MEF UNIVERSITY

PRODUCT RECOMMENDATION FOR C2C MARKETPLACE WITH COLLABORATIVE FILTERING ALS ALGORITHM

Capstone Project

Bilgehan Kıran Çelebi

İSTANBUL, 2021

MEF UNIVERSITY

PRODUCT RECOMMENDATION FOR C2C MARKETPLACE WITH COLLABORATIVE FILTERING ALS ALGORITHM

Capstone Project

Bilgehan Kıran Çelebi

Advisor: Asst. Prof. Dr. Utku Koç

İSTANBUL, 2021

MEF UNIVERSITY

Name of the project: Product Recommendation For C2C Marketplace With Collaborative Filtering ALS Algorithm

Name/Last Name of the Student: Bilgehan Kıran Çelebi
Date of Thesis Defense: 22/01/2021

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".

22/01/2021
Asst. Prof. Dr. Utku Koç

I hereby state that I have examined this graduation project by Your Name (Title Format) 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.

22/01/2021

Director
of
Big Data Analytics Program

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

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Asst. Prof. Dr. Utku Koç	……
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
Bilgehan Kıran Çelebi	22/01/2021	

EXECUTIVE SUMMARY

PRODUCT RECOMMENDATION FOR C2C MARKETPLACE WITH COLLABORATIVE FILTERING ALS ALGORITHM

Bilgehan Kıran Çelebi

Advisor: Asst. Prof. Dr. Utku Koç

JANUARY, 2021, 34 pages

In this project, a machine learning recommendation model is created for an e-commerce company which runs a customer to customer business. The raw data consisted order reviews, order details, product like event information and product details. The explicit and implicit feedbacks are used together and a rating generation logic per user-product couple is applied to create the source data of the model by using Google Cloud BigQuery tool. The ALS algorithm which uses matrix factorization is applied for predicting the top items which have highest ratings for each user. PySpark which is Apache Spark's python API is used for implementing the ALS model. The best hyperparameters are determined comparing the root mean square error results by using grid search and cross validation and 0.78 of RMSE is reached. The predictions for the empty ratings are sorted then top rated 10 products are taken as recommendations. The evaluation of the model is done by comparing those recommendations with the user preferences. The user preferences are specified by using averagely top rated product categories and most interacted product categories in count. The recommendations are observed to be consistent with the user preferences.

.

Key Words: Recommendation Engines, Collaborative Filtering, Matrix Factorization, ALS Algorithm, PySpark, BigQuery

ÖZET

E- TICARET SİTESİ İÇİN COLLABORATIVE FILTERING ALS ALGORITMASI İLE ÜRÜN ÖNERME

Bilgehan Kıran Çelebi

Proje Danışmanı: Asst. Prof. Dr. Utku Koç

OCAK, 2021, 34 Sayfa

Bu projede müşteriden müşteriye satış yapan bir elektronik pazarlama şirketi için ürün önerme üzerine bir makine öğrenmesi modeli kurulmuştur. Şirketten alınan ham data sipariş değerlendirme, sipariş detayı, ürün beğeni ve ürün bilgisi verilerini içermektedir. Açık ve örtülü geribildirimler Google Bulut ortamında BigQuery aracı kullanılarak modelin kaynak verisini oluşturmak için kullanılmıştır. Her kullanıcı için en yüksek puanlama değerini tahmin etmek için matris ayrıştırma kullanan ALS algoritması uygulanmıştır. ALS modelini uygulamak için Apache Spark'ın python programlama arayüzü olan PySpark kullanılmıştır. Hataların ortama karekökü (RMSE) sonuçları karşılaştırılarak, grid arama ve çapraz sağlama ile en iyi hiper parametreler belirlenmiş ve 0.78 değerine ulaşılmıştır. Boş puanlamalar için tahminlenen değerler sıralanıp en yüksek 10 tanesi öneri olarak alınmıştır. Modelin değerlendirmesi bu önerilerin kullanıcı tercihleri ile karşılaştırılması ile yapılmıştır. Kullanıcı tercihleri ortalama ratingleri en yüksek olan ürün kategorileri ve sayı olarak en fazla iletişimde bulunulmuş kategoriler ile belirlenmiştir. Verilen önerilerin kullanıcı tercihleri ile uyumlu olduğu gözlemlenmiştir.

Anahtar Kelimeler: Öneri Sistemleri, Collaborative Filtering, Matris Ayrıştırma, ALS Algoritması, PySpark, BigQuery

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

In the last years, especially after Covid-19 pandemic the consumers' way of making purchase decisions have changed dramatically. Mobile technologies increasingly effect the stages of customer shopping journey. Before making a buying decision people refer to the product reviews, other people's shopping decisions via social media, compare prices between online stores and etc. Moreover, when they decide purchasing an item, growing number of online retailers are ready to deliver the product soon even on the same day. The retail environment is as dynamic as it never did before [21]. The key point of success for a company is to know their customers well and behave them according to their needs and interests. Keeping customers engaged and acquiring new customers by providing a good service is exclusively important.

The data used in the project is obtained from a Turkish startup company which acts as a customer to customer (C2C) marketplace via their mobile application and also their website. The goal of a C2C is to enable relationships, helping buyers and sellers communicate to each other. In this case, the sellers can also be buyers or vice versa. For a high quality service we need to make interaction of these types of customers with a good control mechanism by taking advantage of AI technologies.

E-commerce can be briefly explained as the purchase and sale of goods or services via electronic channels such as the internet. E-commerce was firstly introduced in the early1970s as EFT, electronic funds transfer, which is used for transferring money electronically by mostly among financial institutions [15]. Later with the availability of internet access, popular online sellers took their roles in the history in the 1990s and early 2000s.

C2C is a category of e-commerce which is a business model that facilitates commerce between individuals. It can be for goods or services and it connects people to do business with one another. Some examples for C2C marketplace companies are eBay as Gittigidiyor in Turkey, Sahibinden.com and Letgo. In addition, Amazon has both of business to customer (B2C) and C2C marketplaces [20]. An advantage of C2C business is the chance for users get minimal costs involved with the lack of retailers and wholesalers, keeping the margins higher for sellers and prices lower for buyers. C2C websites are the

platforms for matching buyers to sellers. The products can be new or second hand, the companies do not have control over the quality of those products.

1.1. Recommendation Systems

E-commerce websites had come to the point of offering millions of items to customers. This item density makes it difficult for users to choose the item which is appropriate for them. Moreover, as users enjoy viewing products selected by them, offering them some other products which can also take their interest is a plus for the website. At this point recommendation systems become increasingly important. It helps to provide better customer experience. E-commerce sites and mobile applications allow us automatically collect clickstream data which are every single movement records of a user's page view and click history and they use this data in their recommendation models.

Recommender systems have a significant role in e-commerce field and also in the other fields on web platforms such as music, movie or article recommendations. When using these services users mostly have filtration problems or they are not able to express their needs or demands clearly. Recommendation systems make the systems personalized as they analyze user information and filter the items the users actually want or like, hence they enhance user satisfaction and loyalty.

What personalized recommendation engines basically do is information filtering from an impressive large number of items by trying to find the ones that are interesting for the specific user [2]. Recommendation to users is done by predicting the items that would be highly rated by the user. More precisely a recommender system is any system that predicts the future preference of the user based on her past preferences.

Recommendation systems are divided into two forms Content Based Recommendations and Collaborative Filtering. Content based recommenders works based on the item features. Some examples of movie features can be listed as genre, actors who participate, its popularity of box office. User features can be demographic information such as age, gender, marital status or users' answers to a survey. Content based recommendation systems matches the users with products but collecting that kind of information is not easy and even in some systems it is not available [11]. The other type of recommendation engine Collaborative Filtering which is used in this project, it is based on similar user preferences. Collaborative Filtering is described in section 1.2.

1.2. Collaborative Filtering

Collaborative filtering systems are described as the systems produce recommendations for a set of items which a rating can be provided by users, where these items can be art, books, articles, movies, products and so on [3]. In collaborative filtering there is no need to have information about the items and users, the recommendations are made by examining item similarity and user similarity from data itself. It uses feedbacks given by the users to obtain the users who also have similar interests so that the hided interests of the users comes to the light [1]. The necessary information is just the historical user behavioral transactions including their product rating transactions. [11]

The user feedbacks are gathered from users' historical data, this data shows their opinions by the help of every movement they did through the system. Those can be ratings they gave to items, thumbs up or down events, like events, visit or send to a friend clicks and so on. By the end, user item and rating information is used on collaborative filtering. This information helps us to create the 'interactions matrix' which's values are the ratings that can be taken from explicit feedbacks, implicit feedbacks or from both of them. Explicit and implicit feedbacks will be discussed in detail in the next section.

Figure 1 is a representation of Collaborative Filtering for movie recommendation, user and movie interactions shown on the left and the rating matrix is shown on the right.

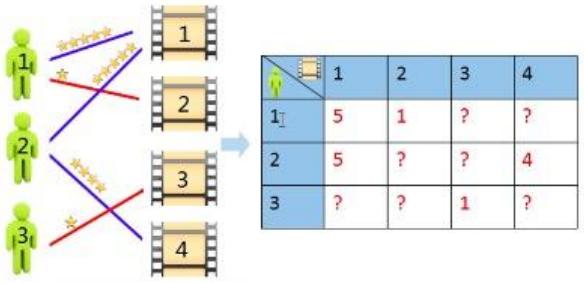

Figure 1: Collaborative Filtering Basics [4]

The collaborative filtering based on memory is used to calculate the historical information of the existing users in the system and the nearest neighbor of the target user. Then it uses the nearest neighbor to predict the degree of preference of the target user to the

item. User based algorithms make the predictions based on similarities as item-based algorithms are made based on similarities between items [3].

1.3. Implicit and explicit feedbacks

Explicit ratings can be explained as the feedbacks when the users are offered to provide an opinion about an item while implicit ratings are the ones inferred from users' actions. The examples of implicit feedbacks are purchase history, browsing history and duration, the patterns of search events or even mouse movements [11]. When a user visits an item page on web or in mobile application, the user may have an interest on it, besides when a user buys an item she might have had a stronger interest on it [3]. In addition to the star ratings 'like' or 'recommend friend' or 'thumbs up/down' selections are typical examples of explicit feedback [5].

Implicit feedback generally stands for the presence or absence of an action, as a result it commonly produces a densely filled matrix. For a TV recommender case, if we call the number of times the user u watched the show i as r_{ui} then $r_{ui} = 0.7$ shows that the user viewed 70% of the tv show, $r_{ui} = 2$ means that the show is watched twice. This way the rating is more truly represented. On the other hand, the dominant part of literature focus on explicit feedback mostly because of its complete information. Even so implicit feedback is necessarily to be used in many practical cases since users are often reluctant about rating items or it is unable to collect those in some systems. As long as data is collected implicit model is enough for user-based recommendation [11].

If we inspect a song recommendation case, comparing three songs A, B and C:

Song A: 1 play → It is not certain that the user liked it or disliked it.

Song B: 0 play → The user may not even know it exists.

Song C: 100 play → It can be said that the user liked it.

We can say that the model is more confident as declaring that the user liked song C than song A. The model is expected to catch this. The question that Implicit feedback models answer is 'Will the user like the item?' [23]

In the study which is made by Lee et al. on the wallpaper images recommender system, the purchase information was used as an implicit signal and, the release date of the items and the purchase time was used for determining the strength of the signal. For boosting

more recent actions to get higher scores, a time-based decay function was used [6]. This idea is used in our study as well.

1.4. ALS Algorithm On Spark Platform

1.4.1. Apache Spark

Apache Spark is a data processing framework which runs data in memory, RAM, by using a dataset concept called Resilient Distributed Dataset (RDD). It can run in stand-alone mode or with a Hadoop cluster by using it as a data source. When its speed of iterative calculation is compared with Hadoop MapReduce, Spark is faster [2].

Apache Spark brings high-level APIs in Java, Scala, Python and R, additionally supports high level tools such as Spark SQL and MLlib for machine Learning. The API for python is called PySpark, it is a language which is good for building machine learning pipelines and creating ETLs (Extract Transform Load). Spark Context object coordinates the application runs to process independently on a cluster, it is the main entry point for Spark functionality.

1.4.2. Matrix factorization and ALS

With Matrix Factorization (MF) the items and users are represented by vectors of factors. These vectors are inferred from item-rating patterns and the correspondence between the item-user factors brings us the recommendation [9]. Here the idea is splitting the matrix into two or more matrices instead of multiplying two matrices to get a new matrix. When the split matrices are multiplied the result is an approximation of the original matrix.

The matrix holds the users in one dimension, items in the other dimension and the ratings as the values. In MF the basic idea is to use latent factors for user preferences in a lower dimension space thus it can be said that it is an effective dimension reduction technique [22]. This idea can be seen better in Figure 2, the rating matrix is decomposed to user matrix (U) -user latent factors and item matrix (P) -item latent factors. The multiplication of those vectors gives us the rating on the rating matrix.3, R.

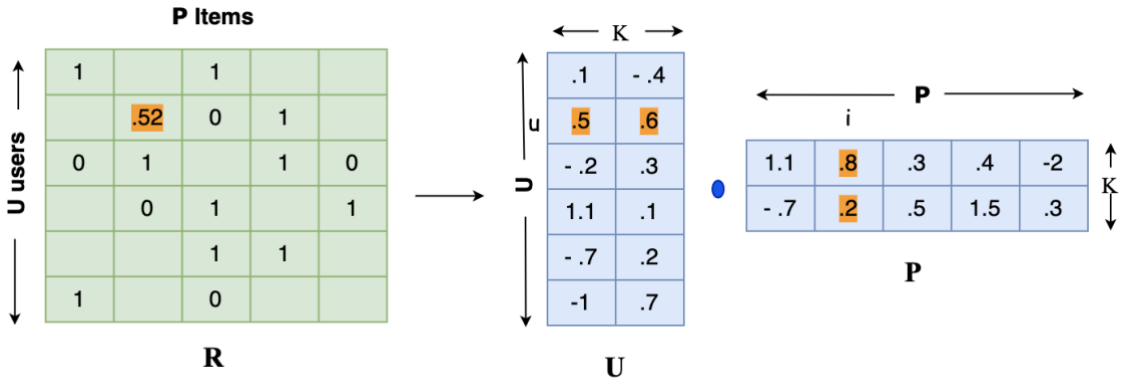

Figure 2: Matrix Factorization

For a basic understanding for matrix factorization we can have a look at the following figure. In the beginning the matrix have missing values but we can see that there are at least one value in every row and column. That lets us to factor the values in the matrix. When we factor this matrix into two factor matrices with 5 latent factors the result of the multiplication of the matrices gives the means to close values which existed in the matrix and the predictions for the empty cells in the matrix [22].

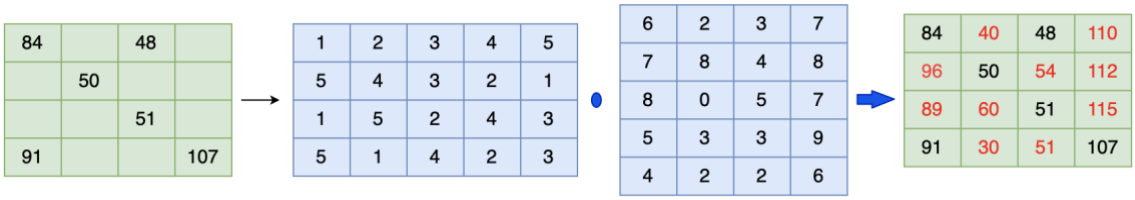

Figure 3: Matrix Factorization Basic Understanding

One of the popular algorithms which uses MF technique for recommendation system is collaborative filtering recommendation algorithm based on Alternating Least Squares (ALS) [2]. With matrix factorization sometimes the approximations of the original matrix can be negative values, this is not a logical result for a rating as a user cannot give negative ratings recommendation cases. ALS uses non-negative factorization for this reason.

Briefly, ALS decomposes the sparse rating matrix and creates the product of the user eigenvectors and item eigenvector matrices. Then the algorithm uses the least squares method and calculate the eigenvectors which makes the sum of squared residuals minimum.

At the end it uses user-item vector matrix for predicting the rating of the product given by the user. The blank cells are filled based on existing patterns.

ALS computes all item vectors separately from the other item factors and does the same for user vectors from the user factors. This is a massive task to do if there is a huge data. One of the advantages of ALS is, ALS's use of parallelization to achieve this task [10]. Its another advantage is its ability to handle looping over each single training case for implicit data where the training set is not sparse.

If we shortly describe what ALS does; to produce the approximation of the rating matrix R, ALS factors it into two different factor matrices. First it fills the factor matrices with random numbers and slightly adjusts the matrices until it reaches the best possible approximation. Step by step it keeps two matrices constant and adjust the other matrix.

ALS holds the R and U (user matrix) as constant then adjusts matrix P (item matrix). After then it multiplies the factor matrices P and U to obtain how far the predictions are from the original matrix. It uses Root Means Square Error-RMSE as an error metric.

The error metric RMSE gives us the answer off the question 'How far off the predictions are from the actual values?'. For the calculation of RMSE, only the values existed in the rating matrix are considered, it does not consider the missing values. It is an important point to keep in mind [22].

ALS continues this iteration until it is instructed to stop with the hyperparameters given. After the minimization of RMSE, ALS multiplies back the factor matrices and fills the missing ratings with predictions.

The ALS steps can be seen in Figure 4.

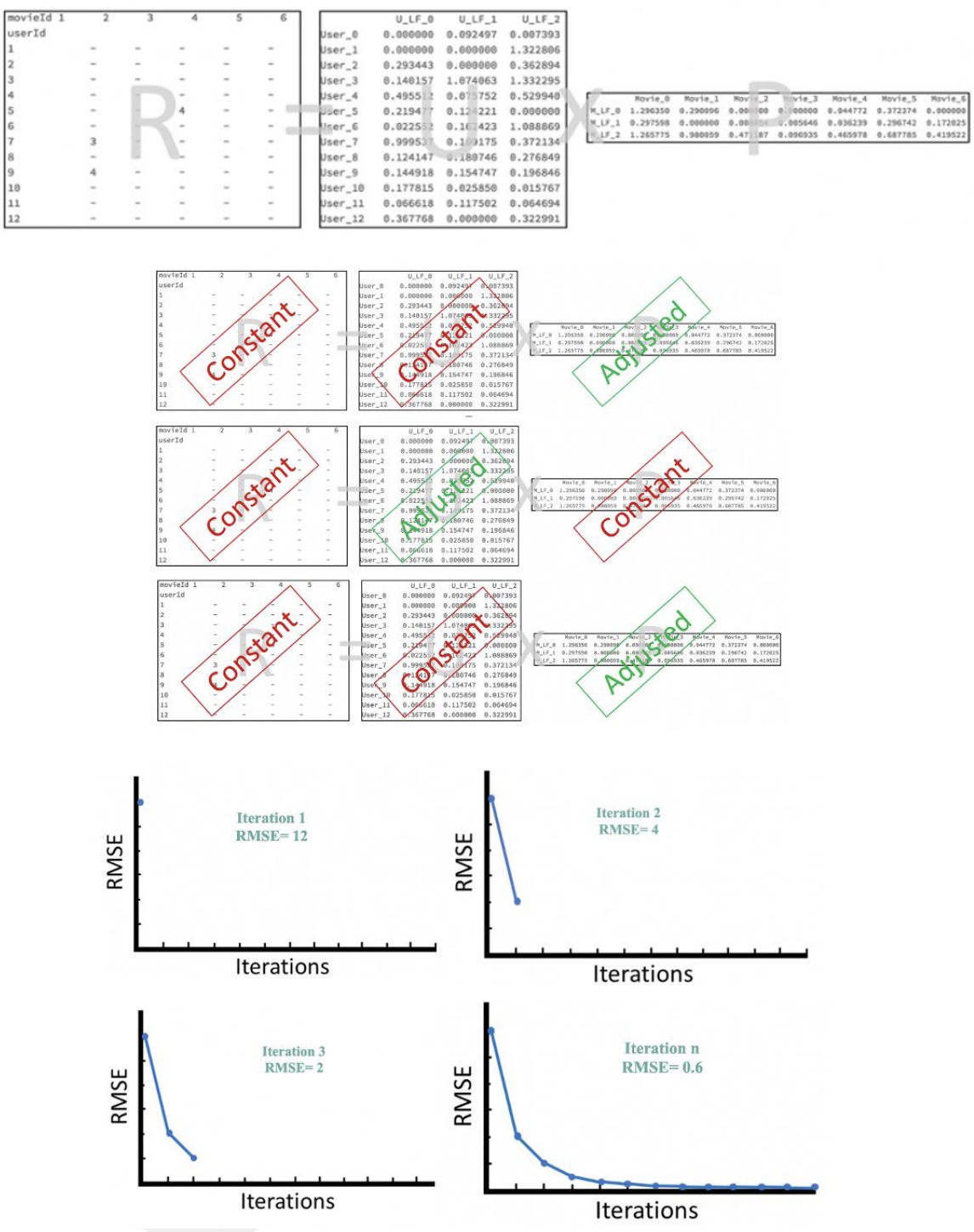

Figure 4: ALS method [22]

The mathematical representation of ALS method is the following equation where r_{ui} is user u's rating of item i, p_u is the user factor vector and q_i is the item factor vector. In order to learn p_u and q_i, the system minimizes the regularized squared error on the set of known ratings. In the equation, κ is the set of the (u, i) pairs for known r_{ui} values. The dot product $q_i{}^T p_u$ is the interaction between user u and item i, means the estimated rating. Here, lambda specifies the regularization parameter [9].

$$\min_{q^*,p^*} \sum_{(u,i)\in\kappa} (r_{ui} - q_i^T p_u)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

When the word comes to implicit ratings a couple of things change in the data preparation, implementation of the model and the evaluation method. We know that explicit ratings are provided by the users and implicit ratings are the ones inferred from user transactions. As we inspected in section 1.3 the more user do something the more she prefers it; such as listening to a song a hundred times, just once a time or never. For indicating confidence levels to make recommendations ALS can convert these numbers into scores. This time the matrix will include the number of times users interacted with the item and also the 0's for the songs that each user had not yet listened to. Additionally, to make meaningful recommendations if the item is preferred by a serious number of users the 0 interactions can be rated higher for that user-item rating or vice versa because of its popularity.

Another topic is evaluation of implicit models, RMSE is used for explicit models but in explicit models we had true measures and could match them with the predictions. In implicit models there are just the number of transactions so another metric called Rank Ordering Error Metric (ROEM) is used for evaluation in implicit models. As ROEM is used the sparsity (imbalance) does not become a problem like it might be in classification problems and ALS can make meaningful recommendations. In addition some strategies can be applied to improve results, for example for a movie recommendation case weighting can be done by specific user behaviors [22].

1.4.3. ALS Hyperparameters and Arguments

The hyperparameters of the ALS model are explained as follows:

Rank: Number of features to use (also referred to as the number of latent factors).

Rank helps to experiment meaningful groupings or categories for the items. Too many of too few ranks makes the model hard to categorize the items so different numbers must be tried to find which one makes sense. Cross validation can be used to find it easily as in the other hyperparameters.

maxIter: Number of iterations of ALS to run. ALS typically converges to a reasonable solution in 20 iterations or less.

To find the lowest RMSE, ALS iterates between the factor matrices by adjusting the values as we covered in section 1.4.2. This parameter must not be so high or low. If it is low the error will not be as less as it can be and if it is high the iteration duration takes much time to complete.

regParam: Specifies the regularization parameter in ALS.

It is mostly called as lambda in other algorithms. This hyperparameter is a number which is added to the error metric to prevent the model overfit or converge quickly [22].

alpha: Confidence parameter.

It is an integer value which is added to the confidence of the model. This hyperparameter is used only if it is implicit rating model. For example, if we say the user item visit event counts represent the user interest to the item, this value is added to model's confidence that a user liked a song or not.

The additional arguments are as follows:

nonNegative (True/ False): Tells to use positive or negative matrix factorization

Coldstart strategy (Drop/ NaN) tells the model to only use the users who have ratings in both training and test tests. If 'Drop' is given, the evaluation metric will be computed over the non-NaN data. Cold start problem is explained in the following section.

ImplicitPrefs (True/False): We need to tell spark whether our ratings are implicit or explicit.

1.4.4. Cold Start Problem

Cold start problem occurs when there is lack of data to be used by the recommender system, in this situation the recommender system cannot perform well. Up to Schafer 2007, there are three scenarios for cold start problems. First one is when the new item is added to the catalog, as it has no explicit or implicit ratings, the recommendations cannot be calculated for that item. This can also happen for unrated items which are actually good. To solve this problem recommending can be done with another technique such as content analysis or users can be randomly asked to rate unrated movies. Secondly the

recommendation problems occur when new user signs up to the system. In order to solve these problems before the user signs up, some information about user's taste can be collected and user can be asked to rate some items. If demographic information can be collected the initial recommendations can be done by using ratings of similar users or some non-personalized items can be recommended to the user such as most popular items. The last and the biggest reason for cold start is a new community. This problem can be solved by working on a subset of the community before serving for all or using different recommendation approaches [3].

1.5. Related Work

J.B. Schafer et al. 2007, tries to examine automated collaborative filtering to see the similarity of user opinions by using historical data. It is told that the collaborative systems can take different forms such as scalar - numerical ratings or ordinal ratings like strongly agree, agree, neutral, disagree; binary ratings like good/bad, unary ratings like purchasing or observing an item or having an opinion on an item like commenting without buying [3].

Jannach & Hegelich (2016) studied on a personalized item recommendation system for an online Mobile Internet games platform where the users purchase games. The platform had 1000 games in its catalog and the item ratings explicit feedbacks were given just by 2% of the total users, so additional implicit feedbacks were added to the study. The rating scale was taken between -2 and +2. For each view event medium score as 0 is given, and for every purchase event 1 (good) rating is assigned and explicit ratings are used to override the implicit ones. The indicated that only very few explicit ratings are possibly gathered from the users so implicit ratings have to be collected. They also stated that how to interpret the implicit feedbacks is an open question [12].

Lerche states that as aggregating the user actions, uniformly weighting is not generally convenient. The example of purchase action in an e-commerce case is stated as being a stronger indicator than an item visit action. It is also added that as there are many different signals as events in different applications assigning a different strength to each separate event is the idea to grade the feedbacks [5].

2. ABOUT THE DATA

2.1. Data Description

The data is taken from a startup company which runs its business in Turkey, the company serves to C2C market via its mobile application as well as their website. Users have the opportunity to sell or buy any kind of product including new and second-hand product or services. Based on the customer reviews, most of the customers have the aim of making value from the products they do not use anymore.

The data is gathered from four different database tables. As the first step, datasets are stored to Google Cloud Platform's Big Query data warehouse tool as separate tables to be easily examined, these tables are as below:

Table 1: Datasets

products

id	INTEGER	NULLABLE
title	STRING	NULLABLE
visibility	INTEGER	NULLABLE
purchase_price	INTEGER	NULLABLE
price	INTEGER	NULLABLE
shipping_included	BOOLEAN	NULLABLE
visits	INTEGER	NULLABLE
label	STRING	NULLABLE
status	STRING	NULLABLE
listed_at	TIMESTAMP	NULLABLE
brand_id	INTEGER	NULLABLE
category_id	INTEGER	NULLABLE
owner_id	INTEGER	NULLABLE

orders

Field name	Type
id	INTEGER
payment_total	INTEGER
product_id	INTEGER
purchaser_id	INTEGER
created	TIMESTAMP
status	STRING

reviews

Field name	Type
review_id	INTEGER
rating	STRING
owner_type	STRING
order_id	INTEGER
recipient_id	INTEGER
created	TIMESTAMP

users

Field name	Type
id	INTEGER
gender	STRING
date_joined	TIMESTAMP
last_activity	TIMESTAMP
on_vacation	BOOLEAN

likes

Field name	Type
id	INTEGER
product_id	INTEGER
user_id	INTEGER
created	TIMESTAMP

categories

Field name	Type
id	INTEGER
category1	STRING
category2	STRING
category3	STRING

Products: Product listing details

Orders: Order details

Likes: User product likes

Reviews: User ratings are given as a seller or as a buyer after orders

Categories: The product categories

Since we needed ‘user_id’, ‘product_id’ and ‘rating’ features to use in our ALS algorithm, we used these tables to catch the user-product interaction.

2.2. Data Preparation

2.2.1. Creating Data Set

We have to inspect all provided tables, the processes are totally done by using SQL by using Google BigQuery tool on Google Cloud Platform. We started from products table, this table is singularized by product_id and title information. Here we noticed that some characters such as ‘,’ and ‘/n’ needed to be replaced by null character in order to keep the products singularized and readable when recommended. The number of distinct products is 3,920,545. We used this table as creating the final rating table also we read product titles from this table when making recommendations.

For the reason that we wanted to use explicit and implicit feedbacks together we moved on preparing a rating table by using the ‘reviews’, ‘likes’ and ‘orders’ tables. Firstly, all of those 3 tables are inspected separately.

‘Like’ data set is the one with the greatest number of user-product interaction. The data had duplicates, so it had to be singularized by taking the latest created event per user and item. There are 12,318,747 rows in this dataset and 154,635 distinct users created like events for 3,717,572 different products between the dates 2017-12-31 and 2020-07-03.

Top 5 users who made most like events are as follows:

Table 2: Top 5 users with most like events

Row	user_id	like_count
1	48390700	52344
2	146150844	34021
3	300132217	29105
4	212291497	25489
5	159270300	20135

If a user purchases an item, review option is opened for this order so the buyer can rate the order as ‘great’, ‘good’ or ‘poor’. This is an explicit feedback, and this was collected as a grade from 1 to 3. The data is filtered by owner_type column as ‘purchaser’ and joined with the orders table on ‘order_id’ column in order to be sure that it is connected to an order and the last created event is selected. As discussed before the ‘reviews’ table consists the ratings given to the seller in connection to orders made. This information was used as it was given to the product. The ‘reviews’ are transformed into a new form by using the ‘order_id’ column to connect it through the ‘orders’ table and get the ‘product_id’ information. By this way the user-product match could be made. Furthermore, the reviews ‘great’, ‘good’ and ‘bad’ are transformed to 3, 2, 1 as ratings. The ‘purchaser_id’ in the ‘reviews’ table is taken as the ‘user_id’. The total review number is 811,295.

The ‘orders’ table is filtered by its ‘status’ column and the rows 'cashed_out', 'completed', 'payment_done', 'shipped' are selected. Since the last recorded step of an order changes in different orders, we took the most recent row in order to its ‘created_date’ information for each unique ‘product_id’, ‘purchaser_id’ couple. We did not prefer to include ‘refunded’ orders but in the other hand the refunded orders may have a review.

In the study ‘likes’ and ‘orders’ are taken as implicit feedback, this information is added to review scores by determining a couple of rule sets. Lerche indicates that different types of user behaviors are observed in different applications. In order to obtain ratings from these feedbacks, unique strengths can be assigned to those signals to reach different levels of graded relevance. Here the question is how to transform those feedbacks into numerical values. This is certainly challenging and most of the times is done by following an arbitrary manner [5]. Therefore, different methods are developed for different data sets.

We need to create the ‘rating’ dataset to catch the user-product interactions. We had three different datasets to use to start preparing the ‘rating’ dataset: ‘reviews’, ‘orders’ and ‘likes’. ‘Orders’ and ‘likes’ consisted user ids and product ids. All three tables are examined to see if there were any duplicates or anomalies.

2.2.2. Obtaining the Scores by Using Explicit and Implicit Feedback

Lee at all, states that user's current choices are better reflected by more recent purchases and recent launched products appeals more to users [6]. With this idea for 'orders' and 'likes' data, using time as determining the strength is decided to reach more realistic ratings.

The time intervals are obtained as 6-month divisions, since the data starts from 2018 January and ends 2020 June it was possible to divide it to 5 divisions. The Covid-19 outbreak was taken into account in this approach as it effected the company's business negatively. So that the last part of the time division was based on dates between March 2020 and June 2020. The first two months of 2020 was added up to the fourth division which was based between 2019 June and 2020 February. The closest time interval when the 'order' or 'like' event occurred was taken as the highest rating of 5 and the remaining continued till the latest event took the value of 1.

Primary Ratings for Orders and Likes:

5: The events between 2020 March – 2020 June

4: The events between 2019 July – 2020 February

3: The events between 2019 January – 2019 June

2: The events between 2018 July – 2018 December

1: The events between 2018 January – 2018 June

In summary, we had 3 kinds of ratings in this step:

Reviews: 1-2-3

Orders: 1-2-3-4-5

Likes: 1-2-3-4-5

As a next step we started creating our final 'rating' dataset. The order information and the reviews are full joined in order to get the interactions data from each set and next, 'like' table is also full joined with these interactions. At the end we had a dataset with 13,212,319 rows for 177,250 users and 4,193,337 products.

Finally, the rating score assignments are made through the rule based given weights. Reviews are the strongest events and then the order comes. If a user bought a product it means that she liked it plus she needed it, in the case of order like event loses importance.

Figure 5 shows the weights basically assigned; the user gave a review the rating is converted to 1-5 rating system from 1-3, if the user made an order of a product without any review and like the order rating value is multiplied by 0.9 and if the user only liked a product the weight is 0.8.

Cases	Review	Order	Like	Rating
Case 1	1,66			1-5
Case 2		0,9		1-5
Case 3			0,8	1-4

Figure 5: Feedback Weights

In the case a user practiced all three events the rating will have the review value. Figure 5 shows the all the cases of event occurrences, given weights and the rating intervals. These weights are all determined experimentally by considering the user buying habits. The scores are rounded down or up to the closest integer number in order to prevent having a wide range of rating values and assign ratings with integers between 1 to 5 to get better results.

For a user and product interaction the user might have liked the item or ordered the item and might then gave a review on it. The first two interactions are considered as positive, but we can get the real feedback from the review. If the review is ‘awful’ the rating score should go lower. For the same item, if it is a single product, this is a highly possible case in C2C marketplace, there is no opportunity for another user to give a review score because the user made the purchase. Another user may just have the opportunity to like the item, so we cannot give like a total weight of 5, the weight should be less.

Lerche states that mapping the separate feedbacks such as viewing, buying an item into a single linear rating score is hard to generalize, sometimes it may be impossible, and a respective domain is needed. A short dwelling time is interpreted as negative feedback as

the user seems not to be interested in the item but there is a probability that she already knows the item [5].

2.3. Exploratory Data Analysis

For gaining insight from the data and use it during the decisions for creating rules for the rating scores the data is analyzed at different aggregation levels. In the section 2.3.1 we inspected the feedback types of the users, for example we wanted to see how many users liked and bought an item then gave a review on it. In section 2.3.2 shows the number of each rating value and the last section 2.3.3 the inspection is done from the product side and the product-user interaction counts are examined.

2.3.1. The Feedback Types

The feedback types are important as determining the weights of the initial ratings as finding the final ratings. Figure 6 shows all cases that can occur for the events review, order and like events. 'Only_like' case had been excluded from the graph whereas it had around 12 million rows. That shows us the main part of the data set is formed of the 'like' events and in that case giving the 'like' event less weight was better as we did while creating the final ratings. It can be seen in Figure 7 that 'only_like' events are majority of the data.

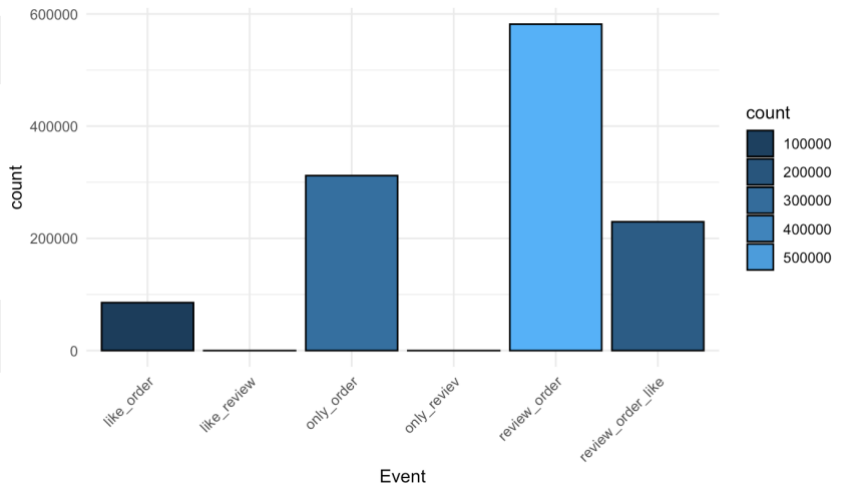

Figure 6: The Feedback Types - 1

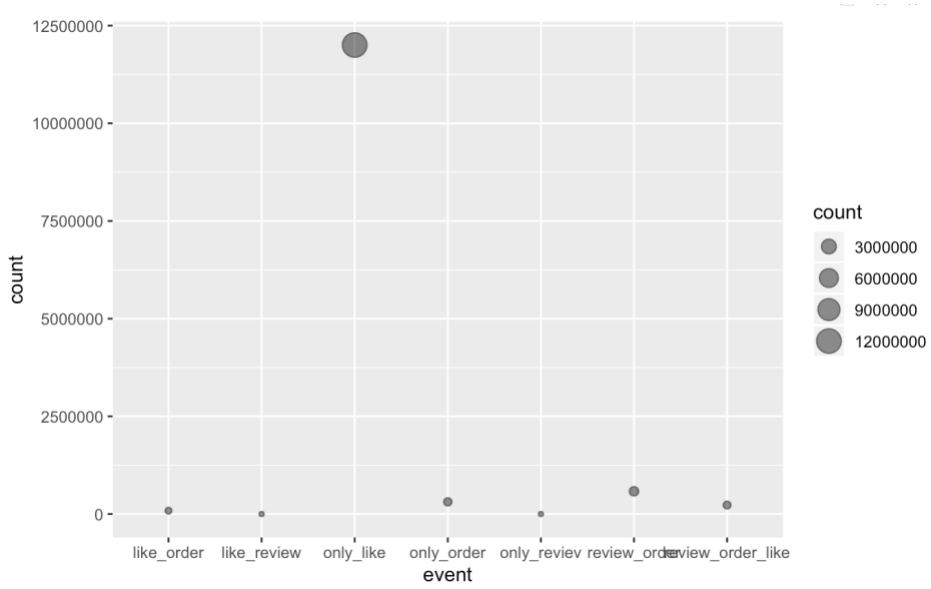

Figure 7: The Feedback Types - 2

2.3.2. Rating Distribution

After the data preparation means creating the rating data, the distribution of the rating counts of detailed version can be seen in Figure 8. The number of ratings 1 is 1,166,675; 2 is 5,973,928; 3 is 4,910,513; 4 is 389,411 and 5 is 771,792

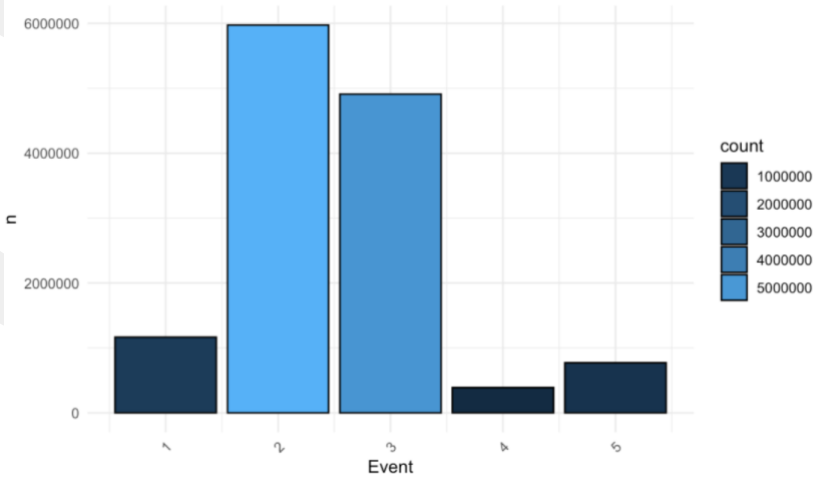

Figure 8: Rating Distribution

2.3.3. Product Interaction Frequency

The number of total products in the data set is 4,193,337. From the data that we have in hand the popularity of the products can be obtained from the user-product interaction. This is one single row for one user, Figure 9 and Figure 10 show that how many interactions have occurred for a single product. The distribution is skewed between 1 and 350. When it is checked how many products had just one interaction, the number was 1,758,521.

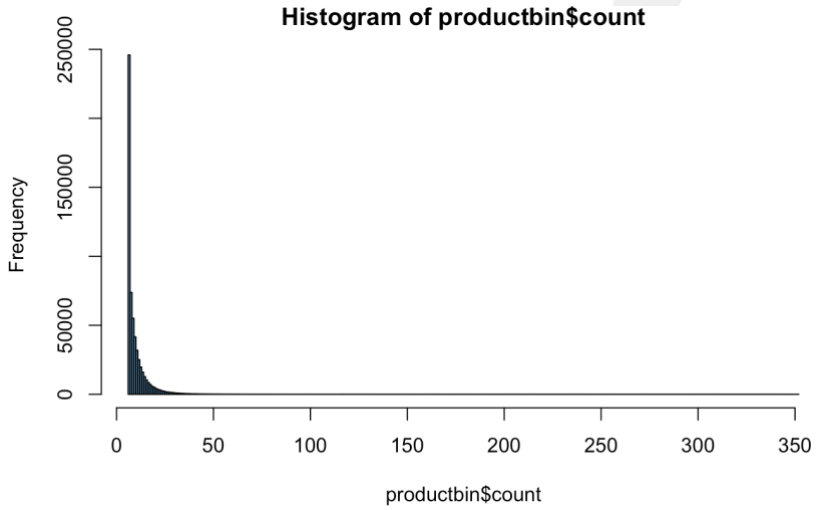

Figure 9: Product Interactions - 1

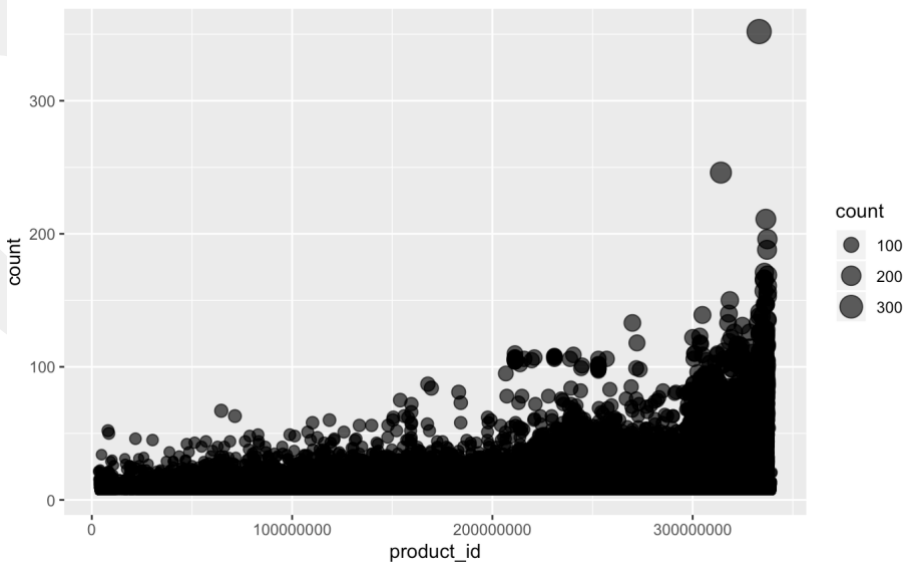

Figure 10: Product Interactions - 2

3. PROJECT DEFINITION

3.1. Problem Statement

E-commerce have been rapidly taking its place in people's lives, an increasing part of the population has started to make online purchases even for small piece of products. The secondhand marketplaces give the opportunity to serve the pieces to others who need them instead of keeping it in the hatches or throwing it away. Furthermore, C2Cs serve as the secondhand and antique stores with lower prices. The sellers meet in the same platform as a consequence the product variety and density rise. Recommendation systems are widely used in helping people to deal with this information overload. With user-based collaborative filtering catching and rating the user interactions then finding the similar users and recommend the items which the similar users like is possible. At this point using as much data as it can and weight those feedbacks are important and the hardest part of the studies, it is mostly done empirically.

3.2. Project Objectives

The objective of the project is to develop a machine learning model to predict the user ratings from the user similarities by using collaborative filtering using the ALS algorithm. As the company serves in the C2C business environment most of the products are unique, and reviews can be done after purchase thus the data is sparse. Order and like feedbacks are combined with reviews data and the ALS algorithm is run to find the non-interacted user and products with different hyperparameters. The results are used for reaching the top ranked items to be recommended and top 10 recommendations are gathered for users.

3.3. Methods, Tools, and Techniques

As the first step, the csv data files are gathered and stored into Google Cloud Storage, then they are created as native tables in Google Cloud BigQuery tool. It is a serverless data warehouse service which supports ANSI SQL for querying.

The data cleaning and data preparation steps are completed in BigQuery as explained in Section 2.2. This part is the most challenging part of the project as data preparation part generally is the highest percentage of work done in most of the machine learning projects. After that the final data set is stored in csv format to be used in PySpark in PyCharm python language IDE. Figure 11 summarizes this pipeline.

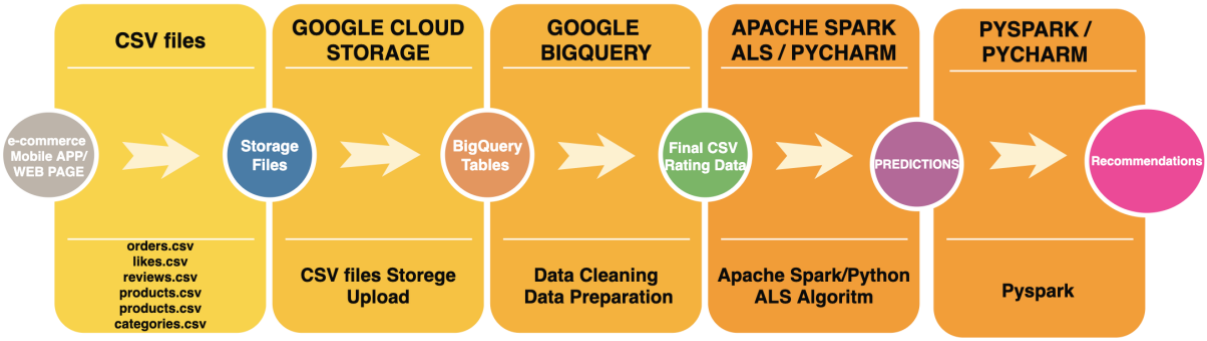

Figure 11: The Project Pipeline

The ALS Algorithm is implemented with different hyperparameters with grid search method and cross validation is also used with 5 folds. These steps are explained in section 3.4. After the experiments with different hyperparameters, the evaluation metric Root Mean Square Error (RMSE) value and the recommendation results for specific users are observed. We used RMSE objective metric to evaluate the accuracy of the prediction. The smaller RMSE means the higher the accuracy. RMSE is widely used for the evaluation of recommender system models, it is defined as following where r_{ui} is user u's rating of item i, r hat is the prediction and Np is the total number of predictions [4]:

$$RMSE = \sqrt{\frac{1}{N_p}\sum_{u,i}(\hat{r}_{ui} - r_{ui})^2},$$

3.4. Evaluation

As discussed in section 1.4.3, different hyperparameters; rank, maxIter and regParam; are tried sequentially with grid search using 5 folds cross validation for the training set. Here the aim is to reach the lowest RMSE value by remembering that RMSE is the difference between the predicted and the actual values for the ratings which were not empty in the beginning.

Following figures show some of the grid parameters which had been run for the rating set. The grid search is not run just one time, it ran for a set of hyperparameters then by using the best hyperparameters of this run the values are changed and another grid search is run. By this way the best results are aimed to be reached. Using a grid search with many parameters is not possible because of performance issues, even when the iteration number is raised the performance problems occurred because of the size of the data.

The results of the first grid search can be seen as follows. Following run tries 8 different runs to obtain the best parameters and the RMSE is 0.808

```
param_grid = ParamGridBuilder() \
             .addGrid(ALS.rank, [10,15]) \
             .addGrid(ALS.maxIter, [15,20]) \
             .addGrid(ALS.regParam, [ .08, .1]) \
             .build()
```

```
====>>> Best model metrics:  {
Param(parent='ALS_e2559a97cb93', name='rank', doc='rank of the
factorization'): 15, Param(parent='ALS_e2559a97cb93', name='maxIter',
doc='max number of iterations (>= 0).'): 20,
Param(parent='ALS_e2559a97cb93', name='regParam', doc='regularization
parameter (>= 0).'): 0.1}
It took my system 2901.10s to find the best params

====>>> RMSE:  0.8085407876532787
```

Up to the best parameters taken the hyperparameters are changed and run started again. The reason of this method is that not more parameters could be given because of run performance with 5 folds. This process is repeated for times and times, the 20, 25, 30, 50, 100, 120 for ranks; 10, 20, 25, 30 values for maxIter and 0.01, 0.05, 0.5, 0.8, 0.1, 0.15, 0.2 for regParam are tried with different combinations. Some of the results with minimum RMSEs as the best model metric outputs are listed in the following table.

Table 3: A list of run hyperparameters and their RMSE outputs

rank	maxIter	regParam	RMSE
20	20	0.01	1.126
20	25	0.01	1.14
20	20	0.05	0.871
20	25	0.05	0.858
15	20	0.1	0.808
30	20	0.1	0.806
30	10	0.1	0.82
20	20	0.1	0.807
50	20	0.1	0.805
100	20	0.1	0.804
20	20	0.15	0.791
30	20	0.15	0.791
30	20	0.2	0.787
20	25	0.2	0.78

Finally, we reached minimum RMSE value of 0.78. We know that RMSE of 0.78 means that on average the model predicts 0.78 above or below values of the original ratings matrix. We got an understanding of the results of the model and have some confidence that it will recommend items to relevant users.

The ALS algorithm gives us the rating predictions for every user-item couple so that top 10 highest predicted scored products are listed. This way 10 recommendations for all users are generated.

In every case we can reach the minimum RMSE but obviously it is a subjective metric. We cannot totally say it is sufficient for meaningful recommendations. We need to know if the recommendations make sense. We need to test it with a set of users to see its consistency and also it is necessary to check the returning of them in live environment and reshape the model. To see if the model generates reasonable results, we can check the recommendations of some users.

In a presentation made by Watsons, she states that before production use of the model, the test of results are made as human checking, in movie case it is done up to the genres. She admits that it is not possible to check everyone's predictions that way. She states,

for real world results it is very hard to incorporate into a system in practice, if you work in production it would be nice to have a checking system for the output. [23]

In our product recommendation case, we can investigate the results for some users, check their item categories and compare those with the categories they rated most. The user with id of 146150844 has ratings for 34,022 products. The top 10 recommendations which the model gives for this user with ratings for the user and the product categories are as follows:

Table 4: Recommended products and their categories for user 146150844

	Product_id	Product_Name	Score	category1	category2	category3
1	303373426	Electronic Books	4,32247591	Books & Stationary	Foreign Books & Dictionaries	Other
2	309826972	Nazar Bracelet	4,15123558	Woman	Jewelry	Bracelets
3	303732845	Aston Martin Sun Glasses	4,09914875	Man	Accesories	Glass, Sun Glasses
4	301832432	New Zara Blazer	4,08713913	Woman	Suits & Jackets	Blazers
5	245182448	5 sectiones Bag	4,074050903	Woman	Bags	Backpacks
6	325924169	Wooden Plate	4,04712534	Handiwork & Arts	Woodwork	Wood items
7	260294921	Rezerved ♥ Beyza Becca liquid brightener	4,027101994	Cosmetics	Make-up	Face
8	294687598	Büyük Yumurta	3,996158361	Hobby & Collections	Collections	Porcelain
9	306427702	4-5 Age Poncho Towel	3,987999916	Baby & Child	Boys	Other
10	255689824	BİODERMA ATODERM PP GEL	3,978824139	Cosmetics	Woman Body Care	Shower Gel & Soaps

For a comparison, firstly we took the average ratings per item category for Category1. In Table 5, the bold categories are the ones which are in the top 10 recommendations. Here we know that the average rating is not totally the sign of the user interest.

Table 5: Product Categories by average rating of products for user 146150844

user_id		avg_rating	category1
146150844	1	26.358	**Handiwork & Arts**
146150844	2	2.537	Food & Market
146150844	3	24.286	Hobby & Toys
146150844	4	2.306	**Hobby & Collections**
146150844	5	22.338	Books
146150844	6	21.977	**Books & Stationary**
146150844	7	21.467	Home & Decorations
146150844	8	20.464	**Baby & Child**
146150844	9	20.366	Other
146150844	10	20.052	Spors & Outdoors
146150844	11	19.677	Game & Consoles
146150844	12	19.361	**Woman**
146150844	13	18.838	**Man**
146150844	14	18.578	Electronics
146150844	15	18.277	**Cosmetics**
146150844	16	1	Arts & Collections

Secondly, we investigated the user-product interaction counts deeper up to Category2. The interactions more than 200 times with the rest of categories in the top 10 recommendation list included to the end, can be seen in Table 6. There are 205 different categories in Category2. We can see that the recommended categories are parallel to the interaction counts. The investigated user is interested in products for hobbies, handiworks, books, woman accessories and clothing and the user tends to buy products for boys and for baby girls.

Table 6: Recommended Product Categories by User-Product interaction for user 146150844

	interaction_count	category1	category2
1	**1623**	**Baby & Child**	**Girls**
2	1562	Home & Decorations	Kitchen & Dine
3	1460	Woman	Top wear
4	1337	Baby & Child	Toys
5	1299	Home & Decorations	Home Decorations
6	861	Woman	Shoes
7	**776**	**Baby & Child**	**Boys**
8	772	Woman	Dresses
9	561	Woman	Bags
10	504	Electronics	Cell Phones
11	**445**	**Woman**	**Jewelry**
12	**430**	**Woman**	**Accesories**
13	390	Other	Other
14	387	Man	Top wear
15	380	Sports & Outdoors	Outdoors
16	368	Woman	Outwear
17	350	Hobby & Collections	Antique products
18	330	Woman	Bottom Wear
19	327	Handiwork & Arts	Needlecraft
20	**307**	**Cosmetics**	**Make-up**
21	306	Man	Accesories
22	274	Man	Shoes
23	271	Books & Stationary	Literature
24	265	Handiwork & Arts	Knitting
25	**264**	**Books & Stationary**	**Foreign Books & Dictionaries**
26	251	Baby & Child	Other
27	230	Home & Decorations	Bedroom
28	**218**	**Hobby & Collections**	**Colletions**
37	**123**	**Woman**	**Suits & Jackets**
85	**32**	**Cosmetics**	**Woman Body Care**
114	**15**	**Handiwork & Arts**	**Woodwork**

Now we can check the same results for a second user, user id of 333166679 is the fifth user who made most interactions. The top 10 recommendation the system gives for this user is shown in Table 7. It seems this user is mostly interested in Home & Decoration categories.

Table 7: Recommended products and their categories for user 333166679

Recommendation	Product_id	Product_Name	Score	category1	category2	category3
1	337460150	Old Season Lcw Home Drying Cloth 2 Pairs unused	4.7482271	Home & Decorations	Kitchen & Dine	Kitchen & Table Cloths
2	336802225	Original Huawei Earpods	4.7167315	Home & Decorations	Other	Other
3	337710758	Marks&spencer Drying Cloth	4.3667702	Elektronik	Tv and Sound Systems	Earpods
4	337185750	Black marble cutting board	4.3076524	Home & Decorations	Kitchen & Dine	Pots & items
5	337397242	COPPER PANS	4.1496872	Home & Decorations	Furniture	Living Room Furnitures
6	337082197	Bedside jug	4.0136790	Home & Decorations	Home Decorations	Decorative Pillows
7	337770945	Starbucks Mug	3.9761950	Home & Decorations	Kitchen & Dine	Other
8	337859792	Pouf from old hand woven rugs	3.9405350	Home & Decorations	Kitchen & Dine	Coffee;Tea & Espresso
9	337879226	Ikea Cushion Cover	3.9236736	Home & Decorations	Kitchen & Dine	Small tools and supplies
10	319203686	Paşabahçe Vase 6	3.9094944	Home & Decorations	Kitchen & Dine	Kitchen & Table Cloths

There are 392 distinct categories up to Category3 which the user interacted. The categories that user interacted more than 50 times are shown in Table 8. The bold products are the ones in top 10 recommended products list. The last product is interacted just once but it is recommended. Of course the reason of this can be the user similarity, we know that the interactions and ratings are important but ALS uses the similarity in the user-product matrix. So, if a similar user rated this item, it is normal that the system can recommend it.

Table 8: Recommended Product Categories by User-Product interaction for user 333166679

Rownum	user_id	interaction_count	category1	category2	category3
1	**333166679**	**474**	**Home & Decoration**	**Kitchen & Dine**	**Coffee;Tea & Espresso**
2	333166679	408	Home & Decorations	Kitchen & Dine	Food Service & Supplies
3	333166679	260	Home & Decorations	Home Decor	Other
4	333166679	246	Home & Decorations	Home Decor	Candle & Candle Holders
5	**333166679**	**162**	**Home & Decoration**	**Other**	**Other**
6	**333166679**	**151**	**Home & Decoration**	**Kitchen & Dine**	**Pots & items**
7	333166679	146	Home & Decorations	Home Decor	Entrance Carpets & Rugs
8	333166679	136	Home & Decorations	Home Decor	Soıvenirs
9	333166679	122	Home & Decorations	Home Decor	Decorative Lamp; Lighting
10	**333166679**	**122**	**Home & Decoration**	**Kitchen & Dine**	**Small tools and supplies**
11	333166679	110	Home & Decorations	Kitchen & Dine	Storage & Organization
12	333166679	104	Other	Other	Other
13	333166679	102	Home & Decorations	Kitchen & Dine	**Food materials**
14	**333166679**	**98**	**Home & Decoration**	**Kitchen & Dine**	**Other**
15	**333166679**	**94**	**Home & Decoration**	**Home Decor**	**Decorative Pillows**
16	333166679	78	Home & Decorations	Home Decor	Trinket
17	333166679	75	Home & Decorations	Home Decor	Vase
18	**333166679**	**72**	**Home & Decoration**	**Kitchen & Dine**	**Kitchen & Table Cloths**
19	333166679	65	Home & Decorations	Bath	Towel & bathrobe
20	**333166679**	**63**	**Home & Decoration**	**Furniture**	**Living Room Furnitures**
21	333166679	61	Home & Decorations	Storage & Arrangement	Basket & Boxes
22	333166679	59	Home & Decorations	Bedroom	Duvet Cover & Sets
23	333166679	53	Home & Decorations	Home Decor	Photo Album & Frames
276	**333166679**	**1**	**Electronics**	**Tv and Sound Systems**	**Earpods**

If we have a look at the categories order for average ratings, as shown in Table 8 the Home & Decoration is the fifth most rated category.

Table 9: Product Categories by average rating of products for user 333166679

user_id		avg_rating	category1
333166679	1	3.1818	Books
333166679	2	3.1346	Man
333166679	3	3.0714	Handwork & Arts
333166679	4	3.0559	Woman
333166679	**5**	**3.049**	**Home & Decorations**
333166679	6	3.044	Cosmetics
333166679	7	3.0426	Books & Stationary
333166679	8	3.0408	Baby & Child
333166679	9	3.0345	Sports & Outdoors
333166679	10	3.0294	Food & Market
333166679	11	3.0284	Electronics
333166679	12	3.026	Other
333166679	13	3.0258	Hobby & Collections
333166679	14	3	Hobby & Toys
333166679	15	3	Game & Consoles

4. RESULTS

In this study three transaction types are used: ‘order’, ‘like’ and ‘review’. In reality these transaction types are a little part of user interactions. As we know that using explicit feedbacks and implicit feedbacks together gives us more accurate results, some other important transaction types which would be used are page view, product search, share, recommend to a friend, add to cart and etc. The three transactions which could be provided from the company are used to create a user-product rating matrix by using different weight assignments and a basic logic with a time-based approach. Those studies are made in the background and the most basic one is ended up to be used in the study because the results were much the same with complex ones. The reason of this is that only three events were used, if there were more events to be used as preparing the rating data the weight assignments would be more challenging.

The user-product matrix is implemented to the ALS algorithm and different hyperparameters are tried to get lower RMSE results by using grid search with 5 fold cross validation. The lowest RMSE of 0.78 is reached by taking rank as 20, maxIter as 25 and regParam as 0.2. Using these hyperparameters ALS model gives us the predicted ratings for the empty user-item couples in our source data. Highest rating means highest interest to a product so recommendations are made through the highest rated products. We chose the top 10 highest predicted products per user to show to each user as recommended items.

We know that the evaluation of the recommendation does not totally implies for lowest RMSE, in real world the results are successful if there is a rise in the order numbers related to the recommendations. After all, before we take the project to the production environment we should check if the recommendations are matching with the user interests. As we examined in the evaluation part, the way to understand this can be investigating the user product category preferences. We examined a set of users and shared two of them in the study. The recommended products seem coherent to the users’ category preferences up to their interaction counts with the categories or average ratings given to a specific category.

The recommended products are compared with the product categories which the user mostly interacted and averagely highly rated. The comparisons show that the recommendations are mostly coherent to user preferences. The recommended product categories match up with the users’ mostly interacted product categories and averagely

highest rated products categories. For example, the first user has the most probability to be a married woman with a child/boy and a baby girl. She tends to buy woman category products including cosmetics plus hobby and crafts categories for herself, books for her son and baby products for her daughter. The top 10 recommended products consist woman products, hobby-craft-collection and boy category items and also man accessories in parallel to these preferences. The other user looks totally interested in Home & Decorations categories as he/she seems to decorate a new house. The recommendations are consistent with his/her preferences as we know that these are determined from the similar user interactions by ALS algorithm.

The real accuracies of the results can be observed by using these recommendations in the live system. If the recommended products are found interesting by the user, they would visit the product pages by clicking on the recommendations or make orders so the recommendations can be evaluated and the model can be updated.

5. CONCLUSION

We implemented ALS model by using a data set created by weights for different cases. The ‘like’ and ‘order’ feedbacks are weighed through a time-based approach. This way of data preparation is a kind of empirical approach besides it leaded us acceptable results. Taking the ranking value as a combination of explicit and implicit rating richens the data and lets the training data set be wider. However, the methods to combine the feedbacks varies for different domains and they are hard to determine. The RMSE is found as 0.78 and can be considered as a good result for a 1- 5 scale ranking.

In 2006 the DVD renting company Netflix made a competition to improve its recommender system and this competition sparked a great rise for the collaborative filtering field. The dataset consisted more than 100 million ratings for 500.000 customers and 17.000 movies. The competitor teams predicted ratings for a test set and the RMSE results are compared by Netflix. At the end of the contest the Netflix system reached 0.9514 of RSMSE value, the grand prize winner model reached 0.8563 of RMSE with an implementation of matrix factorization called funk SVD [9]. This method uses L1 regularization as ALS uses L2 regularization. In this study the data was ready to use and the aim of all competitors was to improve the RMSE by using the same user-product-rating data. In our study the ratings are generated by us with a set of rules. The most challenging part is using the explicit and implicit feedbacks together and create a logic for generating the ratings per user-item couples. This is a necessity because for a movie case we can use user ratings directly and we can get successful results but in an e-commerce product recommendation case or music platform song recommendation case the ratings are not enough, even music platforms do not have any rating feature. We need more user interactions to get better results and our first job is to create a successful rating generation logic. Doing this is only possible by well knowing the data, the economical impacts in different time periods, the strength of transaction types over user behaviors and so on. Google Cloud Storage helped us load csv files to cloud environment and Google Cloud BigQuery tool let us to create them as database tables and examine them easily by using sql queries. Moreover, we had the opportunity to easliy create the rating and product tables with a set of queries and converted them in csv files again to be used in Python development as a source data.

The ALS algorithm is a widely used algorithm for recommendation cases, it is provided by Apache Spark. It is easy to implement after the data preparation and gives good results. It can give good results even for binary rating datasets [22]. Its use of matrix factorization gives us the empty rating predictions and it is advantageous with its parallel fashion runs so it can handle large data sets. Our data has 13,212,319 rows and could perform well enough in a basic scale server.

The recommendation projects are always improvable. We see that the source datasets and data preparation are as important as or sometimes much more important and challenging than finding the best hyperparameters. Starting from low number of user transaction variety the user interactions can be examined, added to the model and taken benefit of. The companies always work on new features for their web sites or mobile applications. Understanding each customer better and better means that there is a need of more data. The new features can be joined to the recommendation dataset so the weights should be revised up to their importance. Sometimes the results in the live systems show the weights given are not appropriate so some transaction types can be eliminated or their weights can be revised. The results are improved with time, up to the feedbacks of the users. Which products are showed as a recommendation to a user, which ones are viewed after the vision, which ones are liked, which ones are shared via social media or personal messaging, which ones are added to cart or which ones are ordered and so on. A company may have more than 50 transaction types according to its size. All these transactions construct the results and evaluation of these show the success of the recommendation system.

Further study can be done by gathering more data such as product views and seller's previous relation with the purchaser as these surely are good indicators while obtaining user preferences and history. In the study, feedback weights were tried to be assigned scientifically by using some methods found in the literature but different approaches can be tried as a future study, the data preparation can be stated as the most important part of the project. The C2C data in its nature mostly lets one order for one product and also the sellers are observed to open personal product lists to be sold only to one person which they call 'reserved'. This makes it harder for the recommendation system to find user-product similarities. Moreover, for a better performance a server with a parallel procession environment like Hadoop can be used for runs, this brings ALS's parallel run feature more effective.

The source data is our basis to construct the model, it means a lot so the weights given to the transaction type or in other words how a transaction type is taken into account means a lot. This effects the results and better RMSE values can be reached. The real results can be evaluated after it is taken to production environment and the feedback of the users are collected.

REFERENCES

[1] TIAN Hong-peng , CHEN En-jie, ”Research on Collaborative Filtering Algorithm Based on Spark Platform”

[2] Li Xie, Wenbo Zhou, Yaosen Li , ”Application of Improved Recommendation System Based on Spark Platform in Big Data Analysis ”

[3] Schafer, Ben & J, Ben & Frankowski, Dan & Dan, & Herlocker, & Jon, & Shilad, & Sen, Shilad. ”Collaborative Filtering Recommender Systems. ”, 2007

[4] Jian Wei, Jianhua He, Kai Chen, Yi Zhou, Zuoyin Tang , ”Collaborative Filtering and Deep Learning Based Recommendation System For Cold Start Items”, October 2016

[5] Lukas Lerche, ”UsingImplicit Feedback for Recommender Systems: Characteristics, Applications, and Challenges”, 2016

[6] T. Q. Lee & Y. Park , ”A Time-based Recommender System using Implicit Feedback”,

[7] Spark Overview, https://spark.apache.org/docs/latest/

[8] Mohammed Fadhel Aljunid , D. H. Manjaiah, ”Movie Recommender System Based on Collaborative Filtering Using Apache Spark ”, 2019

[9] Yehuda Koren, Robert Bell and Chris Volinsky, ” Matrix Factorization Techniques For Recommender Systems”, 2009

[10] Y. Zhou et al., “Large-Scale Parallel Collaborative Filtering for the Netflix Prize,” Proc. 4th Int’l Conf. Algorithmic Aspects in Information and Management, LNCS 5034, Springer, 2008, pp. 337-348.

[11] Yifan Hu, Yehuda Koren, Chris Volinsky, “Collaborative Filtering for Implicit Feedback Datasets“, 2008

[12] Dietmar Jannach, Kolja Hegelich, “A Case Study on the Effectiveness of Recommendations in the Mobile Internet“, 2016

[13] Recommendation Systems with Tensorflow, Coursera Course by Google

[14] Qi Zhao, Yi Zhang, Daniel Friedman, Fangfang Tan, ”E-commerce Recommendation with Personalized Promotion”, 2015

[15] Efraim Turban, David King, Jae Kyu Lee, Ting-Peng Liang, Deborrah C. Turban,” Electronic Commerce” ; 8th Edition, 2015

[16] Yulong Gu, Zhuoye Ding, Shuaiqiang Wang, Dawei Yin, ”Hierarchical User Profiling for E-commerce Recommender Systems”, January 2020

[17] Cai H, Chen Y, Fang H., ”Observational learning: evidence from a randomized Natural Field Experiment. ”, 2009

[18] Rivera A. What is C2C? https://www.businessnewsdaily.com/5084-what-is-c2c.html.

[19] https://spark.apache.org/docs/latest/mllib-collaborative-filtering.html

[20] Rivera A. What is C2C? https://www.businessnewsdaily.com/5084-what-is-c2c.html

[21] McKinsey&Company: How retaliers can keep up with consumers. https://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

[22] Building Recommendation Engines with PySpark, Datacamp Interactive Course

[23] Sophie Watsons, Building an Implicit Recommendation Engine with Spark https://www.youtube.com/watch?v=58OjaDH2FI0&t=488s