583))	0,1827	19/104
3	((49583,), (49583,))	0,1549	22/142
3	((49583,), (49583, 49583))	0,1338	19/142
3	((315947, 315947), (315947,))	0,1593	18/113
3	((315947, 315947), (315947, 315947))	0,1593	18/113
3	((315947,), (315947,))	0,1284	19/148
3	((315947,), (315947, 315947))	0,1216	18/148
3	((108771, 108771), (108771,))	0,1400	7/50
3	((108771,), (108771,))	0,0886	7/79

Table 7.14: SPM - Example sequences, 10-fold cross, monthly, data size=2.5M, support=0.001

Part	Frequent Sequence	Accuracy	Status
4	((49583, 49583), (49583,))	0,1667	17/102
4	((49583, 49583), (49583, 49583))	0,1373	14/102
4	((49583,), (49583,))	0,1318	17/129
4	((49583,), (49583, 49583))	0,1085	14/129
4	((315947, 315947), (315947,))	0,1739	20/115
4	((315947, 315947), (315947, 315947))	0,1652	19/115
4	((315947,), (315947,))	0,1299	20/154
4	((315947,), (315947, 315947))	0,1234	19/154
4	((108771,), (108771,))	0,1967	12/61
5	((49583, 49583), (49583,))	0,1489	14/94
5	((49583, 49583), (49583, 49583))	0,1064	10/94
5	((49583,), (49583,))	0,1203	16/133
5	((49583,), (49583, 49583))	0,0752	10/133
5	((315947, 315947), (315947,))	0,1545	17/110
5	((315947, 315947), (315947, 315947))	0,1455	16/110
5	((315947,), (315947,))	0,1063	17/160
5	((315947,), (315947, 315947))	0,1000	16/160
5	((108771, 108771), (108771,))	0,1731	9/52
5	((108771,), (108771,))	0,1449	10/69
6	((49583, 49583), (49583,))	0,1468	16/109
6	((49583, 49583), (49583, 49583))	0,1284	14/109
6	((49583,), (49583,))	0,1156	17/147
6	((49583,), (49583, 49583))	0,0952	14/147
6	((315947, 315947), (315947,))	0,1905	20/105
6	((315947, 315947), (315947, 315947))	0,1714	18/105
6	((315947,), (315947,))	0,1355	21/155
6	((315947,), (315947, 315947))	0,1161	18/155
6	((108771,), (108771,))	0,1860	16/86
7	((49583, 49583), (49583,))	0,2069	24/116
7	((49583, 49583), (49583, 49583))	0,1724	20/116
7	((49583,), (49583,))	0,1622	24/148
7	((49583,), (49583, 49583))	0,1351	20/148
7	((315947, 315947), (315947,))	0,1391	16/115
7	((315947, 315947), (315947, 315947))	0,1130	13/115
7	((315947,), (315947,))	0,1060	16/151
7	((315947,), (315947, 315947))	0,0861	13/151
7	((108771, 108771), (108771,))	0,1795	7/39
7	((108771,), (108771,))	0,1094	7/64

Table 7.15: SPM - 10-fold cross, monthly, data size=2.5M, support=0.001

Part	Frequent Sequence	Accuracy	Status
8	((49583, 49583), (49583,))	0,2197	29/132
8	((49583, 49583), (49583, 49583))	0,1970	26/132
8	((49583,), (49583,))	0,1696	29/171
8	((49583,), (49583, 49583))	0,1520	26/171
8	((315947, 315947), (315947,))	0,1654	21/127
8	((315947, 315947), (315947, 315947))	0,1417	18/127
8	((315947,), (315947,))	0,1236	22/178
8	((315947,), (315947, 315947))	0,1011	18/178
8	((108771,), (108771,))	0,2115	11/52
9	((49583, 49583), (49583,))	0,1570	19/121
9	((49583, 49583), (49583, 49583))	0,1570	19/121
9	((49583,), (49583,))	0,1138	19/167
9	((49583,), (49583, 49583))	0,1138	19/167
9	((315947, 315947), (315947,))	0,1569	16/102
9	((315947, 315947), (315947, 315947))	0,1275	13/102
9	((315947,), (315947,))	0,1133	17/150
9	((315947,), (315947, 315947))	0,0867	13/150
9	((72025,), (72025,))	0,0921	7/76
9	((108771,), (108771,))	0,1818	12/66

Table 7.16: Examples of monthly frequent sequences with support=0.001

POPULAR SEQUENCE: ((49583, 49583), (49583,)) 49583 - bb all in one krem aciktan ortaya-50 ml (makyaj|yuz) Subseq. len = 1: 113 customers Subseq. len = 2: 29 customers - 25.66% (29/113)
POPULAR SEQUENCE: ((315947,), (315947, 315947)) 315947 - vfx pro camera ready makyaj bazi-20 ml (makyaj|yuz) Subseq. len = 1: 161 customers Subseq. len = 2: 19 customers - 11.80% (19/161)
POPULAR SEQUENCE: ((108771,), (108771,)) 108771 - eurofresh misvakli beyazlatici dis macunu-112gr (kisisel bakim|agiz bakim) Subseq. len = 1: 73 customers Subseq. len = 2: 14 customers - 19.18% (14/73)

CHAPTER 8

CONCLUSIONS

In this thesis work, we inspected some of the data mining methods and latent semantic analysis methods, to present a better recommender system for e-commerce applications. These methods are mainly singular value decomposition (SVD) for latent semantic analysis (LSA), association rule mining (ARM) and sequential pattern mining (SPM) methods.

Applying these methods to real data was a big challenge. The number of customers was high, this is due to the fact that a customer is defined with its platform name, customer nickname, customer full name, customer city, customer phone columns. The customer-product matrix was very sparse, since most of the shopping carts contain only one item. To apply the above methods, we need to increase the density of customer-product matrix by filtering out unpopular items and customers with a small number of shopping experiences.

SVD is a good way of reducing the dimensions of the data and a good way of extracting context from the data. By doing it on transactional data, one can develop a good recommender system. Considering our results, SVD performed better for finding product-to-product similarities than finding customer-to-customer similarities when data size is small. When data size was increased, the model performed better for the customer-to-customer recommendation. We observed that as the data size increases, it becomes easier to find similar customers and easier to find groupings of customers.

ARM is also beneficial for recommending products to users. After association rules are found, a recommender system can benefit from those rules and increase its recommendation accuracy. We observed that when association rules are found within a

specific platform (such as n11.com), their precision (accuracy) on the test partition is higher than the rules extracted using whole data.

For SPM, accuracy results on the test partition were not high as ARM. This may be due to the fact that the percentage of 1-cart sequences is almost 90% of all shopping experiences. When these are ignored, frequent sequences are found but they are mainly sequences of the same products in continuous carts. If the data has enough distribution considering the size of cart sequences, the recommendation system can perform better. Our experiments showed that a recommender system can detect which product is going to be repeatedly purchased. These products are mainly from cosmetics, pet foods, electrical components such as cables, diapers, etc. This can be useful to provide discounts on such products for customers and make customers spend money earlier than usual.

As future work, we can investigate whether better recommendations can be provided or not by extracting association rules from transactions within the same region or from the transactions within the same time period. This time period can be Valentine's Day, Mother's Day, Father's Day, Black Friday, etc. If shopping behavior is extracted from the special days, the profit that the recommender system provides will be better.

As another future work, hierarchical categories can be converted to tags and recommendations can be done using these tags. We can say two products are strongly related if both of them contain similar category tags. Since product category parts in the transactions are not ordered from high level to low level, which is a brand-new task to accomplish.

REFERENCES

[1] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2000). Application of dimensionality reduction in recommender system-a case study (No. TR-00-043). Minnesota Univ Minneapolis Dept of Computer Science.

[2] Salton, G., Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information processing management*, 24(5), 513-523.

[3] Sarwar, B. M., Karypis, G., Konstan, J. A., Riedl, J. (2001). Item-based collaborative filtering recommendation algorithms. Www, 1, 285-295.

[4] Linden, G., Smith, B., York, J. (2003). Amazon. com recommendations: Item-to-item collaborative filtering. IEEE Internet computing, (1), 76-80.

[5] Jacobi, J. A., Benson, E. A., Linden, G. D. (2001). *U.S. Patent No. 6,317,722*. Washington, DC: U.S. Patent and Trademark Office.

[6] Wu, L., Shah, S., Choi, S., Tiwari, M., Posse, C. (2014, October). The Browsemaps: Collaborative Filtering at LinkedIn. In RSWeb@ RecSys.

[7] Lam, X. N., Vu, T., Le, T. D., Duong, A. D. (2008, January). Addressing cold-start problem in recommendation systems. In Proceedings of the 2nd international conference on Ubiquitous information management and communication (pp. 208-211). ACM.

[8] Pazzani, M. J. (1999). A framework for collaborative, content-based and demographic filtering. Artificial intelligence review, 13(5-6), 393-408.

[9] Blum, A. (1997). Empirical support for winnow and weighted-majority algorithms: Results on a calendar scheduling domain. Machine Learning, 26(1), 5-23.

[10] Abbassi, Z., Amer-Yahia, S., Lakshmanan, L. V., Vassilvitskii, S., Yu, C. (2009, October). Getting recommender systems to think outside the box. In Proceed-

ings of the third ACM conference on Recommender systems (pp. 285-288). ACM.

[11] Basilico, J., Hofmann, T. (2004, July). Unifying collaborative and content-based filtering. In Proceedings of the twenty-first international conference on Machine learning (p. 9). ACM.

[12] Claypool, M., Gokhale, A., Miranda, T., Murnikov, P., Netes, D., Sartin, M. (1999). Combing content-based and collaborative filters in an online newspaper.

[13] Balabanović, M., Shoham, Y. (1997). Fab: content-based, collaborative recommendation. Communications of the ACM, 40(3), 66-72.

[14] Asuero, A. G., Sayago, A., Gonzalez, A. G. (2006). The correlation coefficient: An overview. Critical reviews in analytical chemistry, 36(1), 41-59.

[15] Davis, P. S., Harveston, P. D. (2000). Internationalization and organizational growth: The impact of Internet usage and technology involvement among entrepreneur-led family businesses. Family Business Review, 13(2), 107-120.

[16] Reinsel, D., Gantz, J., Rydning, J. (2018). The digitization of the world: from edge to core. IDC White Paper Doc US44413318. Viewed March.

[17] Zikopoulos, P., Eaton, C. (2011). Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

[18] Frawley, W. J., Piatetsky-Shapiro, G., Matheus, C. J. (1992). Knowledge discovery in databases: An overview. AI magazine, 13(3), 57-57.

[19] Agrawal, R., Imieliński, T., Swami, A. (1993, June). Mining association rules between sets of items in large databases. In *Acm sigmod record* (Vol. 22, No. 2, pp. 207-216). ACM.

[20] Agrawal, R., Srikant, R. (1994, September). Fast algorithms for mining association rules. In Proc. 20th int. conf. very large data bases, VLDB (Vol. 1215, pp. 487-499).

[21] Vaidya, J., Clifton, C. (2002, July). Privacy preserving association rule mining in vertically partitioned data. In Proceedings of the eighth ACM SIGKDD in-

ternational conference on Knowledge discovery and data mining (pp. 639-644). ACM.

[22] Tao, F., Murtagh, F., Farid, M. (2003, August). Weighted association rule mining using weighted support and significance framework. In Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 661-666). ACM.

[23] Liu, B., Hsu, W., Ma, Y. (1999, August). Mining association rules with multiple minimum supports. In Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 337-341). ACM.

[24] Agrawal, R., Srikant, R. (1995, March). Mining sequential patterns. In icde (Vol. 95, pp. 3-14).

[25] Abouelhoda, M., Ghanem, M. (2009). String mining in bioinformatics. In Scientific Data Mining and Knowledge Discovery (pp. 207-247). Springer, Berlin, Heidelberg.

[26] Ayres, J., Flannick, J., Gehrke, J., Yiu, T. (2002, July). Sequential pattern mining using a bitmap representation. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 429-435). ACM.

[27] Tarjan, R. (1972). Depth-first search and linear graph algorithms. SIAM journal on computing, 1(2), 146-160.

[28] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M. C. (2001). Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. In *Proceedings 17th international conference on data engineering* (pp. 215-224). IEEE.

[29] Landauer, T. K., Foltz, P. W., Laham, D. (1998). An introduction to latent semantic analysis. Discourse processes, 25(2-3), 259-284.

[30] De Lathauwer, L., De Moor, B., Vandewalle, J. (2000). A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications, 21*(4), 1253-1278.

[31] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., Harshman, R. (1990). Indexing by latent semantic analysis. Journal of the American society for information science, 41(6), 391-407.

[32] Hofmann, T. (1999, July). Probabilistic latent semantic analysis. In Proceedings of the Fifteenth conference on Uncertainty in artificial intelligence (pp. 289-296). Morgan Kaufmann Publishers Inc..

[33] Srebro, N., Jaakkola, T. (2003). Weighted low-rank approximations. In Proceedings of the 20th International Conference on Machine Learning (ICML-03) (pp. 720-727).

[34] Landauer, T. K., Dumais, S. T. (1997). A solution to Plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological review, 104(2), 211.

[35] Behrens, C. A., Egan, D. E., Ho, Y. Y., Lochbaum, C., Rosenstein, M. (2003). U.S. Patent No. 6,615,208. Washington, DC: U.S. Patent and Trademark Office.

[36] Blum, A., Hopcroft, J., Kannan, R. (2016). Foundations of data science. Vorab-version eines Lehrbuchs.

[37] Berry, M. W., Dumais, S. T., O'Brien, G. W. (1995). Using linear algebra for intelligent information retrieval. SIAM review, 37(4), 573-595.

[38] Bryt, O., Elad, M. (2008). Compression of facial images using the K-SVD algorithm. Journal of Visual Communication and Image Representation, 19(4), 270-282.

[39] Sarwar, B., Karypis, G., Konstan, J., Riedl, J. (2002, December). Incremental singular value decomposition algorithms for highly scalable recommender systems. In Fifth international conference on computer and information science (Vol. 27, p. 28).

[40] Herlocker, J. L., Konstan, J. A., Terveen, L. G., Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), 5-53.

[41] Bholowalia, P., Kumar, A. (2014). EBK-means: A clustering technique based on elbow method and k-means in WSN. *International Journal of Computer Applications, 105*(9).

[42] Fayyad, U., Piatetsky-Shapiro, G., Smyth, P. (1996). From data mining to knowledge discovery in databases. AI magazine, 17(3), 37-37.