MEF UNIVERSITY

BENCHMARKING OF RECOMMENDATION MODELS FOR AN ON-LINE FAST FASHION RETAILER

Capstone Project

Mustafa TİLKAT

İSTANBUL, 2018

MEF UNIVERSITY

BENCHMARKING OF RECOMMENDATION MODELS FOR AN ON-LINE FAST FASHION RETAILER

Capstone Project

Mustafa Tilkat

Advisor: Hande KÜÇÜKAYDIN

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Benchmarking of Recommendation Models for an On-line Fast Fashion Retailer
Name/Last Name of the Student: Mustafa TİLKAT
Date of Thesis Defense: 10/09/2018

I hereby state that the graduation project prepared by Your Name (Title Format) has been completed under my supervision. I accept this work as a "Graduation Project".

10/09/2018
Hande KÜÇÜKAYDIN(Asst. Prof)

I hereby state that I have examined this graduation project by Mustafa Tilkat which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

10/09/2018

Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of __________ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Hande KÜÇÜKAYDIN	………………………..
2. ………………………..	………………………..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name	Date	Signature
Mustafa Tilkat	10/09/2018	

EXECUTIVE SUMMARY

BENCHMARKING OF RECOMMENDATION MODELS FOR AN ON-LINE FAST FASHION RETAILER

Mustafa TİLKAT

Advisor: Hande KÜÇÜKAYDIN

SEPTEMBER, 2018, 54 pages

This project studies the usage of the recommendation engines to improve the sales in an online fashion retailer. Fashion retailers sale variety of products throughout their online channels. Since the number of products can be huge compared to an in-line shop, customers may miss some of them while shopping online. Hence, it is crucial to display products that are more likely to be purchased by a customer when the customer is surfing on the website. Our problem is motivated by practice at an online fashion retailer in Turkey. Four collaborative filtering-based algorithms and a random recommender are utilized to design a recommendation engine. 80% of the data is used for training while the other 20% is to used test the designed method. Based on our experiments, User Based Collaborative Filtering (UBCF) using Pearson correlation outperform the other algorithms based on Receiver Operating Characteristic (ROC) curve.

Key Words: Online Fashion Retailer, Recommendation engine, User Based Collaborative Filtering, Item Based Collaborative Filtering, Collaborative Filtering.

ÖZET

BİR ONLINE HAZIR GİYİM MODA PERAKENDECİSİNDE ÖNERİ SİSTEMLERİNİN KARŞILAŞTIRILMASI

Mustafa TİLKAT

Tez Danışmanı:Hande KÜÇÜKAYDIN

EYLÜL, 2018, 54 sayfa

Bu projede, bir online moda perakendecisinde satışları iyileştirmek için öneri sistemlerinin nasıl uygulanacağı anlatılmıştır. Moda perakendecilerinin online kanallarında ürün çeşitliliği oldukça fazla olabilmektedir. Ürünlerin sayısı normal bir mağazayla karşılaştırıldığında çok büyük olabileceğinden, müşteriler online alışveriş yaparken bazı modelleri gözden kaçırabilmekte veya aradıkları ürünleri kolayca bulamayabilmektedirler. Bu nedenle, müşteri bir web sitesinde gezinirken bir müşteri tarafından satın alınma olasılığı daha yüksek olan ürünleri müşteriye sunabilme kabiliyeti oldukça önemlidir. Problemimiz, Türkiye'de bir online moda perakendecisi dataları üzerinde uygulama yaparak tatmin edici sonuçlar bulmak üzerine motive edilmiştir. Bir öneri motoru tasarlamak için dört farklı işbirlikçi filtreleme (Collaborative filtering) tabanlı algoritma ve rastgele çeşitli öneriler sunabilecek arı bir baz model kullanılmaktadır. Verilerin% 80'i eğitim seti,% 20'si ise tasarlanan yöntemi test etmek için kullanılmıştır. Deneylerimize dayanarak, Pearson korelasyonunu kullanan Kullanıcı Tabanlı İşbirlikli Filtreleme (User Based Collaborative Filtering) modelinin, ROC eğrisine bakıldığında diğer algoritmalara göre daha iyi bir performans ortaya koyduğu gözlemlenmiştir.

Anahtar Kelimeler: İşbirlikçi Filtreleme, Collaborative Filtering, Online Moda Perakendesi, Öneri Sistemleri

TABLE OF CONTENTS

.

1. INTRODUCTION

Recommendation engine is becoming very popular in many domains specifically in online shops like Amazon, Facebook, Google and ect (Suchal and Navrat 2010). Recommendation systems are based on web usage mining to obtain meaningful knowledge about user to use this knowledge in order to respond to the user's interests (Nasraoui and Petenes 2003).

Characteristics of the items that a user likes determine the items that a user may like, or other users' likes and dislikes. Similarity index between users and recommend items are computed according to them.

Both methods can be combined in order to create more robust recommendation engine. On the other hand, it is fundamental to select an algorithm that fits the problem.

Today, as people emphasis on fashion, outfit select difficulty became a serious problem in daily life and as massive amounts of fashion items are available in markets and online, needs for efficient recommendation services has grown significantly. Recommender systems bolster clients in customized way for the identification of product based on the history of the user. In on-line retail sector, since customer can only see a limited number of products on the screen, it is crucial to recommend products which fit the customers the best. Since huge data is available for any given customer (e.g., which products s/he visited, how long s/he spent on each page, prices, etc), it is desirable for the retailers to recommend products to customers. Currently, this retailer uses rule of thumb to find the best solution for the recommendation engine. However, this solution can be improved by the state of art available in the literature.

Because of the difficulty to dress people on-line with their lifestyles and budgets and to make them feel good, recommending new products is a very challenging category for fast fashion retailing. Every retailer aims to increase the amount of products it sells and increase its profits.

In off-line, retailers can make product recommendations through sales representatives (sometimes it can be a disadvantage) but in on-line there is no possibility to do. Even there is no possibility to use sales representatives, there is a huge amount of data to use on on-line retail which is a very strong advantage to use. So in on-line retail every customer’s historical sales are being tracked and with using this data retailers can recommend their products with helping of the very powerful algorithms.

To solve the recommendation problem on on-line retail, one of most powerful model called collaborative filtering mothods which is used by Amazon and Netflix, utilized to reach more accurate results for the recommendation engine which can company use it on their current online site. In this project this method is utilized to obtain a solution for the recommendation engine

1.1. About the Company

The fashion retailer whose data we use for the project aims to consolidate its leading position within apparel and footwear specialist retailers and to become one of the three most successful apparel and footwear specialist retailers across Europe, through new outlet openings and increasing its total sales area through the opening of new, larger flagship outlets within Turkey. By 2023 the company plans to reach turnover of 10 billion USD, to reach 500 outlets in Turkey and 1,000 outlets outside of Turkey, with an investment of 2 billion USD.

This retailer launched its online shop in March 2011, and since then sales through its online store have been registering significant growth, as the company heavily invested in service improvements such as payment and delivery systems. The company invested in its online operations to supplement its store-based sales, and also to bolster its overall sales. The company provides over 10,000 products through its online shopping channel. The company opened its mobile online store in 2013, which also supports its sales through non-store retailing.

This fashion retailer continued to actively use social media in 2016. The company has 115,000 followers on Twitter and 485,000 followers on Instagram. The company is also highly popular on Facebook, which is the most popular social media platform in the country. Many low- and middle-income consumers use Facebook extensively, and also constitute the main target segments for the company.

1.2. Literature Review

In the literature, there are continuously growing number of papers which use different techniques in different places for different purposes. Whilst reviewing the literature, we have used some papers which is to mention fundamental features of recommender systems. In this part of my research, two papers drew my attention. The first one explains and shows the stages of developing a recommender system and demonstrates technical sides of them which are useful for stronger understanding of the system design (Isinkaye, Folajimi, & Ojokoh, 2015).

Secondly, we intended to read a paper and benefit from it in order to see how the information we get from the first one can be implemented on and to notice the differences between items-based system like first paper referred and user-based system analyzed in the latter. In the second paper, social network recommender system design is evaluated to present calculations for generalized models (Hemant Kumar & Jeysree, 2014). In the paper, Kumar et al. (2014) modelled two different system and these models are deployed a specific social network website as a medium, namely Facebook. In the study, to determine relationships, they defined two different trust metrics, global trust metrics and local trust metrics. By local trust metrics, researchers purposed to find out the close friends of the person, who is the target and they are called as ego person in the paper. Why the authors aimed to figure out the people's closeness to the ego is that if ego's preferences and close relations' are learnt then a super recommender system can be developed since the factors which affect the ego's preferences and habits may be much more revealed. First trust calculation is based on the rate of ego user and friends' of his/her activity on ego's activity. The other is based on friends' activities and users' activities ratio.

When calculations and formulations are provided, they targeted to determine best criteria to understand achievement of study and their total set is combination of four:

"1. A: Set of all recommendations on social networks S_{ar}: this is the set of the all product and services which is liked by users on social networks frequently.

2. B: Set of products liked in closed group S_{g1}: This is the product and services which is being liked frequently by the ego user's friends group

3. C: Set of products liked by closest friend S_{cl}: These are the product and services which are recommended by ego user's close friends.

4. D: Set of ego user's preferences S_{el}: These are the products and services which the ego user like most."

(Hemant Kumar & Jeysree, 2014)

They concluded the paper by mentioning two main problems, problem of trust metrics calculation of a person who is not directly connected to ego user, and how to make recommender system more efficient if there are conflicting facts.

After a general aspect and a case study are inspected, we got our attention onto usage of recent improvements. As an example of usage of technology and predict tech consumers' behaviors, first of all, the paper which investigates Netflix's contest on developing a recommendation system to recommend users the movie they mostly like (Hallinan & Striphas, 2016). In this paper, they analyzed the novel contest in tech culture the conceptual and semantic work required to render algorithmic information processing systems, as they stated. To dive deeper into recommender use with novel technologies, we reviewed a paper named as "Deep Learning based Recommender System: A Survey and New Perspectives" (Shuai Zhang, 2017) which aims to provide a comprehensive review of recent research efforts on deep learning-based recommender systems towards fostering innovations of recommender system research. In this paper, authors recognized newer artificial neural network architectures for recommender systems.

At the end of reviewing, we studied on a paper in which mathematical formulations and approaches evaluated. In this earlier study, authors try to improve performance of RSs by dimensionality reduction techniques (Badrul, George, Joseph, & Riedl, 2000). They used main techniques to reduce dimensions like Singular Value Decomposition or Principal Component Analysis and used basic metrics to calculate success of these methods. The paper argues that SVD is a useful technique to reduce the filtering systems. The use of dimension reduction needed because of three limitations of systems developed for raw data, sparsity, scalability and synonymy of data and implementation. Sparsity occurs because of to find a relationship between two customers, at least two products are necessary in common. Since more recent recommender systems are applied to commercial areas which have large domain of products. In practice, most of data becomes sparse as data grows. Therefore, to avoid such problem, dimension reduction is useful technique. Scalability is a problem due to as data get larger as it needs more computational power and to reduce number of data is also a cure for this problem. The meaning of synonymy in the paper is mostly refers to product groups which have different names and same product groups. For synonymy, they used Latent Semantic Indexing and then paper demonstrates how dimension reduction is implemented. They have used Singular Value Decomposition to have less sparse and scaled data of customer-items matrix and then data is ordered according to the most likely to be purchased by a specific user and N of them are selected as best. At the rest of method’s explanations are metrics they have used.

Even though that most of SVD methods implemented on filtering systems were worse than more traditionally methods, they claim that if it is implemented on dense data, it can be useful and much faster than others.

As already mentioned, there is numerous works on the RSs, however we tried to get deeper understanding of fundamental sides, some popular use cases those became efficient and recognize novel technologies.

2. PROJECT DEFINITION

In this section, the objective and scope of the project will be discussed by highlighting the business priorities.

2.1. Project Definition

With the popularization of e-commerce, every online retailer is aiming to keep its customers shopping on their own channels. With increasing competition, online retailers are having difficulty in providing customer loyalty due to both annual growth and annual turnover targets. Especially in fast fashion retail, it can be twice as difficult to provide customer loyalty. The most important reason why it is difficult to provide customer loyalty in fast fashion retail is that the product variety is very high. Instead of spending most of their time searching for products, customers often want to find the product they are looking for quickly. For this reason, the ability of to offer the product which customers looking for from among many products is a key point for online retailers to provide a significant return on profit and customer loyalty.

2.2. Project Objective

The objective of the project is to develop a baseline algorithm that recommends the product types for online users. In the meantime, understanding how the existing recommendation algorithms works and measuring their performance is another fundamental objective.

The list of project objectives are as follows:

- Analyzing the dataset provided by an online fast fashion retailer
- Developing a User Based Collaborative Filtering algorithm with default parameters
- Developing an Item Based Collaborative Filtering algorithm with default parameters
- Comparing the Collaborative Filtering algorithms with different distance calculation methods.
- Measuring performance of the algorithms and selecting the best one.

The company does not use a recommendation system currently. Hence, there is not a current success criterion to compare to our solution at the end of the project.

2.2. Project Scope

At the scope of this project, we focus on the question “What kind of product an online retailer should recommends their customers?”. We use item-based and user-based recommendation techniques for generating product type recommendations. Our analysis does not consider all the categories because of the computational time factor. Because of this situation, our dataset has been created with only a single product category to build a baseline recommender system for the company. All the other categories remain as areas of future research.

3. ABOUT THE DATA

At first glance, to see statistical sides and some fundamental patterns and recognize better the data we have, we made an Exploratory Data Analysis (EDA). Since the retailer whose data we use for the project has a tremendous rate, we needed to subsample data and therefore we used different types of apparels which belong to same hierarchal sub-group. In these groups data has differences according to their colors and ranges, briefly range means specialties and features of same type of apparel like whether having lines or points on a shirt. In this EDA, we will introduce the data, then we will demonstrate some fundamental and statistical results and show visualizations in order to model better algorithms and implement them.

Firstly, our data consists of a sub-type of adult women clothing and number of products a customer purchased on a year. Length of data is 769,449 and distinct number of users is 58,663. Moreover, we had 22 different colors and 53 different ranges at all. Most of clients purchased once among this group of products, namely the quantity who bought once is 637,983. Basic statistical numbers of purchase quantities over one are demonstrated below:

Mean = 3.13

Standard Deviation = 10.44

Median = 2

Max = 1089

As seen above, purchase quantities' distribution is highly skewed left hand-side (towards to minimum). For instance, even though the median is two, there were 16 customers who purchased over 500 products. As a result, it is needed to eliminate lots of data by z-test confidence interval selection. Our 95% confidence interval occurred between 0.01 and 2.74 so we extracted other values but in this interval. Figure 1 and Figure 2 shows the plot before and after eliminations.

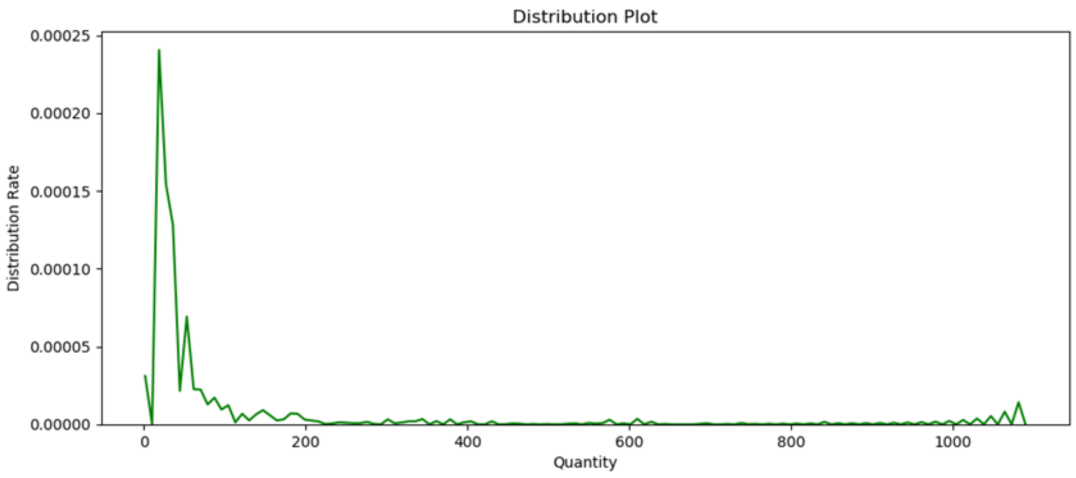


Figure 1: Distribution Plot of Quantity Before Elimination

As graph shows, even if quantity one is eliminated, graph is still left skewed. Thus, we have grouped data by its color and range. Here is the distribution of color-based group.

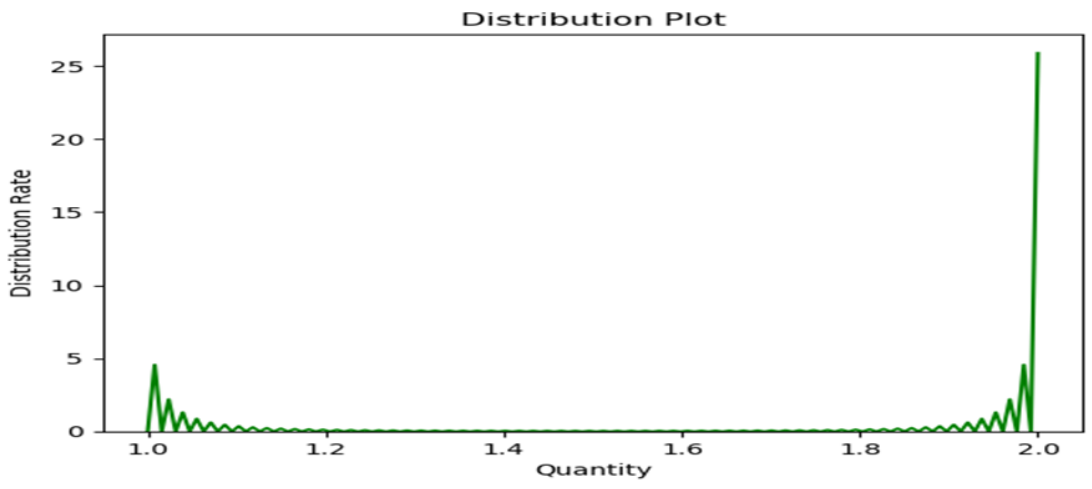


Figure 2: Distribution Plot of Quantity After Elimination

Although it is not a Gaussian Normal Distribution shape plot, the x labels limits are more reasonable shown at Figure 2. At the end, we may easily, however, call this data is not homogenously distributed.

To make further examination over data, we grouped data with two different features, colors and ranges respectively. We will try to show, contrary to whole set, how distribution of data is transformed to more normal.

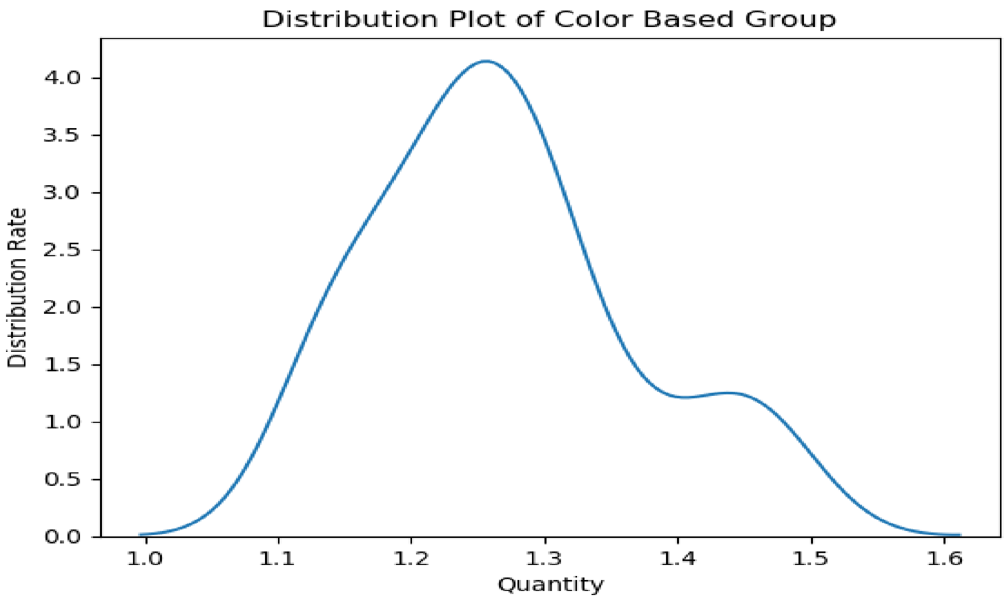


Figure 3: Distribution Plot of Color Based Group Data

After grouping data based on their colors' average purchase quantity, distribution became highly normal as shown by Figure 3.

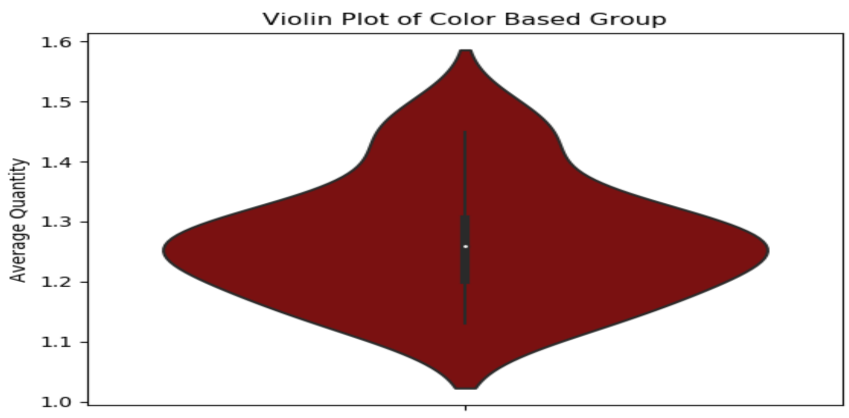


Figure 4: Violin Plot of Color Based Group Data

Violin plot is highly useful technique to show distribution by demonstrating both density and edges of data. Here, we have violin plot of data grouped by its colors. The median of the data is between 1.2 and 1.3 and the density of the quantity data is mostly around the median.

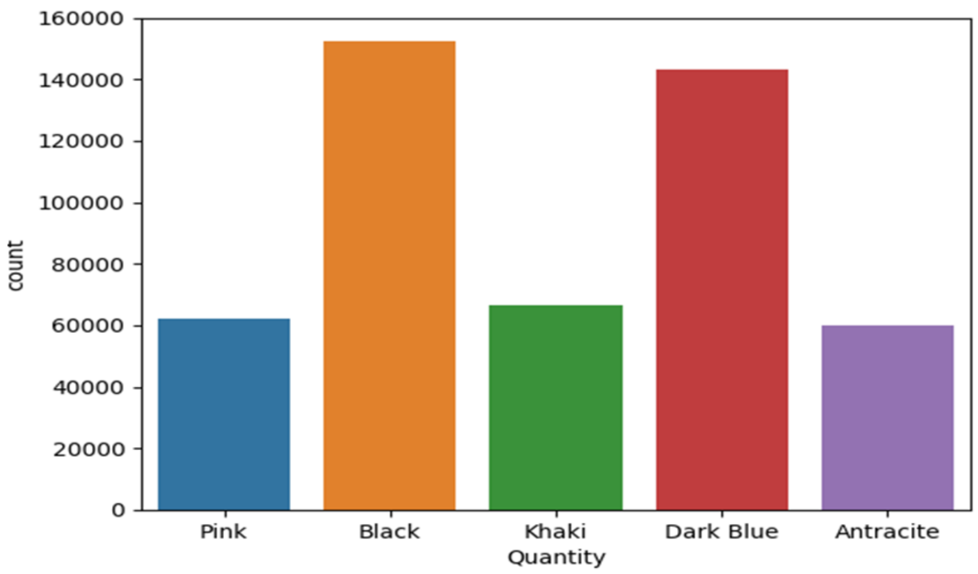


Figure 5: Histogram of Top 5 Colors

In Figure 5, it is shown that some colors are sold much more than others. It is obviously seen that dark colors are preferred more than light colors in top 5 colors.

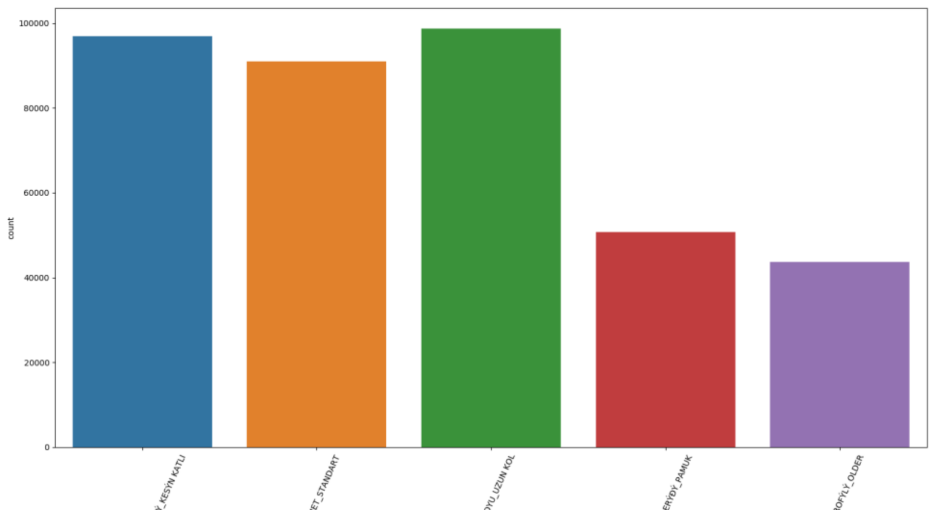


Figure 6: Histogram of Range Criteria

In Figure 6, it is shown that some range criteria are sold much more than others. It is seen that “Kesin Katlı”, “Siluet_Standart” are preferred more than some other criteria.

For further exploration, we may look onto range-based groupings and below, we have related plots.

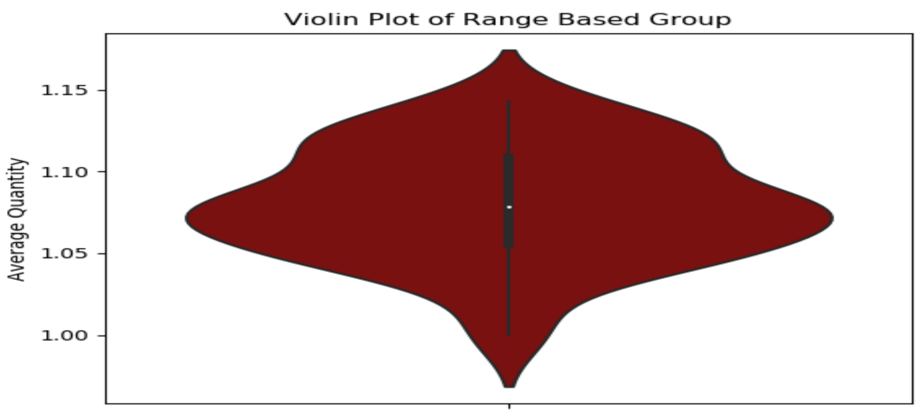


Figure 7: Violin Plot of Quantity (Group by range-based)

The density distribution of range based grouped dataset is shown in Figure 7. The median is between 1.05 and 1.1 and it is seen that dataset become much more normal when it is grouped on range based.

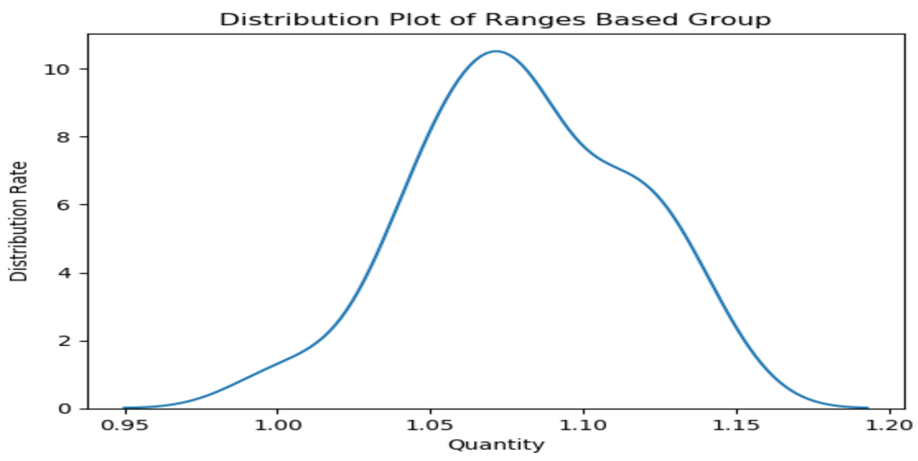


Figure 8: Distribution Plot of Quantity (Group by range-based)

In Figure 8 we can see more statistical and classical way of violin plot and the distribution looks more likely as a normal distribution.

Before user-based plots, we will demonstrate box plots of processed data's over color-based and range-based datasets. To clarify why we could not plotted whole dataset, we need to purport that after rescaling data, we are left over just one and two number of purchases.

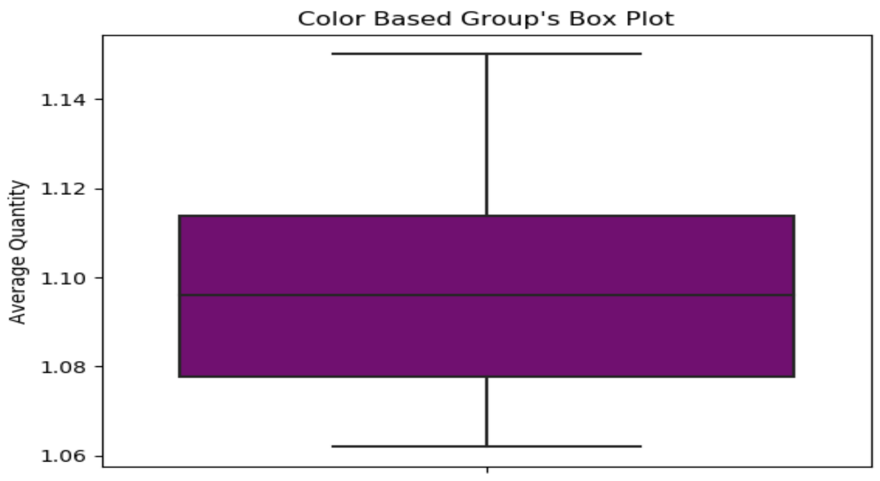


Figure 9: Box Plot of Quantity (Group by color-based)

Box plots are another useful technique to represent and visualize of numerical data and here Figure 9 shows how quartiles distributed over data. The 1st quartile is nearly at 1.08 and median is nearly at 1.1 and the 3rd quartile is nearly at 1.11 this shows that distribution looks like a normal distribution.

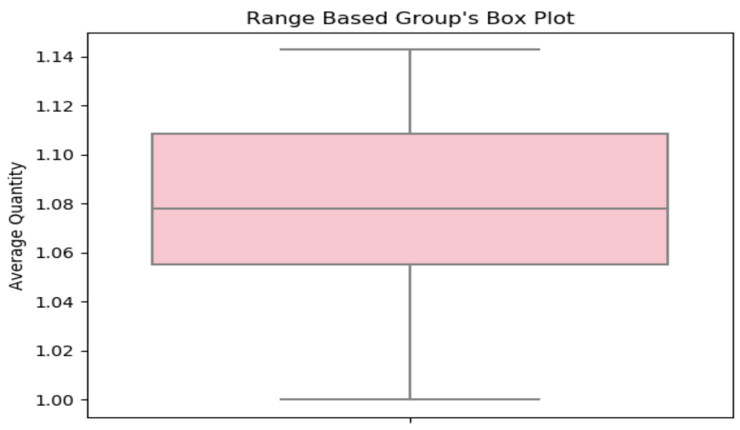


Figure 10: Box Plot of Quantity (Group by range-based)

The box plot with group by range-based demonstrates that the distribution of data more skewed to right side, on the other hand the data which is grouped by color-based skewed to left side.

Lastly, we concentrated onto the data grouped by user-based. We used raw data and deleted outlier after averagely grouped over user ids. Hence, some statistical visualizations and number may differ from former ones.

Like others, grouping is made over average numbers and here some fundamental numbers of the group:

Count: 58,662

Mean: 1.15

Maximum: 80.51

Minimum: 1

Standard Deviation: 0.56

First Three Quartiles: 1

After elimination of outliers by z-test confidential interval, these numbers become:

Count: 57,482

Mean: 1.11

Maximum: 2.32

Minimum: 1

Standard Deviation: 0.25

First Three Quartiles: 1

As seen above, in user-based group, there were many outliers than expected. To be briefer and more understandable, we will only put same graphs made for earlier examples.

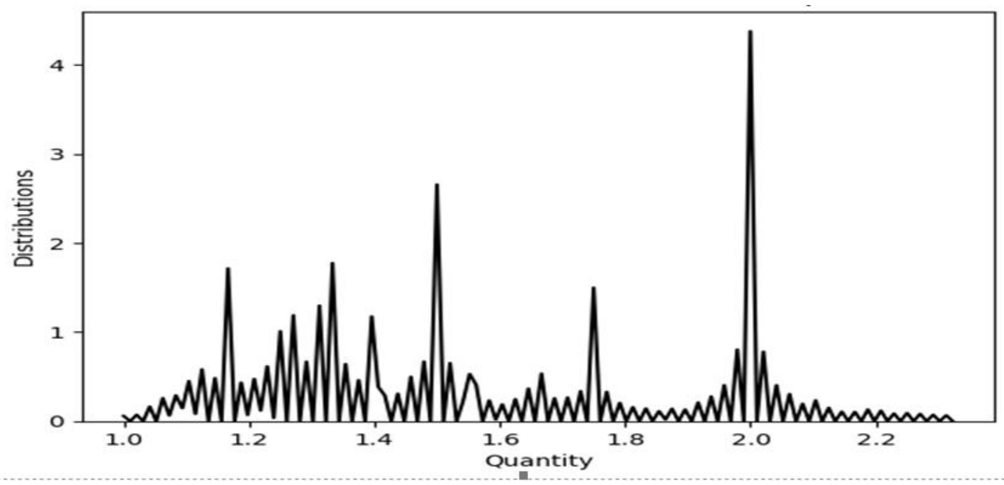


Figure 11: Distribution Plot of Quantity After Elimination (Group by User-based)

In the Figure 11 we can see some concentrations on some numbers like 1.1 or 2. So this plot is also more useful and helpful than statistic itself only.

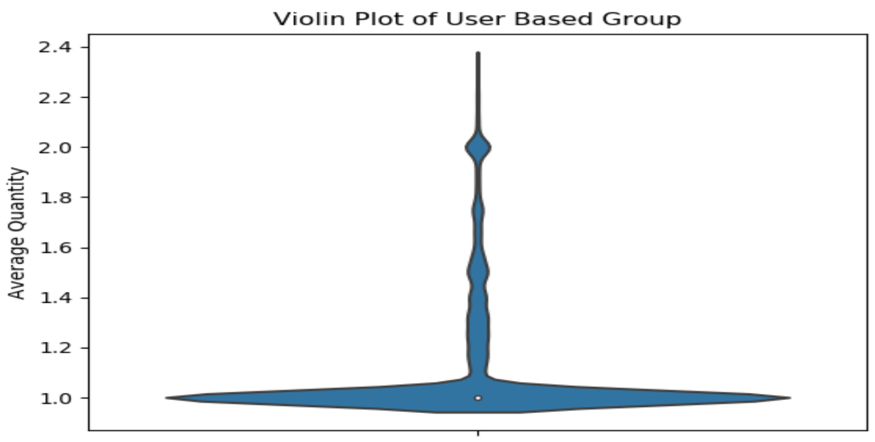


Figure 12: Violin Plot of Quantity (Group by User-based)

In Figure 12 it is obviously seen that user behaviors not well distributed. A lot of data group around the 1.

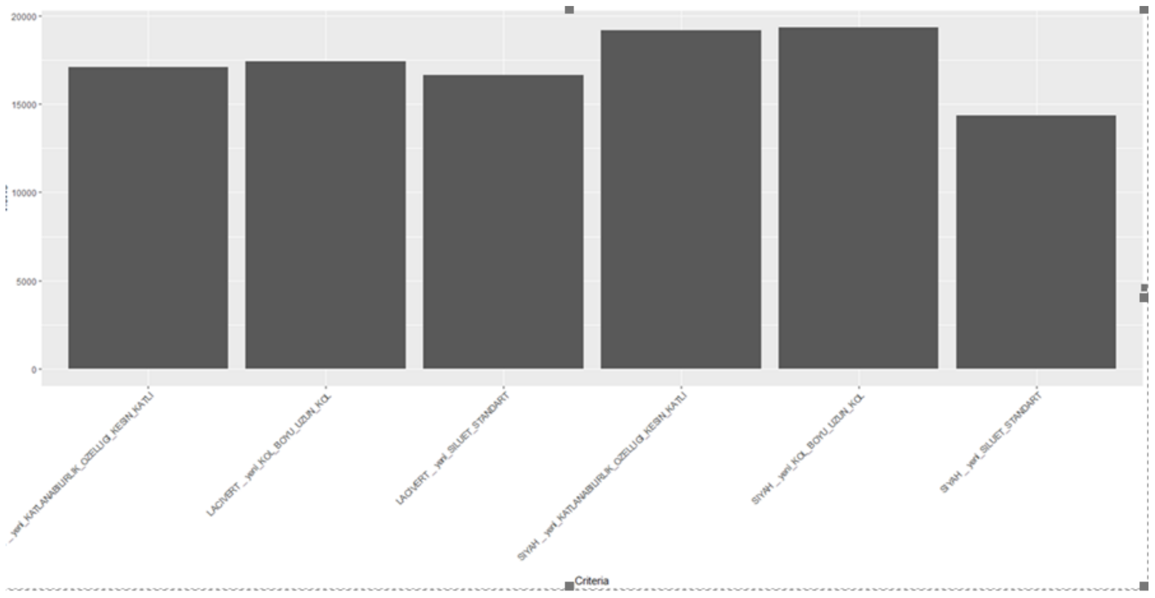
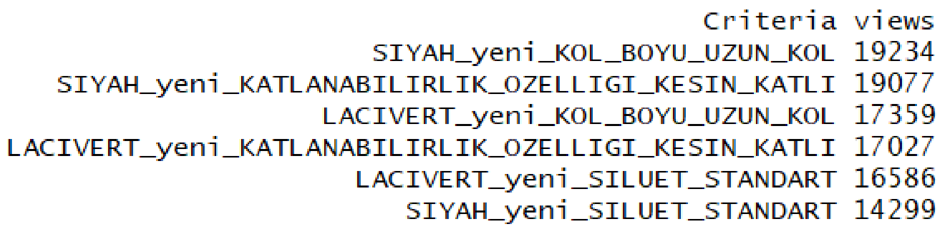


Figure 13: The histogram of the first 6 range criteria



```
                                      Criteria views
                      SIYAH_yeni_KOL_BOYU_UZUN_KOL 19234
     SIYAH_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI 19077
                   LACIVERT_yeni_KOL_BOYU_UZUN_KOL 17359
  LACIVERT_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI 17027
                    LACIVERT_yeni_SILUET_STANDART 16586
                       SIYAH_yeni_SILUET_STANDART 14299
```

Table 1: Number of Criteria in dataset

We see that in Figure 13, "SIYAH_yeni_KOL_BOYU_UZUN_KOL" is the most preferred criteria, exceeding the second most preferred "SIYAH_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI" by 19077 times.

To calculate the similarities between users and between criteria we transform the data into a different format. The columns include the criteria and the rows include the user at the new data format. Thus, we can calculate the similarities.

Below we can see the similarity of the first 10 users with each other by creating and visualizing similarity matrix that uses the cosine distance:

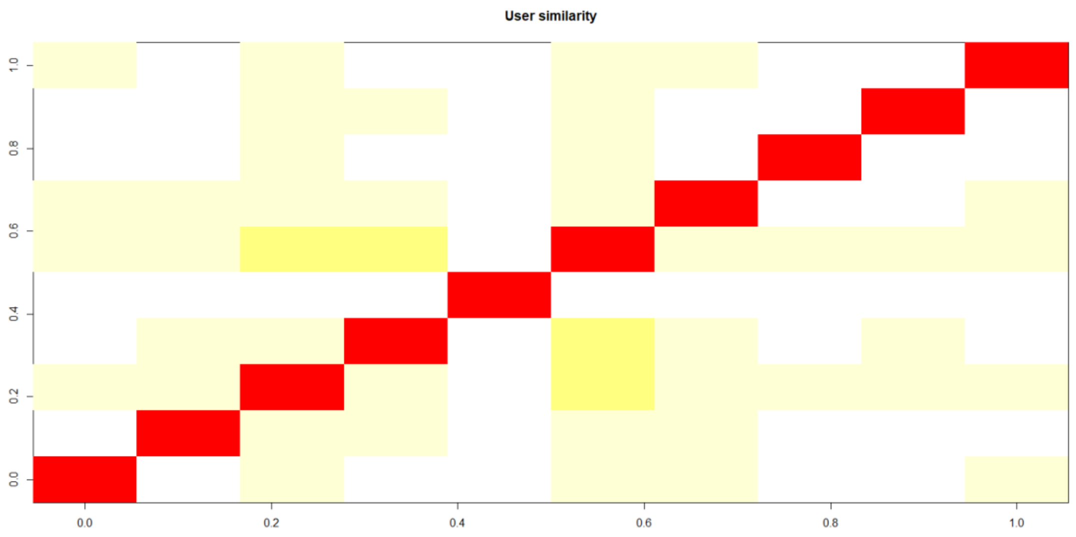


Figure 14: The similarity of the first 10 customers with each other

In the given matrix, each row and each column corresponds to a user, and each cell corresponds to the similarity between two users. The more red the cell is, the more similar two users are. Note that the diagonal is red, since it's comparing each user with itself.

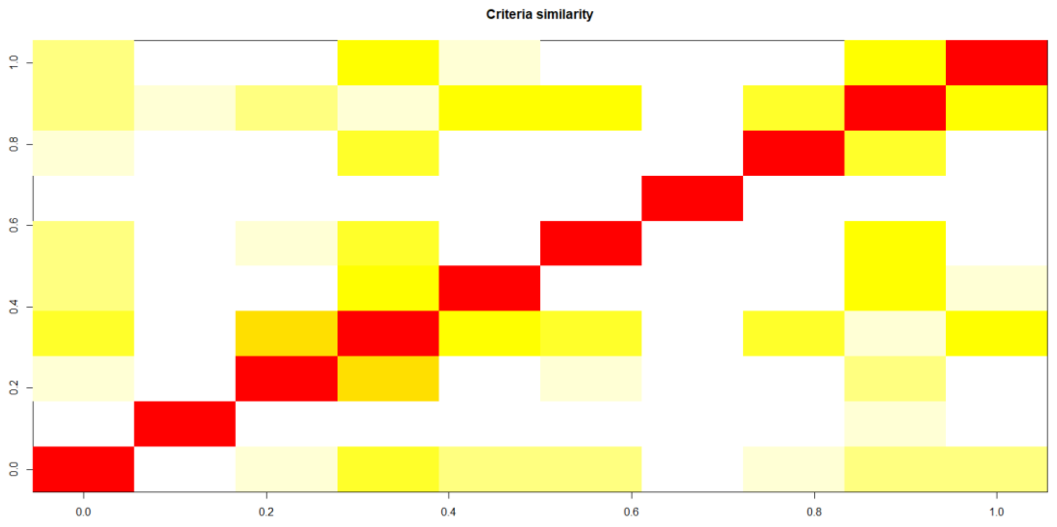


Figure 15: The similarity of the first 10 range criteria with each other

Using with the same approach, we compute similarity between the first 10 range criteria.Figure 14 and Figure 15 show that there are more similarities between range criteria than users.

We can visualize the whole matrix of quantities by building a heat map. Each row of the matrix corresponds to a user, each column to a criterion, and each cell to its quantity. Because of there are too many users and criteria, that may cause difficulty of read. Thus, we focused to display some random users and criteria. We selected the most relevant users and criteria and visualize only the users who have purchased many criteria and the criteria that have been purchased by many users.

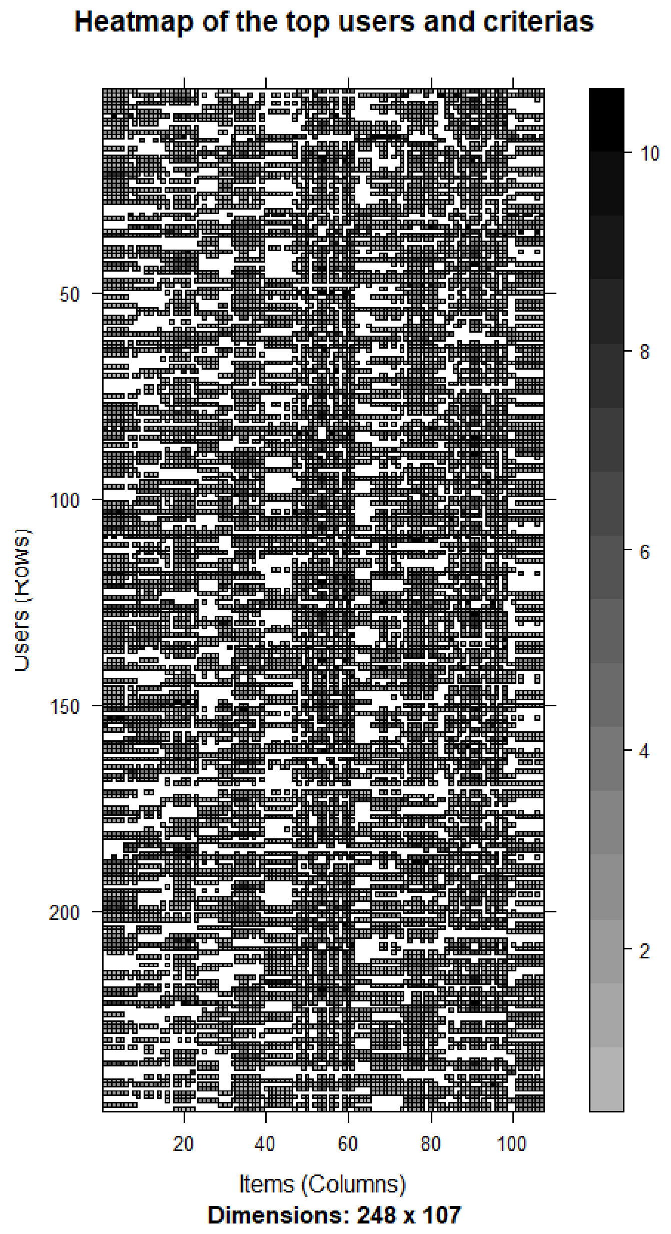


Figure 16: Heat map of the top users and criteria

If we focus on the users who purchase at a lot of criteria, most of them have purchase all the top criteria, and this is expected normally. Some columns of the heatmap are darker than the others, means that these columns show the most preferable criteria. On the other hand, darker rows show higher quantity of the criteria. Because of this, normalization might be helpful.

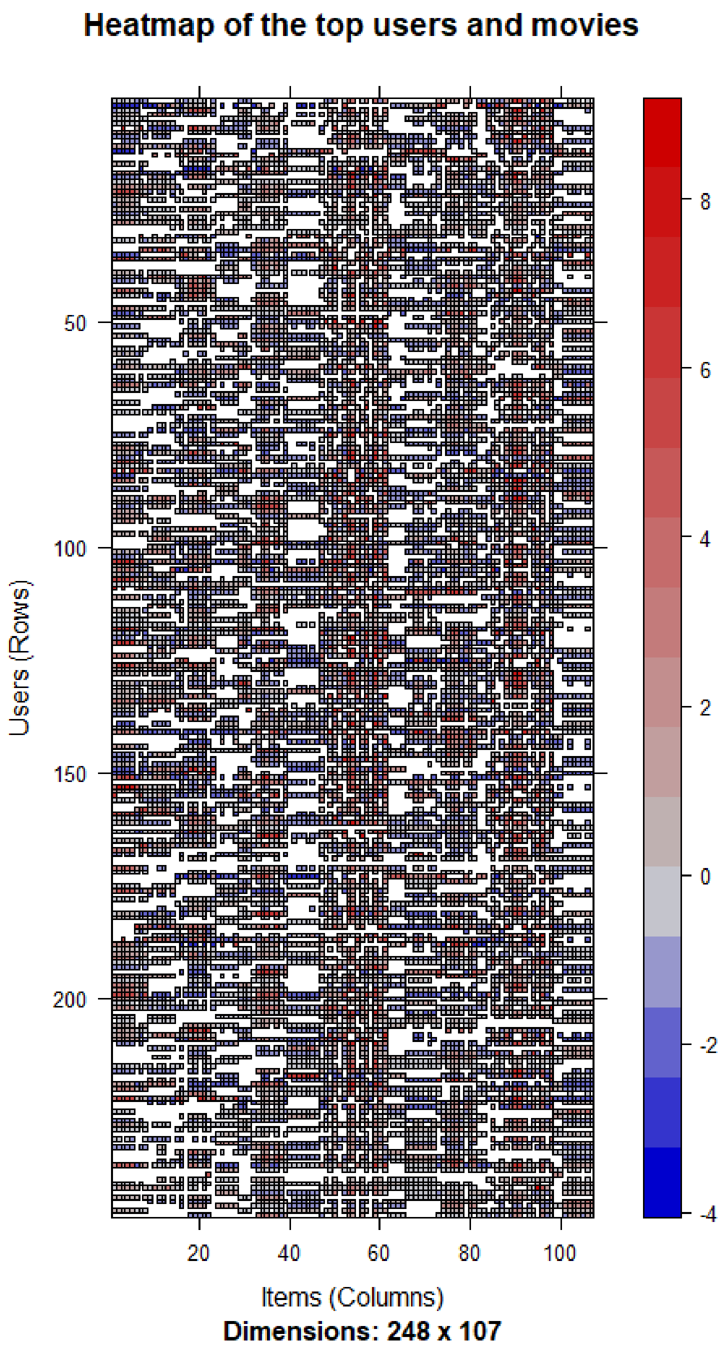


Figure 17: Heat map of the top users and criteria after normalization

In Figure 17 it is it is seen that users have different purchasing skills more clearly than Figure 16.

Some recommendation models work on binary data, so it might be useful to binarize the data, which means defining a table containing only 0's and 1's. The 0's will assume as missing values or as bad data.

We define a matrix having 1 if the user preferred a criterion, and 0 if the user not.

We define a matrix having 1 if the quantity is above or equal to a definite threshold (for example, 3), and 0 if it is not. Figure 18 and Figure 19 represents the matrix build on this rule.

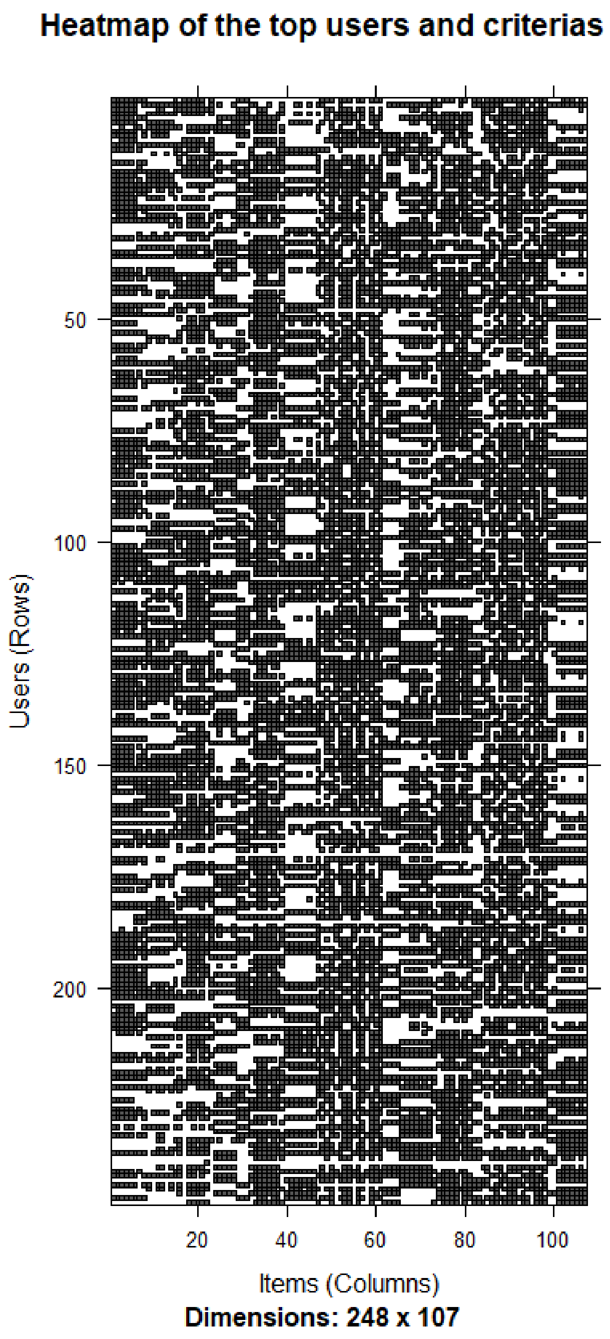


Figure 18: Heat map of the top users and criteria after binarization (without threshold)

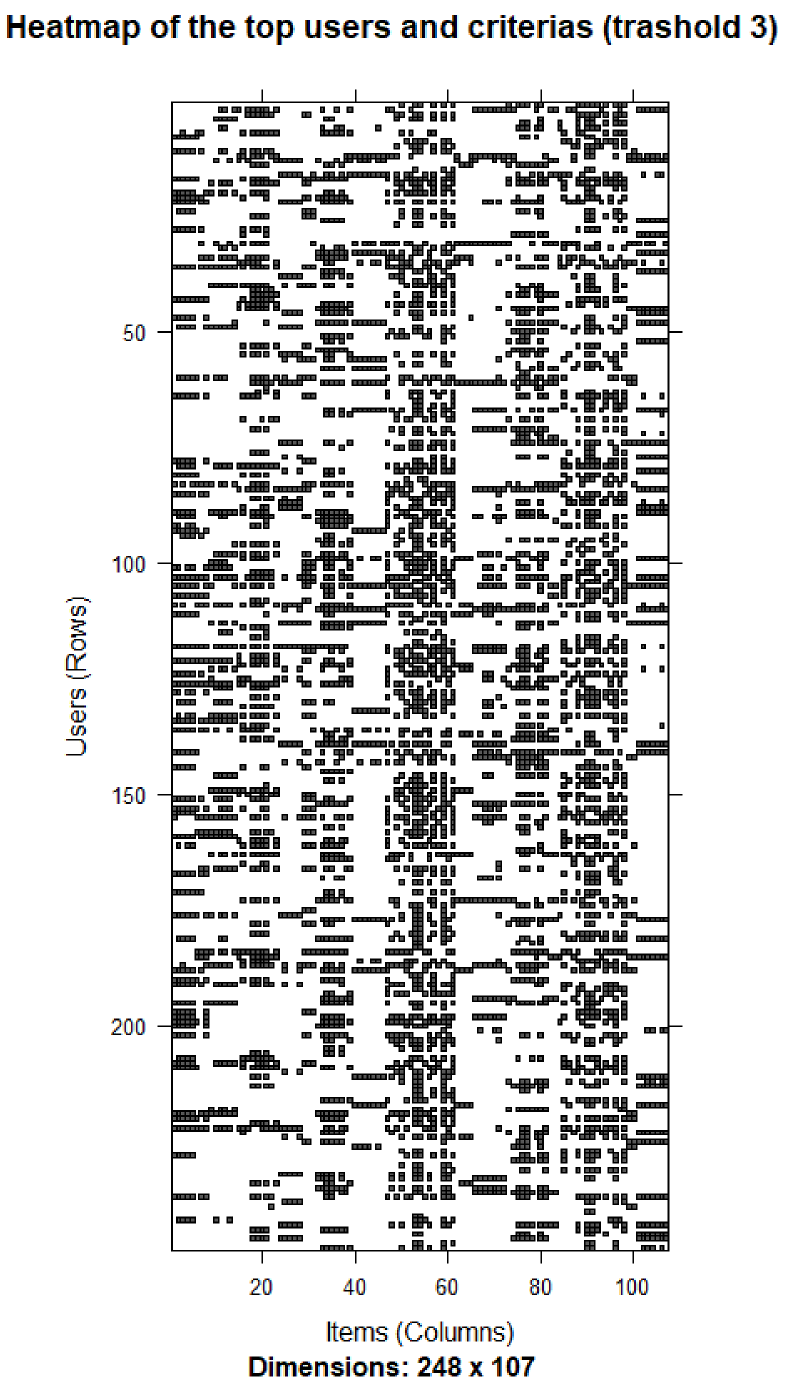


Figure 19: Heat map of the top users and criteria after binarization (Threshold=3)

There are more white cells in Figure 19, which shows that there are more criteria with no or small quantity than those that were not purchased by customers.

4. METHODOLOGHY

Recommendation engines utilize collaborative filtering. As the name suggests, collaborative filtering is a method that uses data from other people (or "users" on the platform) to make its prediction. Collaborative filtering can work in a few different ways. A collaborative filtering algorithm could 'filter' similar purchases users made in the past to generate and then recommend a list of items that go well together in combination. In this example, items that do not occur together frequently enough in past purchasing data would be dropped from the list, and the recommendation engine would make recommendation from a final set of items that have a strong history of being purchased together.

Recommendation algorithms generally make recommendations based on two types of collaborative filtering algorithms, user-based collaborative or item-based collaborative filtering.

4.1. User-based collaborative filtering systems

A user-based recommendation engine recommends product types based on what other users with similar profiles have bought and liked in the past. As an example of a user-based recommender, imagine there is a user type who purchased several types of products regularly, every weekend. A user-based recommender could go and look up product types recommendations based on what other similar profiles, who purchased regularly have liked.

4.2. Item-based collaborative filtering systems

An item-based recommender would make recommendations based on similarities between product types; in other words, it would recommend product types that are similar ones that a user already purchased. As an example of this, imagine you purchased a product type like long sleeve t-shirt with color yellow and supreme fabric and bought several times. The item-based collaborative filtering system would look into similar product types from the same fabric type and then recommend product types based on the preference you indicated when you bought before. In fact, an item-based collaborative filtering system can even make recommendations based on any variety of common elements, such as product about color and range criteria, product from the same category, etc. For this example, it is most likely that the primary suggestions will include "fabric type" followed by other cases.

Collaborative filtering is the most successful method of recommendation systems. Collaborative filtering algorithms consist of 2 different calculation steps before the estimation step: similarity calculation and neighborhood selection

4.3. Similarity calculations

The first step in collaborative filtering prediction algorithm is to weight all users with respect the similarity with the active users. (Herlocker & Konstan & Borchers & Riedl 1999).

At this stage, similarity calculations are performed on the products that the active user and other users evaluating the product evaluate together. To calculate the similarity, cosine similarity, Pearson correlation coefficient, adjustable cosine similarity etc. many techniques are used. The most successful of these is the Pearson correlation coefficient.

4.3.1. Cosine Similarity

In this similarity, each user is treated as a vector of previous evaluations. In this case, the angle cosine value between the two vectors expresses the similarity between the two vectors. The cosine similarity is calculated by Equation 1.

$$sim(a,u) = \frac{\sum_{i\in I} R_{a,i} \cdot R_{u,i}}{\sqrt{\sum_{i\in I} {R_{a,i}}^2} \cdot \sqrt{\sum_{i\in I} {R_{u,i}}^2}}$$

Equation 1: Equation of calculating Cosine similarity (Bulut & Milli 2014)

In Equation 1, $sim\ (a,\ u)$ represents the similarity value between active user and user u. The I represents the set of products that the active user and u user both have evaluated in the past. R_a represents the value of the active user gives to the product i. R_u represents the value of the user u gives to the product i.

In cosine similarity, the similarity between two users is between 0 and 1. If the result is closer to 1, it means that the users are very similar to each other.

4.3.2. Pearson Correlation Similarity

Users may perceive the evaluation scale differently. Some users prefer to use the upper values of the scale, while others prefer lower values while others can use a relatively homogeneous usage. Cosine similarity ignores users' perception of evaluation scale. The Pearson correlation coefficient reduces the negative effect by adding user averages. In other words, in the Pearson correlation coefficient method, the user evaluation vector is converted to the user preference vector and the preference is calculated. The Pearson Correlation Coefficient is as shown in Equation 2.

$$sim(a,u) = \frac{\sum_{i \in I}(R_{a,i} - \overline{R_a}).(R_{u,i} - \overline{R_u})}{\sqrt{\sum_{i \in I}(R_{a,i} - \overline{R_a})^2}.\sqrt{\sum_{i \in I}(R_{u,i} - \overline{R_u})}}$$

Equation 2: Equation of calculating Pearson correlation similarity (Bulut & Milli 2014)

In Equation 2, $sim\ (a,\ u)$ represents the similarity value between the active user and the u user. The I represents the set of products that the active user and u user have both evaluated in the past. $R_{a,i}$ represents the value that the active user gives to product i. $R_{u,i}$ represents the value that the user u gave to product i. $\overline{R_a}$ represents the average of the values that the user a gives to the products. $\overline{R_u}$ represents the averages of the values that the user u gives to the products.

Similarity between two users in the Pearson correlation coefficient similarity takes values between -1 and 1. If the result is closer to 1, it means that the users are very similar to each other.

4.4. Neighborhood Selection

The second stage of collaborative filtering algorithms is neighborhood selection. The similarity values found in the previous step are used to find the closest neighbors to the active user at this stage. In this stage, the threshold method and k nearest neighbor algorithms are the most used neighborhood selection methods.

In the threshold value method, users who have similarities to the active user over a certain value are selected as neighbors of the active user and included in the calculation, in k nearest neighbors, k users with the highest similarity to the active user are selected and included in the calculation.

The disadvantage of this algorithm is that it is difficult to select a k value that is appropriate to the data set in the nearest neighbors method. Choosing an appropriate k value will affect the accuracy of the calculation. If we select k low, some of the users that are similar to the active user will not be included in the calculation, so if we choose k high, the accuracy of the estimate will be negatively affected, since users who are not similar to the active user will be included in the calculation.

4.5. Why We Use Collaborative Filtering?

Collaborative filtering systems have many advantages over content-based recommenders. These advantages include:

- They can handle huge, high-dimensional datasets.
- They can suggest niche items (items popular among only a specific segment of users).
- They can suggest items which may be from a completely different product category all together.
- Based on the type of data you have; a collaborative filtering system can suggest items purchased by similar users solely depending upon their ratings for these items.

5. EXPERIMENTAL RESULTS

Collaborative filtering algorithms are based on measuring the similarity between users or between items. For this purpose, in R-Project program, there is a package called “recommenderlab” which contains the similarity function. The supported methods to compute similarities are cosine, pearson, and jaccard.

5.1. Results of Item Based Collaborative Filtering

Before start to build an Item Based Collaborative Filtering (IBCF) model, the data must be transformed to a sparse matrix which includes each criteria as a new feature by each user. The core algorithm is based on these steps:

- The algorithm calculates how similar they are in terms of having purchased similar quantities by similar users for each two criteria.
- To identify the k most similar items for each criterion.
- To identify the items that are most similar to the user's purchases for each user.

First, we start with defining train and test datasets. We define %80 of the data as training dataset and %20 of the dataset as test set.

When we look at the default parameters of IBCF model in package "recommenderlab", *k* is the number of items to compute the similarities among them in the first step. After, for each item, the algorithm identifies its *k* most similar items and stores the number. Method is a similarity function, which is *Cosine* by default, may also be *Pearson*. We built the model using the default parameters of method = Cosine and k=30.

In the IBCF model, a *dgCMatrix* matrix created which includes similarity rates. According these similarity rates Figure 19 and Figure 20 generated.

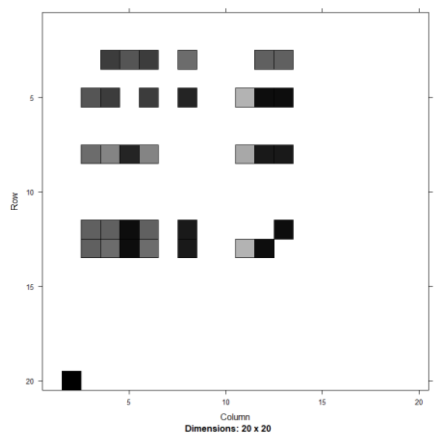


Figure 19: The heatmap of first 20 criteria

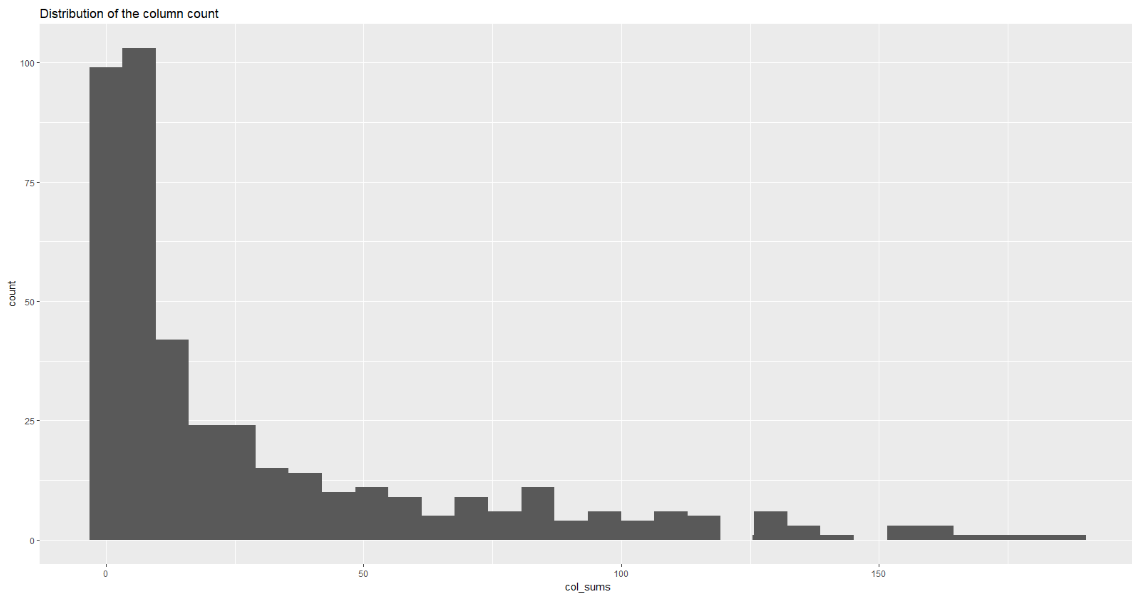


Figure 20: The distribution of the number of elements by column for IBCF model

Figure 20 shows that many values are equal to 0. The reason is that each row includes only 30 (*k*) elements that are greater than 0. The number of greater than 0 elements for each column depends on how many times the corresponding criteria was included in the top *k* of another criteria. Thus, the matrix is not necessarily symmetric.

The chart of the distribution of the number of elements by column shows there are a few criteria that are similar with many others.

It is possible to recommend criteria to the users in the test set. We define *n_recommended*" parameter equal to 10 that specifies the number of criteria to recommend to each user.

For each user, the algorithm extracts its criteria. For each criterion, it identifies all its similar criteria, starting from the similarity matrix. Then, the algorithm ranks each similar criterion in this way:

- Extract the quantities of each purchased associated with these criteria. The quantity is used as a weight.
- Extract the similarity of the criteria with each purchase associated with these criteria.
- Multiply each weight with the related similarity.
- Summation of all the rows.

For instance, the algorithm identifies the top 10 recommendations for the first user:

```
 [1] "BEJ_yeni_ANA_RANGE_KEY_JAKARLI"             "SARI_yeni_ELYAF_ICERIGI_VISKON_ELASTAN"
 [3] "BORDO_yeni_ANA_RANGE_KEY_DUZ"               "BEJ_yeni_SILUET_HIRKA"
 [5] "BORDO_yeni_ANA_RANGE_KEY_LACE"              "GRI_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_ASKILI"
 [7] "MAVI_yeni_ANA_RANGE_KEY_CIZGILI"            "MAVI_yeni_ELYAF_ICERIGI_VISKON_POLYESTER"
 [9] "MAVI_yeni_KUMAS_MALZEME_TIPI_CUT_SEW"       "KIRMIZI_yeni_ELYAF_ICERIGI_VISKON_ELASTAN"
```

Table 2: Top 10 recommended criteria for the first user for IBCF model

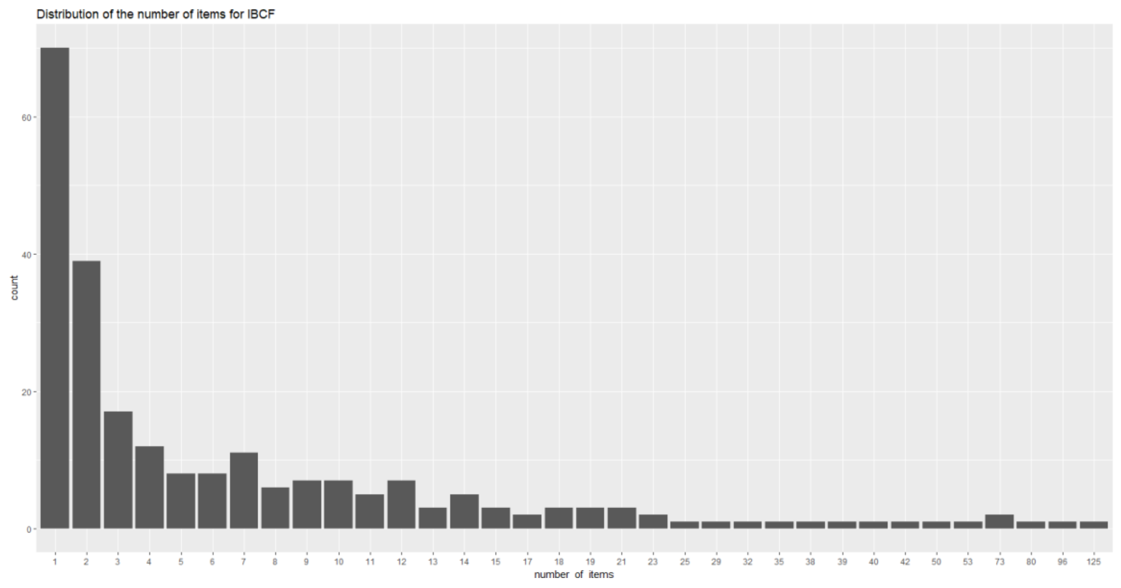
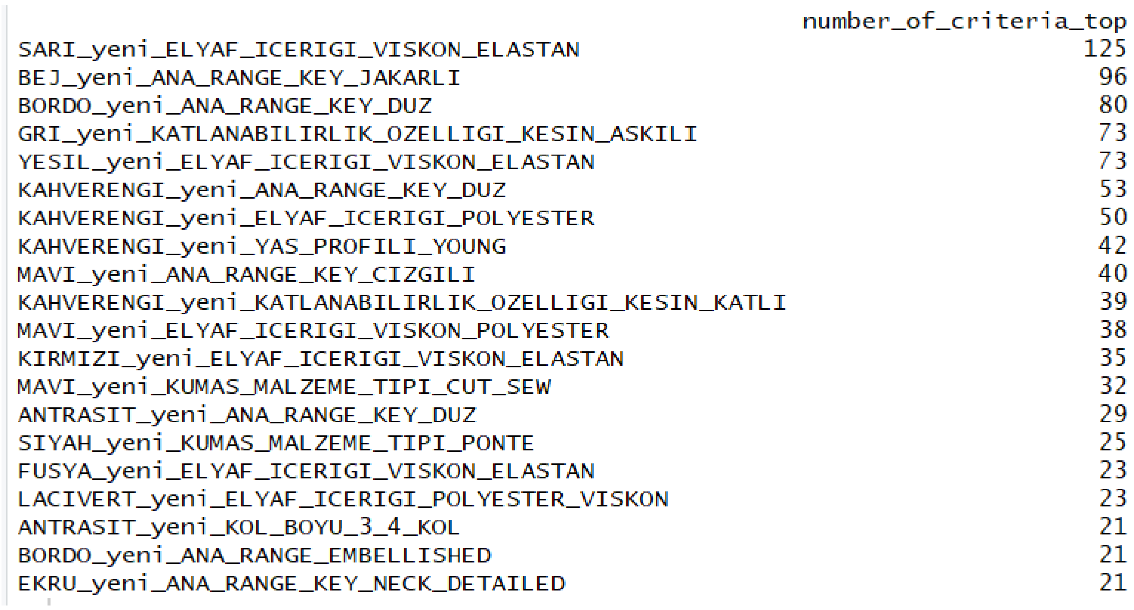


Figure 21: The histogram of the recommendations for IBCF model

Figure 21 shows that most of the criteria have been recommended only a few times, some criterias have been recommended more than 20 times which is shown at below. The maximum recommendation for a criteria is 125.



```
                                                          number_of_criteria_top
SARI_yeni_ELYAF_ICERIGI_VISKON_ELASTAN                                       125
BEJ_yeni_ANA_RANGE_KEY_JAKARLI                                                96
BORDO_yeni_ANA_RANGE_KEY_DUZ                                                  80
GRI_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_ASKILI                                73
YESIL_yeni_ELYAF_ICERIGI_VISKON_ELASTAN                                       73
KAHVERENGI_yeni_ANA_RANGE_KEY_DUZ                                             53
KAHVERENGI_yeni_ELYAF_ICERIGI_POLYESTER                                       50
KAHVERENGI_yeni_YAS_PROFILI_YOUNG                                             42
MAVI_yeni_ANA_RANGE_KEY_CIZGILI                                               40
KAHVERENGI_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                          39
MAVI_yeni_ELYAF_ICERIGI_VISKON_POLYESTER                                      38
KIRMIZI_yeni_ELYAF_ICERIGI_VISKON_ELASTAN                                     35
MAVI_yeni_KUMAS_MALZEME_TIPI_CUT_SEW                                          32
ANTRASIT_yeni_ANA_RANGE_KEY_DUZ                                               29
SIYAH_yeni_KUMAS_MALZEME_TIPI_PONTE                                           25
FUSYA_yeni_ELYAF_ICERIGI_VISKON_ELASTAN                                       23
LACIVERT_yeni_ELYAF_ICERIGI_POLYESTER_VISKON                                  23
ANTRASIT_yeni_KOL_BOYU_3_4_KOL                                                21
BORDO_yeni_ANA_RANGE_EMBELLISHED                                              21
EKRU_yeni_ANA_RANGE_KEY_NECK_DETAILED                                         21
```

Table 3: Criteria recommended more than 20 times for IBCF model

As shown in Table 3, the items with "SARI|_yeni_ELYAF_ICERIGI_VISKON_ELASTAN" criterion is the most common recommended criterion at the model.

IBCF recommends items on the basis of the similarity matrix. It's an eager-learning model, that is, once it is built, it does not need to access the initial data. For each item, the model stores the *k*-most similar, so the amount of information is small once the model is built. This is an advantage in the presence of lots of data.

5.2. User Based Collaborative Filtering

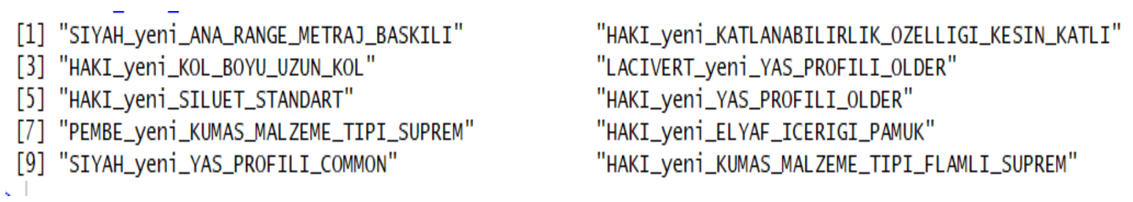
According to this approach, given a new user, firstly the similar users identified by the algorithm. Then, the top-rated items which rated by similar users are recommended.

For each new user, algorithm fallows these steps:

- Measuring the similarity of users is to the new one.
- Identifying the most similar users. The options are:
  - Selecting the top k users (k-nearest_neighbors)
  - Selecting the users whose similarity is higher than a defined threshold

- Rating the items rated by the most similar users. The rating is the average rating among similar users and the approaches are:
    - Average rating
    - Weighted average rating, using the similarities as weights
- Pick the top-rated items.

For instance, the User Based Collaborative Filtering(UBCF) algorithm identifies the top 10 recommendations for the first user:



```
[1] "SIYAH_yeni_ANA_RANGE_METRAJ_BASKILI"        "HAKI_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI"
[3] "HAKI_yeni_KOL_BOYU_UZUN_KOL"                "LACIVERT_yeni_YAS_PROFILI_OLDER"
[5] "HAKI_yeni_SILUET_STANDART"                  "HAKI_yeni_YAS_PROFILI_OLDER"
[7] "PEMBE_yeni_KUMAS_MALZEME_TIPI_SUPREM"       "HAKI_yeni_ELYAF_ICERIGI_PAMUK"
[9] "SIYAH_yeni_YAS_PROFILI_COMMON"              "HAKI_yeni_KUMAS_MALZEME_TIPI_FLAMLI_SUPREM"
```

Table 4: Top 10 recommended criteria for the first user at UBCF model

The default parameters of UBCF model, "*nn*" is the number of similar users, and method is a similarity function, which is cosine by default. We build a recommender model leaving the parameters to their defaults and using the training set. The default "*nn*" parameter is 25 for UBCF model.

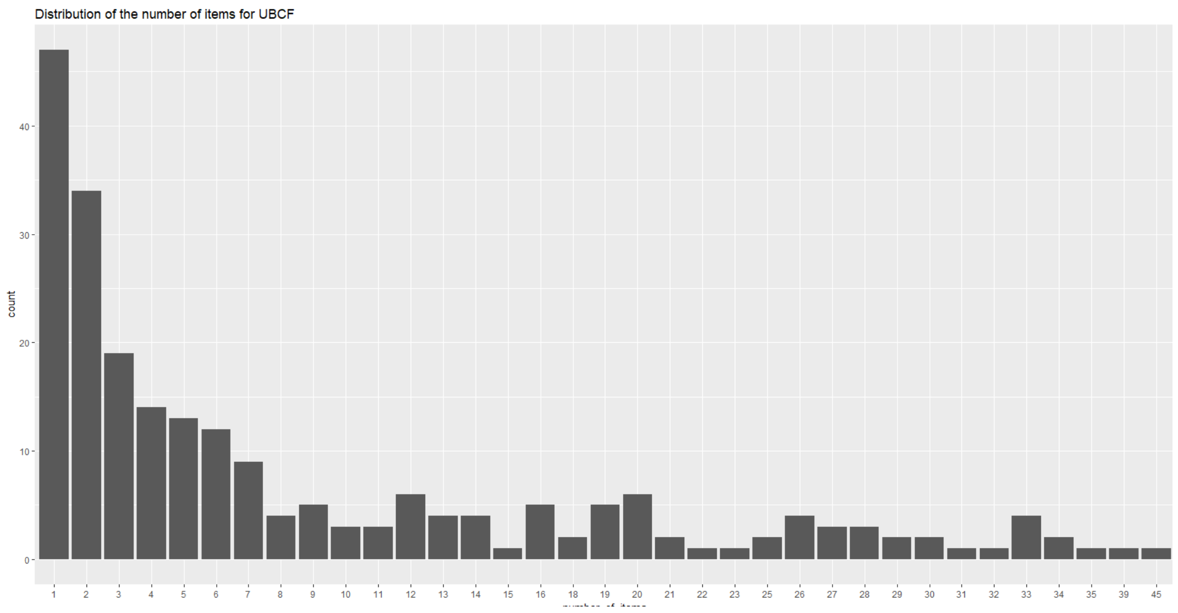
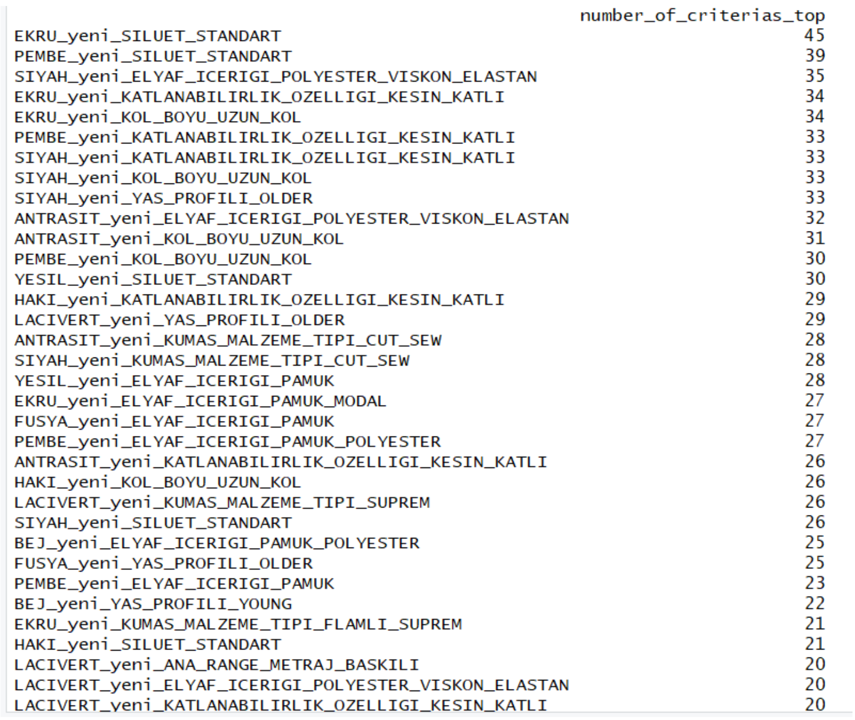


Figure 22: The histogram of the recommendations for UBCF model

Compared with the IBCF, the distribution has a long tail. On one hand, some criteria are recommended only one time, on the other hand some criteria recommend more than 20 times as shown in Figure 22. The maximum is more than 45 and at IBCF model it was 125. Some criteria have been recommended more than 20 times which is shown at below.



```
                                                    number_of_criterias_top
EKRU_yeni_SILUET_STANDART                                                45
PEMBE_yeni_SILUET_STANDART                                               39
SIYAH_yeni_ELYAF_ICERIGI_POLYESTER_VISKON_ELASTAN                        35
EKRU_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                           34
EKRU_yeni_KOL_BOYU_UZUN_KOL                                              34
PEMBE_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                          33
SIYAH_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                          33
SIYAH_yeni_KOL_BOYU_UZUN_KOL                                             33
SIYAH_yeni_YAS_PROFILI_OLDER                                             33
ANTRASIT_yeni_ELYAF_ICERIGI_POLYESTER_VISKON_ELASTAN                     32
ANTRASIT_yeni_KOL_BOYU_UZUN_KOL                                          31
PEMBE_yeni_KOL_BOYU_UZUN_KOL                                             30
YESIL_yeni_SILUET_STANDART                                               30
HAKI_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                           29
LACIVERT_yeni_YAS_PROFILI_OLDER                                          29
ANTRASIT_yeni_KUMAS_MALZEME_TIPI_CUT_SEW                                 28
SIYAH_yeni_KUMAS_MALZEME_TIPI_CUT_SEW                                    28
YESIL_yeni_ELYAF_ICERIGI_PAMUK                                           28
EKRU_yeni_ELYAF_ICERIGI_PAMUK_MODAL                                      27
FUSYA_yeni_ELYAF_ICERIGI_PAMUK                                           27
PEMBE_yeni_ELYAF_ICERIGI_PAMUK_POLYESTER                                 27
ANTRASIT_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                       26
HAKI_yeni_KOL_BOYU_UZUN_KOL                                              26
LACIVERT_yeni_KUMAS_MALZEME_TIPI_SUPREM                                  26
SIYAH_yeni_SILUET_STANDART                                               26
BEJ_yeni_ELYAF_ICERIGI_PAMUK_POLYESTER                                   25
FUSYA_yeni_YAS_PROFILI_OLDER                                             25
PEMBE_yeni_ELYAF_ICERIGI_PAMUK                                           23
BEJ_yeni_YAS_PROFILI_YOUNG                                               22
EKRU_yeni_KUMAS_MALZEME_TIPI_FLAMLI_SUPREM                               21
HAKI_yeni_SILUET_STANDART                                                21
LACIVERT_yeni_ANA_RANGE_METRAJ_BASKILI                                   20
LACIVERT_yeni_ELYAF_ICERIGI_POLYESTER_VISKON_ELASTAN                     20
LACIVERT_yeni_KATLANABILIRLIK_OZELLIGI_KESIN_KATLI                       20
```

Table 5: Criteria recommended more than 20 times at UBCF model

5.3. Comparing the Models

To compare the different models, we will define several methods with different parameters with the following list:

- Item-based collaborative filtering, using the Cosine as the distance function
- Item-based collaborative filtering, using the Pearson correlation as the distance function
- User-based collaborative filtering, using the Cosine as the distance function
- User-based collaborative filtering, using the Pearson correlation as the distance function
- Random recommendations

The following tables presents as an example the first rows of the performance evaluation matrix for each model at different number of recommendations. We can see at all tables when number of recommendations is getting higher precision rate is getting lower. Despite the precision rate getting lower, True Positive Rate (TPR) and False Positive Rate (FPR) is getting higher. This means that the model makes accurate predictions when the number of recommendations is getting higher.

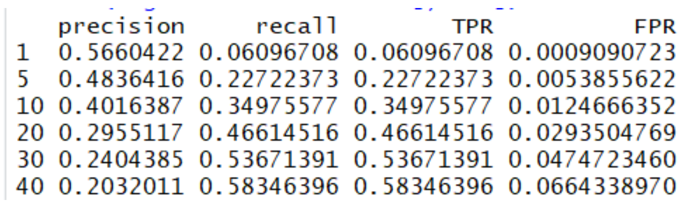

```
   precision     recall        TPR          FPR
1  0.5660422 0.06096708 0.06096708 0.0009090723
5  0.4836416 0.22722373 0.22722373 0.0053855622
10 0.4016387 0.34975577 0.34975577 0.0124666352
20 0.2955117 0.46614516 0.46614516 0.0293504769
30 0.2404385 0.53671391 0.53671391 0.0474723460
40 0.2032011 0.58346396 0.58346396 0.0664338970
```

Table 6: The evaluation matrix for UBCF model using pearson correlation distance (UBCF_cor)

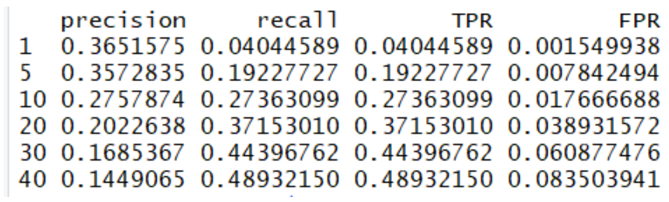

```
   precision     recall        TPR         FPR
1  0.3651575 0.04044589 0.04044589 0.001549938
5  0.3572835 0.19227727 0.19227727 0.007842494
10 0.2757874 0.27363099 0.27363099 0.017666688
20 0.2022638 0.37153010 0.37153010 0.038931572
30 0.1685367 0.44396762 0.44396762 0.060877476
40 0.1449065 0.48932150 0.48932150 0.083503941
```

Table 7: The evaluation matrix for UBCF model using cosine distance (UBCF_cos)

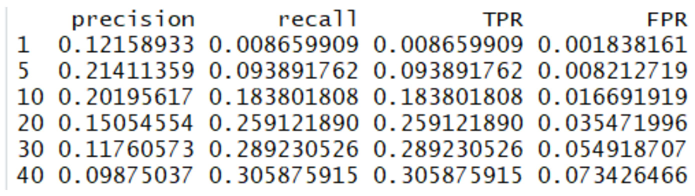

```
    precision      recall         TPR         FPR
1  0.12158933 0.008659909 0.008659909 0.001838161
5  0.21411359 0.093891762 0.093891762 0.008212719
10 0.20195617 0.183801808 0.183801808 0.016691919
20 0.15054554 0.259121890 0.259121890 0.035471996
30 0.11760573 0.289230526 0.289230526 0.054918707
40 0.09875037 0.305875915 0.305875915 0.073426466
```

Table 8: The evaluation matrix for IBCF model using pearson correlation distance (IBCF_cor)

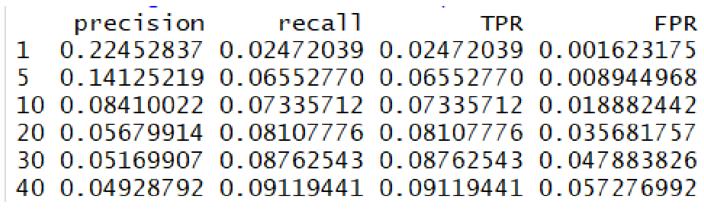

```
    precision     recall        TPR         FPR
1  0.22452837 0.02472039 0.02472039 0.001623175
5  0.14125219 0.06552770 0.06552770 0.008944968
10 0.08410022 0.07335712 0.07335712 0.018882442
20 0.05679914 0.08107776 0.08107776 0.035681757
30 0.05169907 0.08762543 0.08762543 0.047883826
40 0.04928792 0.09119441 0.09119441 0.057276992
```

Table 9: The evaluation matrix for IBCF model using cosine distance (IBCF_cos)

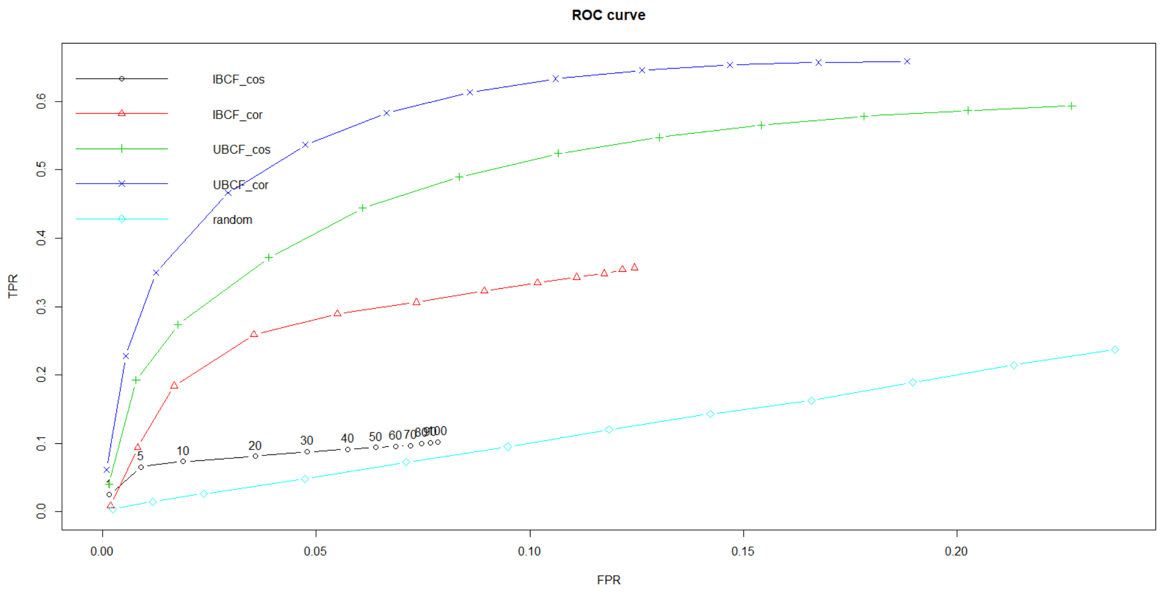


Figure 23: The ROC Curve of each model

A good way to understand the performance of a model is the area under the curve (AUC) which means that if AUC gets higher model performance is also gets higher. As shown in Figure 23, even without computing the certain AUC value, the chart shows that the highest performance is at the UBCF model with pearson correlation distance. The UBCF_cor model's TPR and FPR is getting higher when the number of recommendations getting higher as we mentioned before at Table 6.

With these results we can clearly say that User Based Collaborative Filtering method with using Pearson correlation distance outperform the other algorithms based on ROC curve and the evaluation matrixes.

6. CONCLUSION & DISCUSSION

In this project, we have developed and evaluated a collaborative filtering recommender (CFR) systems for recommending criteria for fast fashion online retailer users.

Recommendation systems have become an important strategic platform to understand the patterns and get information of internet users. It also helps to alleviate the problem of information overload which is a very common phenomenon with information retrieval systems and enables users to have access to products and services which are not readily available to users on the system. (Isinkaye & Folajimi & Ojokoh 2015). Web recommendation systems help the website visitors for easy navigation of web pages, quickly reaching their destination and to obtain relevant information (Suguna & Sharmila 2013). In this project we discussed the two traditional recommendation techniques and strategies used to improve their performances.

Our problem is motivated by practice at an online fashion retailer in Turkey. Four collaborative filtering-based algorithms and a random recommender are utilized to design a recommendation engine. 80% of the data is used for training while the other 20% is to used test the designed method. The expectation from the project was to make suggestions (predictions) on the test data with models using the similarity matrix. After building both user-based and item-based models with default parameters, the models re-trained to compare performances of different models with changing the distance calculation functions and the number of recommendations parameter. Based on our experiments, User Based Collaborative Filtering (UBCF) using Pearson correlation outperform the other 4 algorithms based on Receiver Operating Characteristic (ROC) curve.

In addition to all this information, some of the extra information about cons and pros of recommendation systems are to be emphasized;

User-based Collaborative Filtering offers suggestions that may be appropriate for the item that the user has previously purchased or rated. This can be a stronger proposition than a salesperson can do at a real store.

Collaborative Filtering uses all user data in the database to create recommendations and it is a memory-based method. Because of there are a lot of users, to look all the pairwise correlations between the users is not a good way and may have a huge computational time. A good way to solve this problem would be to apply a dimensionality reduction method, like a Principal Component Analysis.

Our future plans for this project to improve this model on different product categories of the online retailer which we used their data. We will try each possibility with changing level of parameters. Dimensionality reduction methods will also be applied to the model because of the number of feature of data will get higher with using the data of different categories.

7. REFERENCES

Badrul, S., George, K., Joseph, K., & Riedl, a. J. (2000). *Application of Dimensionality Reduction in Recommender System - A Case Study.* Minnesota: Department of Computer Science and Engineering University of Minnesota.

Hallinan, B., & Striphas, T. (2016). Recommend for you: the Netflix Proze and algorithmic culture. *New Media & Society*, 117-137.

Hemant Kumar, K., & Jeysree, J. (2014). Personalized Recommender System. *International Journal of Computer Applications*, 1-7.

Isinkaye, F. O., Folajimi, Y. O., & Ojokoh, B. O. (2015). Recommandetion Systems: Principles, Methods and Evaluation. *Egyptian Informatics Journal* , 261-273.

Shuai Zhang, L. Y. (2017). Deep learning based recommender system: A survey and new perspectives. *ArXiv*, 1-35.

Suguna,R. & Sharmila, D. (2013) An Efficient Web Recommendation System using Collaborative Filtering and Pattern Discovery Algorithms. *International Journal of Computer Applications,* 37-44.

Nasraoui,O. & Petenes,C. (2003) An intelligent Web recommendation engine based on fuzzy approximate reasoning. *The 12th IEEE International Conference on Fuzzy Systems,*

Suchal,J. & Navrat,P. (2010) Full text search engine as scalable k-nearest neighbor recommendation system. *IFIP International Conference on Artificial Intelligence in Theory and Practice,*1-7

Herlocker, J.L. & Konstan, J.A. & Borchers, A. & Riedl, J. (1999) An algorithmic framework for performing collaborative filtering, *22nd Annual International ACM SIGIR'99, Conference on Research and Development in Information Retrieval,* 15-19

Bulut, H. & Milli, M (2014) New prediction methods for collaborative filtering, *Pamukkale University Journal of Engineering Sciences,* 124-125

Isinkaye, F.O. & Folajimi, Y.O & Ojokoh, B.A. (2015) Recommendation systems: Principles, methods andevaluation, *Egyptian Informatics Journal 2015-16,* 261-273

APPENDIX A

Python Codes for Explanatory Data Analysis

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.api import stats as st

# Reading the data and changing the name of columns
re=pd.read_csv(r'C:\Users\HP\Desktop\Recommendation System
Mtilkat\REdata.csv',encoding='Latin-1')
re=re.iloc[:,1:]
re.columns=['musteri','mag','bug','klasman','renk','range','miktar']

# First we look the ranking by quantity which represent 'miktar'
# Applying z-score to whole data set and select without outliers.
re.nlargest(18,'miktar')
z_score_all=st.zconfint(re.miktar)
mean_all=re.miktar.mean()
re=re[(re.miktar< (mean_all + z_score_all[1])) & (re.miktar > (mean_all - z_score_all[0]))]

# Looking the lengths of the customers and the colors and the range
len(re.musteri.unique())
len(re.renk.unique())
len(re.range.unique())
# Basic statistics of data
re[re.miktar!=1].miktar.describe()
# Plotting the data
sns.countplot(re.renk)
sns.countplot(re.range)
```

```
# Drawing a distribution plot of the data
sns.distplot(re.miktar,hist=False,color='Green')
plt.title('Distribution Plot')
plt.xlabel('Quantity')
plt.ylabel('Distribution Rate')

# Data with group by color
color=re.groupby('renk')['miktar'].mean()

# Violin plot with data grouped by color
sns.violinplot(y=color,color= 'DarkRed')
plt.title('Violin Plot of Color Based Group')
plt.ylabel('Average Quantity')

# Data with group by range
ranges=re.groupby('range')['miktar'].sum()

# Distribution plot for data with grouped by range
sns.distplot(ranges,hist=False)
plt.title('Distribution Plot of Ranges Based Group')
plt.xlabel('Quantity')
plt.ylabel('Distribution Rate')

# 'Violin Plot of Range Based Group'
sns.violinplot(y=ranges,color= 'DarkRed')
plt.title('Violin Plot of Range Based Group')
plt.ylabel('Average Quantity')

# Range Based and Color Based Group's Box Plot
sns.boxplot(y=ranges,color='pink')
plt.ylabel('Average Quantity')
plt.title("Range Based Group's Box Plot")
```

```
sns.boxplot(y=color,color='purple')
plt.ylabel('Average Quantity')
plt.title("Color Based Group's Box Plot")

# Data with group by users
user=re.groupby('musteri')['miktar'].mean()

# Distribution plot for data with grouped by users
sns.distplot(user,hist=False)
sns.boxplot(user)

# Finding confidence interval and scaling the data
user=user[(user > (user.mean()- st.zconfint(user)[0])) & ( user < (user.mean() +
st.zconfint(user)[1]))]

# All values of user data
user.describe()

# Violin and Distribution Plot of User Based Group'
sns.violinplot(y=user)
plt.title('Violin Plot of User Based Group')
plt.ylabel('Average Quantity')

sns.distplot(user,color='Black',hist=False)
plt.title('Distribution Plot of User Bssed Group')
plt.ylabel('Distributions')
plt.xlabel('Quantity')

# Re defining the colors
re[re['renk']=='PEMBE'],re[re['renk']=='SÝYAH'],re[re['renk']=='HAKÝ'],re[re['renk']=='L
ACÝVERT'],re[re['renk']=='ANTRASÝT']='Pink','Black','Khaki','Dark Blue','Antracite'
```

```
# Finding the best seller ranges and colors
top_color=re.groupby('renk')['miktar'].sum().sort_values(ascending=False)
top_range=re.groupby('range')['miktar'].sum().sort_values(ascending=False)

# Histogram of the colors and range which are best sellers
sns.countplot(re[re['renk'].isin(['Pink','Black','Khaki','Dark Blue','Antracite'])]['miktar'])
plt.xlabel('Quantity')
sns.countplot(re[re['range'].isin(top_range.index[:5])]['range'])
plt.xticks(rotation=66)
plt.xticks()
```

APPENDIX B

R Codes for Recommendation Models

```
#Loading the libraries
library(recommenderlab)
library(ggplot2)
library(data.table)
library(reshape2)
library(methods)
library(recommenderlab)
library(data.table)
library(ggplot2)
library(knitr)
library(stringr)
set.seed(123)
#Read the data
data<-read.csv("C:/Users/HP/Desktop/BDA/capstone/REdata.csv")
data<-data[,-1]
data<-data.table(data)
data[Miktar<=0,Miktar:=NA]
data <- na.omit(data)

#Changing the types of Turkish charachters
data$Range<-str_replace_all(data$Range," ","_")
data$Range<-str_replace_all(data$Range,"Ü","U")
data$Range<-str_replace_all(data$Range,"Ö","O")
data$Range<-str_replace_all(data$Range,"Ş","S")
data$Range<-str_replace_all(data$Range,"Ç","C")
data$Range<-str_replace_all(data$Range,"Ğ","G")
data$Range<-str_replace_all(data$Range,"İ","I")
data$Range<-str_replace_all(data$Range,"/","_")
data$Range<-str_replace_all(data$Range,"&","_")
```

```
data$Range<-str_replace_all(data$Range,"-","_")


data$AnaRenkTanim<-str_trim(data$AnaRenkTanim)

data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim," ","_")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"Ü","U")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"Ö","O")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"Ş","S")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"Ç","C")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"Ğ","G")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"İ","I")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"/","_")
data$AnaRenkTanim<-str_replace_all(data$AnaRenkTanim,"&","_")

#Creating the Criteria column combining with Range and Color column
data[,Criteria:=str_c(AnaRenkTanim,"_",Range,sep ="")]
data[,musteriref:=str_c("U",musteriref,sep ="")]

data[,MerchAltGrupKod:=NULL]
data[,BuyerGrupTanim:=NULL]
data[,UrunKlasmanTanim:=NULL]
data[,AnaRenkTanim:=NULL]
data[,Range:=NULL]

#Quick look at the data on column quantity
quantile(data$Miktar,probs =c(seq(0,1,0.001)))

#Create quantity matrix. Rows = userId, Columns = Criteria
quantitymat <- dcast(data[Miktar<=10,], musteriref~Criteria, value.var = "Miktar",
na.rm=FALSE)
quantitymat <- as.matrix(quantitymat[,-1]) #remove userIds
```

```
#Convert quantity matrix into a recommenderlab sparse matrix
quantitymat <- as(quantitymat, "realRatingMatrix")
quantitymat

#Transforming data into real rating matrix
recommender_models <- recommenderRegistry$get_entries(dataType =
"realRatingMatrix")
names(recommender_models)



#calculating the similarity matrix for users
similarity_users <- similarity(quantitymat[1:10, ],
                              method = "cosine",
                              which = "users")
as.matrix(similarity_users)
image(as.matrix(similarity_users), main = "User similarity")

#calculating the similarity matrix for items
similarity_items <- similarity(quantitymat[, 1:10], method =
                               "cosine", which = "items")
as.matrix(similarity_items)
image(as.matrix(similarity_items), main = "Criteria similarity")

# count quantity for each criteria
count_per_criteria <- colCounts(quantitymat)

table_counts <- data.frame(Criteria = names(count_per_criteria),
                          views = count_per_criteria) # create dataframe of quantity
table_counts <- table_counts[order(table_counts$views,
                               decreasing = TRUE), ] # sort by number of quantity
```

```
table_counts[1:6,]

ggplot(table_counts[1:6, ], aes(x = Criteria, y = views)) +
  geom_bar(stat="identity") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  ggtitle("Number of count of the top criteria")



average_quantites <- colMeans(quantitymat)

min_n_criteria <- quantile(rowCounts(quantitymat), 0.75)
min_n_users <- quantile(colCounts(quantitymat), 0.75)
print("Minimum number of criteria per user:")
min_n_criteria
print("Minimum number of users per criteria:")
min_n_users

#Heat map with all dimensions
image(quantitymat[rowCounts(quantitymat) > min_n_criteria,
                colCounts(quantitymat) > min_n_users],
      main = "Heatmap of the top users and criteria")

#Selecting more the more important criteria and customer
quantity_criteria <- quantitymat[rowCounts(quantitymat) > 50,
                             colCounts(quantitymat) > 50]
quantity_criteria
#ratingmat

#Heatmap of the top users and criterias
min_criteria <- quantile(rowCounts(quantity_criteria), 0.75)
min_users <- quantile(colCounts(quantity_criteria), 0.75)
image(quantity_criteria[rowCounts(quantity_criteria) > min_criteria,
                    colCounts(quantity_criteria) > min_users],
      main = "Heatmap of the top users and criterias")

#Distribution of the average quantites per user
average_quantities_per_user <- rowMeans(quantity_criteria)
qplot(average_quantities_per_user) + stat_bin(binwidth = 0.1) +
  ggtitle("Distribution of the average quantites per user")

#Normalizing data
quantities_criteria_norm <- normalize(quantity_criteria)
sum(rowMeans(quantities_criteria_norm) > 0.00001)

#Heat map with normalized data
image(quantities_criteria_norm[rowCounts(quantities_criteria_norm) > min_criteria,
                         colCounts(quantities_criteria_norm) > min_users],
      main = "Heatmap of the top users and criteria")

#Heat map with binarizing data
quantities_criteria_purchased <- binarize(quantity_criteria, minRating = 1)
min_criteria_binary <- quantile(rowCounts(quantity_criteria), 0.75)
min_users_binary <- quantile(colCounts(quantity_criteria), 0.75)
image(quantities_criteria_purchased[rowCounts(quantity_criteria) > min_criteria_binary,
                            colCounts(quantity_criteria) > min_users_binary],
      main = "Heatmap of the top users and criterias")

#Heat map binarized data with a threshold.
quantities_criteria_good <- binarize(quantity_criteria, minRating = 3)
image(quantities_criteria_good[rowCounts(quantity_criteria) > min_criteria_binary,
                         colCounts(quantity_criteria) > min_users_binary],
      main = "Heatmap of the top users and criterias (trashold 3)")

which_train <- sample(x = c(TRUE, FALSE),
                    size = nrow(quantity_criteria),
                    replace = TRUE,
                    prob = c(0.8, 0.2))
#head(which_train)

recc_data_train <- quantity_criteria[which_train, ]
recc_data_test <- quantity_criteria[!which_train, ]



#Building IBCF model
recommender_models <- recommenderRegistry$get_entries(dataType
="realRatingMatrix")
recommender_models$IBCF_realRatingMatrix$parameters

recc_model <- Recommender(data = recc_data_train,
                         method = "IBCF",
                         parameter = list(k = 30))

recc_model
class(recc_model)



model_details <- getModel(recc_model)
#model_details$description
#model_details$k

class(model_details$sim) # this contains a similarity matrix
dim(model_details$sim)

n_items_top <- 20
image(model_details$sim[1:n_items_top, 1:n_items_top],
      main = "Heatmap of the first rows and columns")
```

```
row_sums <- rowSums(model_details$sim > 0)
table(row_sums)
col_sums <- colSums(model_details$sim > 0)
qplot(col_sums) + stat_bin(binwidth = 1) + ggtitle("Distribution of the column count")

n_recommended <- 10 # the number of items to recommend to each user

recc_predicted <- predict(object = recc_model,
                          newdata = recc_data_test,
                          n = n_recommended)
recc_predicted

#class(recc_predicted)
#slotNames(recc_predicted)

# recommendation for the first user
recc_user_1 <- recc_predicted@items[[1]]
criteria_user_1 <- recc_predicted@itemLabels[recc_user_1]
criteria_user_1



# matrix with the recommendations for each user
recc_matrix <- sapply(recc_predicted@items,
                      function(x){ as.character(colnames(quantity_criteria)[x]) })
#dim(recc_matrix)

number_of_items <- factor(table(unlist(recc_matrix)))

chart_title <- "Distribution of the number of items for IBCF"
qplot(number_of_items) + ggtitle(chart_title)
```

```
number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
number_of_criteria_top <- head(number_of_items_sorted, n = 20)
table_top <- data.frame(number_of_criteria_top)

table_top

#Building UBCF model
recommender_models <- recommenderRegistry$get_entries(dataType
="realRatingMatrix")
recommender_models$UBCF_realRatingMatrix$parameters
recc_model <- Recommender(data = recc_data_train, method = "UBCF")
recc_model
model_details <- getModel(recc_model)
#names(model_details)
model_details$data
model_details$method

n_recommended <- 10
recc_predicted <- predict(object = recc_model,
                          newdata = recc_data_test,
                          n = n_recommended)
recc_predicted

# recommendation for the first user
recc_user_1 <- recc_predicted@items[[1]]
criteria_user_1 <- recc_predicted@itemLabels[recc_user_1]
criteria_user_1



recc_matrix <- sapply(recc_predicted@items,
                      function(x){ as.character(colnames(quantity_criteria)[x]) })
#dim(recc_matrix)
```

```
recc_matrix[, 1:4]

number_of_items <- factor(table(recc_matrix))

chart_title <- "Distribution of the number of items for UBCF"
qplot(number_of_items) + ggtitle(chart_title)

number_of_items_sorted <- sort(number_of_items, decreasing = TRUE)
number_of_criterias_top <- head(number_of_items_sorted, n = 37)
table_top <- data.frame(number_of_criterias_top)

table_top

min(rowCounts(quantity_criteria))
items_to_keep <- 5 #number of items to generate recommendations
quantity_threshold <- 3 # threshold with the minimum quantity that is considered good
n_eval <- 1 #number of times to run evaluation
percentage_training <- 0.8
n_fold <- 4

eval_sets <- evaluationScheme(data = quantity_criteria,
                              method = "cross-validation",
                              k = n_fold,
                              given = items_to_keep,
                              goodRating = quantity_threshold)

#Selecting model and parameters
models_to_evaluate <- list(
  IBCF_cos = list(name = "IBCF",
                  param = list(method = "cosine")),
  IBCF_cor = list(name = "IBCF",
                  param = list(method = "pearson")),
  UBCF_cos = list(name = "UBCF",
                  param = list(method = "cosine")),
  UBCF_cor = list(name = "UBCF",
                  param = list(method = "pearson")),
  random = list(name = "RANDOM", param=NULL)
)

n_recommendations <- c(1, 5, seq(10, 100, 10))
list_results <- evaluate(x = eval_sets,
                         method = models_to_evaluate,
                         n = n_recommendations)

sapply(list_results, class) == "evaluationResults"

avg_matrices <- lapply(list_results, avg)
head(avg_matrices$UBCF_cor[, 5:8])
head(avg_matrices$UBCF_cos[, 5:8])
head(avg_matrices$IBCF_cor[, 5:8])
head(avg_matrices$IBCF_cos[, 5:8])

plot(list_results, annotate = 1, legend = "topleft")
title("ROC curve")
```