DEMOCRATIC AND POPULAR REPUBLIC Of AlGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF SCIENCE AND TECHNOLOGY HOUARI BOUMEDIENE

A Thesis Presented to The Faculty of the Department of statistics

---

TITLE

Partial Fulfillment of the Requirements for the Degree Master of Statistics and Applied Probability

by

Nihad Senhadji

June 8, 2022

..

..

ACKNOWLEDGEMENTS

Abstract

0.1 Notation

RSs Recommendation Systems.
CF Collaborative Filtering.
r_{ui} Rating of user u for item i.
$\bar{r_u}$ Average rating of the user u.
U Set of users in the system.
I Set of items.
U_i the subset of users that have rated an item i
I_u the subset of items that have been rated by a user u
$I_u \cap I_v$ the items that have been rated by two users u and v
U_{ij} the set of users that have rated both items i and j.
$U = u_1, u_2, u_3, ..., u_m$ the set of m users.
$I = i_1, i_2, i_3, ..., i_n$ the set of n users.
M Number of users.
N Number of items.
R the matrix of the logs of ratings r_{ui}.
i, j any item in the system.
u, v any user in the system.
Items Set of all items.
Users Set of all users .
|Items|=M Total number of all items.
|Users|=N Total number of all users.

Contents

List of Figures

List of Tables

Chapter 1

YASSIR EXPRESS

YASSIR [1] is a mobile application development company made by Algerians in order to facilitate the daily lives of citizens via an application that can be used either with your smartphone, tablet, or Pc.It is free to download via Google Play and the Apple store. The products that Yassir company own are Yassir Express [2] , Yassir Go [3] , Yassir Business [4] , Yassir market [5] .
Yassir Go Initiator and leader of VTC in Algeria, and an innovative transport service.
Yassir Businessis a transportation solution for businesses.
Yassir Gromadaire groceries application that markets wholesale food products.
Yassir Express

Figure 1.1: YASSIR EXPRESS:

Application developed by the company YASSIR, bringing together meal delivery services from restaurants partner, Yassir dark stores, and other services such as dry cleaning.Available on APP STORE about the smartphones under IOS, and Google

[1] https://yassir.com/en/home/
[2] https://express.yassir.io/
[3] https://app.yassir.io/auth
[4] https://yassir.com/en/business-en/
[5] https://market.yassir.io/

Play for Smartphones under Android.
Yassir Express offers a service by which E-consumers will be put in touch with E-Suppliers for the purchase of products/services/Meals remotely presented on the application, and benefit from delivery to the destination of their choice.In this context, it is recalled that Yassir Express acts as an intermediary, its role is limited to hosting offers of products/services/meals from E-suppliers and putting them in contact with E-consumers.

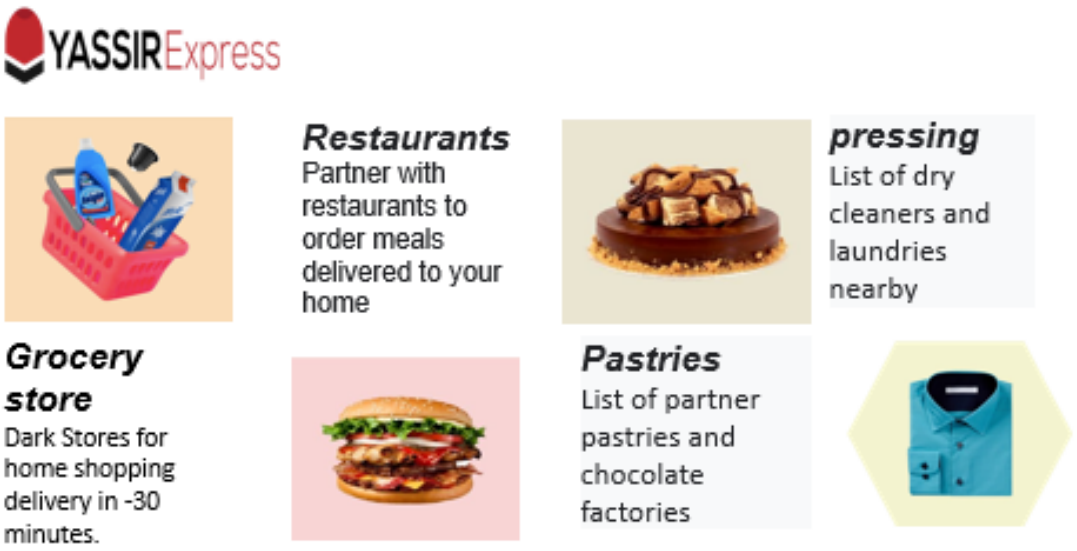

Figure 1.2: YASSIR EXPRESS Services

E-consumer means any natural or legal person who acquires, for consideration or free of charge, a good or service through the YASSIR EXPRESS Applications from an E-supplier for end-use.
E-Supplier means any natural or legal person who markets or offers the supply of goods or services or meals by means of electronic communications.
E-payment or electronic means of payment" means any payment instrument, authorized in accordance with the legislation in force, allowing its holder to make local or remote payments through an electronic system.
User Content means the textual, audio and/or visual content and information communicated during your registration, including comments and feedback relating to the Services or Third-Party Services, as well as the content and information communicated during requests for assistance and on the occasion of participation in competitions and promotional operations.
E-shop designates the dedicated space within the YASSIR EXPRESS made available to the E-supplier to offer their products/services/menus for sale.
Servicesrefers to the services available on the YASSIR EXPRESS applications and/or website.
Website means the dynamic interface accessible at the addresses:..... any other website operated by YASSIR and made available as part of the Services.
Prices means the prices applicable to the Services which will be expressly communicated to you via the Application and/or Website at the time of validation of the corresponding order.

Access to the application and the service: The service is accessible, subject to the restrictions provided for on the YASSIR EXPRESS application:

- To any natural person having full legal capacity in accordance with the law in force in the countries eligible to commit to these general conditions.
- To any legal person acting through a natural person having the legal capacity to contract in the name and on behalf of the legal person.

Acceptance of terms and conditions: The acceptance of these general conditions is materialized by a checkbox in the registration form of the YASSIR EXPRESS application and / or website. This acceptance can only be full and complete. Any conditional membership is considered null and void. E-consumers acknowledge that the use of the YASSIR EXPRESS application must only be for the service provided for this purpose mentioned in the subject. The use of the application for a reason other than that mentioned in the subject, exposes its author to penalties, without prejudice to the legal actions that YASSIR may bring for the damage caused.

Registration on the site and / or the application: Access to the service requires the E-consumer to register on the YASSIR EXPRESS application, by completing the form provided for this purpose. The E-consumer must provide all the information marked as mandatory (Name, first name, address and especially his telephone number) Any incomplete registration will not be validated.

The E-consumer acknowledges that for the creation of his profile, YASSIR will be required to communicate his telephone number to one of his suppliers for authentication.
After verification of the telephone number, the E-consumer's profile is automatically created, (The "profile"), thus giving him access to a personal space (hereafter: "the personal space") which allows him to manage his use of the service, according to the form and technical means that YASSIR deems most appropriate to provide the said Service.

Service Description: The Service consists of putting the E-consumer in direct contact with an E-supplier through the YASSIR EXPRESS application and this, in all the eligible countries mentioned in article 2 hereof. Access to the service necessarily requires that the E-consumer have a Smartphone and/or a device with internet access (3G and/or 4G connection). Any E-consumer wishing to place an order, must simply connect to the application, once done, he will just have to choose a product / meal and / or service from one of the existing E-suppliers on the platform, all of the price he has to pay is displayed on his screen, the E-consumer can then validate by pressing the "validate the order" button. The request will be sent instantly to the E-supplier.

Product compliance: YASSIR EXPRESS will combine its best efforts so that the photographic representation of the products is as faithful as possible to the real products; however, it is possible that the perception by the E-consumer of the photographic representation of the products does not correspond exactly to the product itself.

Ordered: Each item chosen through a simple click is followed by a descriptive sheet, a photo and a compatible accessory list, for which the E-supplier is solely responsible. With a simple click on the link or on the Order button, the item of your choice is slipped into your basket. The latter contains the products/meals that the E-consumer plans to buy. It is possible at any time to access the basket (by clicking on the button representing a basket) and to modify its contents by adding or deleting an article.

- Validation of the order by the E-consumer: the E-consumer can consult the basket and click on Validate your order. Any order not validated cannot be honored.
- Availability: Product availability is displayed on the site/application, on each product/meal sheet. If a product/meal is unavailable, the mention "Unavailable" will be displayed on the product/meal sheet.
  If after the order a product/meal becomes unavailable, YASSIR EXPRESS will inform the E-consumer by email of this unavailability and will give the E-consumer the possibility of choosing between:
  - make a pre-order of the product/meal, as soon as the product is available, the order is then deemed to have been validated.
  - cancel the order.
- Pre-order: is the opportunity to reserve the product/meal as soon as it is released and as soon as it is in stock/available. The order will be delivered to the E-consumer when the last pre-order product/meal is in stock or available.
- Cancel the order: The E-consumer will have the right to cancel his order provided that:
  - The cancellation is done within 10 minutes of its validation
  - The delivery person has not picked it up yet.
  - If the delivery person delivers the order to the destination, the E-consumer has the obligation to pay the service and delivery costs displayed beforehand on the YASSIR EXPRESS application even in the event of cancellation of this order

Delivery: The Products/meals are delivered to the delivery address chosen by the E-consumer as indicated when ordering. The E-consumer undertakes to receive the Products/meals thus ordered. Receipt of the order is followed by an acknowledgment of receipt by the E-consumer, a copy will be sent afterward.

- In the event of delivery to the workplace of the E-consumer, the deliverer, the E-consumer must take all necessary precautions so that the deliverer can deliver the order to him and the E-consumer can receive it, in the case otherwise, the E-consumer is obliged to pay the full amount of the order.
- YASSIR is responsible for the order of the E-consumer from the moment the delivery person picks it up from the E-supplier and delivers it to the destination indicated on the YASSIR EXPRESS application when ordering, beyond

this period of time and once the delivery has been confirmed, the E-consumer understands that YASSIR is not responsible for what may happen to the said product/meal delivered.

- The E-consumer acknowledges, for the smooth running of the service offered by YASSIR EXPRESS, that he will be required to communicate to the delivery person the following elements:

  - His identity document.
  - Order number.
  - His email address.

Delivery time: The date and type of delivery are chosen by the E-consumer according to the delivery slots available on the SITE/APPLICATION. The date is valid unless proven otherwise, by agreement between the Parties. YASSIR EXPRESS AND ITS PARTNERS will then implement their best efforts to meet the delivery times of the products ordered. In the event that these deadlines cannot be met, the E-consumer will be automatically informed by any means, in particular by e-mail, SMS or telephone.

Delivery costs: Delivery costs may vary depending on the amount of the basket, the place of delivery, the frequency of purchases by the E-consumer between two orders and the day and time chosen. Delivery costs are calculated automatically and indicated when choosing the day and delivery window. In other words, the E-consumer can find out the amount of the delivery costs by clicking on the "Basket" icon on the line "delivery costs and service costs".

Data The E-consumer acknowledges and expressly accepts: That the data collected on the YASSIR EXPRESS application and/or website and on YASSIR's computer equipment are proof of the reality of the operations carried out within the framework of these presents; - That these data constitute the only mode of proof accepted between the parties, in particular for the calculation of the sums due to YASSIR.The E-consumer can access this data in his Personal Space.

The obligation of YASSIR

- Offer the E-consumer the free download of the YASSIR EXPRESS application.
- Provide the E-consumer with a presentation page on the YASSIR EXPRESS application based on the information he has provided.
- Make available to the E-consumer in the YASSIR EXPRESS Application a management service for his account,in particular, the history of his orders made.
- Deliver the order on time. Offer the E-consumer an evaluation and rating service in order to improve the quality of the service.
- Carry out a telephone follow-up to ensure the delivery of the order.
- Process order returns for products that arrive after the deadline.

- Process returns of orders for products that do not comply with the request made by the E-consumer.

Chapter 2

An Introduction To Recommendation Systems

2.1 Introduction

The enormous amount of digital information generated by the increasing number of users on the internet has created the challenge of information overload. It becomes challenging, tedious, and time-consuming for users to retrieve the exact information on time from the web. Users need suggestions from friends or experts who have knowledge about the item.

While this freedom of purchase has made online commerce a multibillion-dollar industry, A standard catalog of an online video on demand service such as Netflix [1] can have more than 100.000 titles. The e-commerce site Amazon [2] now contains several million product references. URLs of websites today are counted in billions or tens of billions, We are entering an era of huge databases (big data) [3] where the users cannot consider themselves to have an overview of what is available and what might be interesting to them it also made it more difficult to select the products that best fit their needs. This has increased the demand for recommendation systems.which provide automated and personalized suggestions of products to consumers.
Today, users are not even willing to bother to search for items. Instead, they prefer a more convenient way to find items matching their interests and preferences. recommendation systems are heavily applied as personalized filters. Consequently, they are an integral part of users' daily lives. Whether users are watching movies, looking for new job positions,or ordering food or browsing in an online catalog, a recommendation system is running in the background to generate personalized recommendations based on the users'behavior and feedback. These include, for instance, rating , clicking ,or reviewing items. A recommendation system is also important from a business perspective since it has proven to increase a company's revenue. This statement is based on the observation that appropriate recommendations can strengthen user engagement and user satisfaction.As a result, users tend to spend more time consuming a service or purchasing more items. Recommender systems have become an attractive research field to study both from industry and academia alike. In this regard, most research

[1] https://www.netflix.com/dz-en/
[2] https://www.amazon.com/
[3] Bigdata is a term used to describe a collection of data that is huge in size and yet growing exponentially with time.

still focuses on improving recommendation quality as far as possible by developing new approaches.

The history of recommendation systems dates back to the year 1979 with relation to cognitive science [**Rich**, **1979**], in which he uses stereotypes to perform a user modeling task .
Recommendation systems gained prominence among other application areas such as approximation theory[**Powell**, **1981**], information retrieval [**Salton**, **Gerard**, **1989**], forecasting theories [**Armstrong**, **2001**].
In the mid-1990s, recommendation systems became active in the research domain when the focus was shifted to recommendation problems by researchers that explicitly rely on user rating structure and also emerged as an independent research area.
The recommendation problem can be defined as estimating the response of a user for new items, based on historical information stored in the system, and suggesting to this user novel and original items for which the predicted response is high.

2.2 Recommendation Systems

is a specific type of information filtering technique that attempts to present information items (such as movies, music, news, books, restaurants...etc.) that are likely of interest to the user. which helps users navigate through large product assortments. They have great importance for the success of the e-commerce and IT industry nowadays and gradually have gained popularity in various applications RSs generate recommendations using various types of knowledge and data about users, the available items, and previous transactions stored in customized databases. The user can then browse the recommendations and may accept them or not and may providen them by implicit or explicit feedback.
All these user actions and feedback can be stored in the RSs[4] database and may be used for generating new recommendations. In recent years, the interest in recommendation systems has dramatically increased, as the following facts indicate: Recommendation systems play an important role in such highly rated Internet sites as Amazon.com, Netflix,YouTube, Yahoo[5], Tripadvisor[6], Last.fm[7], and IMDb[8].
Moreover, many media companies are now developing and deploying RSs as part of the services they provide to their subscribers. For example, Netflix is a streaming service that offers a wide variety of TV shows, movies, have made The Netflix Prize In 2006[9] Challenge A significant boost to research into recommendation systems was given when Netflix offered a prize of $1.000.000 to the first person or team to beat their own recommendation algorithm, called CineMatch, by 10%. Initially there was a large explosion in the number of teams that submitted results, over 20.000 teams from over 150 countries registered in the first eight months After over three years of work, the prize was awarded in September 2009.The Netflix challenge consisted of a published dataset, giving the ratings by approximately half a million users on (typically small subsets of) approximately 17,000 movies. This data was selected from a larger dataset, and proposed algorithms were tested on their ability to predict the ratings in a secret remainder of the larger dataset. The information for each (user, movie) pair in the published dataset included a rating (1–5 stars) and the date on which the rating was made. The RMSE was used to measure the performance of algorithms. cinematic has an RMSE of approximately 0.95.

2.3 Data and Knowledge Sources

Recommendation systems are knowledge extraction systems that actively collect different types of information to make suggestions.
Available data sources are very large and diverse, their use for making recommendations largely depends upon the recommendation techniques to be used.
as a general classification, data used refers to three kinds of objects:
items, users, and transactions, that is relations between the users and the items.

[4] RSs Recomendation Systems
[5] https://mail.yahoo.com/
[6] https://www.tripadvisor.com/
[7] https://www.last.fm/
[8] https://www.imdb.com/
[9] https://www.wikiwand.com/en/Netflix_Prize

1. ***Items :*** **T**= t_1,t_2,t_3,....,t_m Products and Services that a recommendation system recommends are referred to as items. There are items having low difficulty and value are news, web articles, e-books, DVDs and movies,....etc.
   Items having high complexity and value are laptops, mobile phones, digital cameras, electrical appliances, PCs,Insurance policies, travel plans, financial investments, and jobs are considered as most complex items.

2. ***Users:*** **U**= u_1,u_2,u_3,....,u_n Users of a recommendation system may have very diverse aims and features. In order to consider positive recommendations if the suggested product is beneficial for the consumer and if the product is not meeting user requirements, then the recommendation is negative, therefore the system should exploit a variety of data about the users. This data can be organized in several means it depending upon the recommendation approach.

3. ***Transactions:*** Any recorded communication between a recommendation system and the user is referred to as a transaction. These are logs that collect essential information generated at the time of interaction and these are valuable for the recommendation technique. The log may also have explicit feedback given by the user like ranking to the particular product. Ratings are the most common method of transaction data that a recommendation system gathers. There are varieties of forms that ratings could adopt (Schafer et al., 2007a) as follows:

   - ***Binary rating*** scale allowing users to assign items to two different classes (like/dislike) and a good example is YouTube that allows users to rate movies with either thump up or down.
   - ***Ordinal ratings***, such as "strongly agree, agree, neutral, disagree, strongly disagree" where users are asked to select the term that best indicates their opinion regarding an item (usually via questionnaire)
   - ***Unary rating***, by contrast, allows users to assign items only to a single class, which is positive in most cases and a prominent example is the Facebook's "Like"-button. Purchased products in a web shop or clicked links on a new page can be implicit unary ratings and also in addition, unary rating can signal the purchase or observation of an item by users, or rating them positively. With the above cases, the absence of a rating indicates that there is no information relating the user to the item (perhaps a purchase was made somewhere else)

2.4 Functions of Recommendation Systems

The recommendation system offers suggestions to the user about a particular item that the user wants to use. Recommendation system plays different roles according to the user for example it's used by travel intermediary is usually used to increase the revenue like Expedia.com.
The following are different reasons to exploit RS technology by service providers:

- **Increase the revenue:**

  The major objective of a commercial RS is to increase the numbers of items that are sold.This function is most likely to be the most important in an industrial

commercial RS. The goal here is to actually sell more items than there would have been without any recommendations.

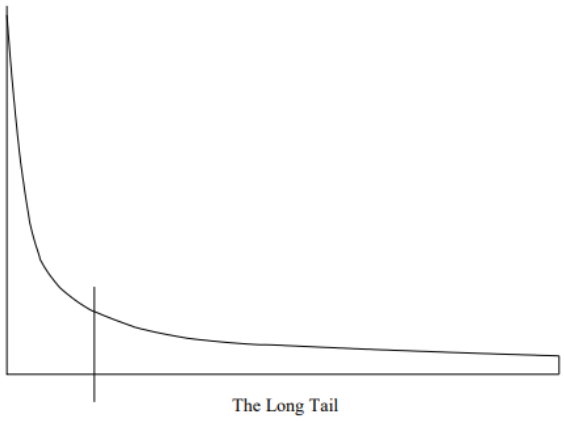

Figure 2.1: The Long Tail: physical institutions ca only provide what is popular, while online institutions can make everything availible

Recommendations are provided considering that suggested products and services meet the customer's requirements. Non-commercial recommendations are used for similar objectives. Consider an example of a content writer who wants to increase the number of newsreaders on his site.

- **Increase diversity of items sold** The aim of this function is to incite users to select items that would remain unknown without recommendation. For example, in the case of a book RS (e.g Amazon bookstore), the service provider wants to be able to sell books from all its catalogue and not only the top 5 most popular ones.

- **User satisfaction:** The recommendation system helps in improving the experience of the person with the application or website. It provides interesting, significant, and, relevant recommendations as well as provides better human-computer interaction.

- **User loyalty:** A customer always prefers to use a website or application which identifies its old users and treats him as a respected/valuable customer. It is a common feature of a recommendation system as it computes recommendations.

- **Better understanding of user needs:** The recommendation system is acting as an active learner to user's preferences by collecting explicitly or predictions made by the system. The business holders may then re-use this information for improving the stock management or production of items.

2.5 Recommendation Systems Applications in The industry:

The recommendation system has been expanded and used in various service fields, Diverse applications in areas such as e-commerce, search, Internet music and video, gaming, and even online dating apply similar techniques that leverage large volumes of data to better fulfill a user's needs.Nowadays Recommendation systems play a vital role in companies like Yahoo, Amazon Netflix, YouTube, TripAdvisor, Google, IMDB.
The recommendation system was used were classified into six main categories:
Streaming Service, Social Network Service, Tourism Service, E-Commerce Service, Healthcare Service, Education Service, Academic Information Service.
The seven main categories are divided based on the list of services that use a recommendation system with increasing users or increasing business value,

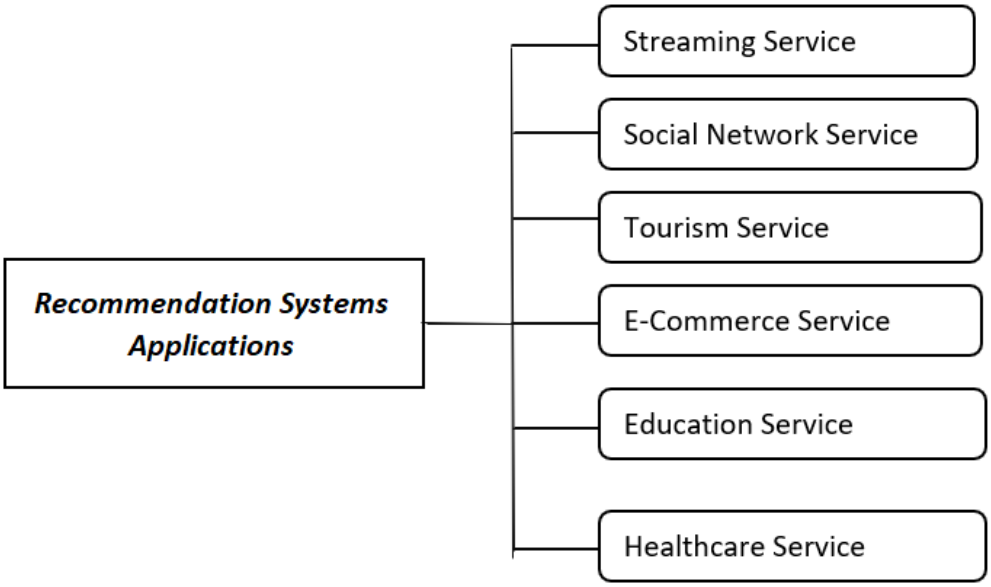

Figure 2.2: Recommendation Systems Applications

- **Streaming Service** In the past, video content, such as movies, were mainly consumed by users through TV or movie theaters. However, recently, a significant amount of video content is consumed through streaming platforms such as Netflix and YouTube. Audio content is also changing from downloading and consuming files to a user's local device to consuming content through streaming platforms such as Spotify [10]. Streaming services related to media content have been developed along with the recommendation system because it is necessary to reduce users' worries about choosing a vast amount of content and to provide content that is tailored to each user.

[10] https://www.spotify.com/dz-fr/

- **Social Network Service** Online social network services (SNS) such as Facebook[11], Instagram[12], Twitter[13], and LinkedIn [14] are huge digital-based social exchanges where users can not only lifelog their daily life, hobbies, interests etc ..., but also provide a field of interaction with other users.The vast increase in the use of SNS was also accompanied by a vast increase in user-related data. It is possible to collect content information that users register with posts through SNS.In addition, user evaluation data can be collected; as well as rating data, these include various types of feedback data, such as likes and comments. The collected data are not only used for recommendations within SNS, but are also open to utilization in recommendation systems for other businesses.

- **Tourism Service** As the demand for travel has increased, recommendation systems have begun to be used in the tourism service field to recommend tourist destinations, route recommendations,and transportation methods.
  As the travel-related recommendation system uses situational data, such as review data and location data, user location, time, and weather, collected through SNS, research on recommendation systems using SNS has increased in the tourism service field.

- **E-Commerce Service** In recent years, with the development of digital platforms such as the web and applications, the form of consuming items has changed through e-commerce platforms such as Amazon, eBay[15], and Alibaba[16]. E-commerce offers consumers many items and various options in the online environment and provides sellers with an easy way to sell. In particular, consumers were unable to go outside due to the lockdown measures due to COVID-19[17], and as a result, they were unable to use physical Stores. Consequently,consumption using digital platforms increased exponentially.

- **Education Service** From the traditional form of education in the classroom or lecture hall, a new education trend, called Smart Learning, has formed through e-learning, in which learning is conducted through an online environment .
  Smart education has started to be gradually used in education due to the increase in the spread of various smart devices.Smart education can access vast digital resources and seamlessly provide personalized learning tailored to the needs, goals, talents, and interests of learners without time and space constraints. In this domain, recommendation systems help users to find, for instance, video courses and digital content by their current skills,and experiences.Therefore, this domain is predestined for recommendation systems due to the number of available subjects and different skill levels of the users. Providing users with the right and exciting content strengthens his or her confidence in these services. Famous services in

[11] https://www.facebook.com/
[12] https://www.instagram.com/
[13] https://twitter.com/home
[14] https://www.linkedin.com/
[15] https://www.ebay.com/
[16] https://www.alibaba.com/
[17] https://www.who.int/emergencies/diseases/novel-coronavirus-2019

this regard are Coursera [18], Udemy [19]or Skillshare [20].

- **Healthcare Service** the field of health recommendation systems that help users with professional treatment,the main goal is to provide suitable treatment methods according to the symptoms of various types of diseases and the stages of each disease. To this end, the health recommendation system analyzes the patient's information and the characteristics of the disease, offers an accurate diagnosis of the disease to the patient, and recommends an appropriate treatment according to the diagnosed disease.

2.6 Each domain has its own unique recommendation challenges

Today most e-commerce sites and applications are likely to have some sort of recommendation engine powering their user experience.
The first large company to be credited as having included a recommendation system at the core of their experience is Amazon.
Amazon initially employed a simple item collaborative filtering approach, other retail companies such as eBay have followed the lead and incorporated recommendations in their experience.
News is also an area in that companies have applied recommendation approaches to personalize and focus on the interests of a user. For example, Google News[21] was powered by some kind of recommendations for news articles from the beginning. Yahoo! Has also invested in personalizing news and other web content. For news recommendation, some of the key challenges are freshness, where relevant articles may have a very limited period, and diversity, where there can be a large number of articles about the same topic.
Video recommendation has always been an active area of research, so it is not surprising that it is used in industry to recommend a variety of types of video spanning movies, TV shows. For instance, recommendations have been an important component of the YouTube experience to help navigate the vast amounts of user-generated video.
Music recommendation is also an active application area where there have been interesting developments in the past years. Pandora, for instance, created a complete business model around the idea of creating personalized music stations. They created an approach that combined traditional collaborative filtering techniques with a curated approach called the Music Genome Project. Similarly, Apple's iTunes[22] application uses information about a user's music library to drive personalized mixes and playlists. More recently, Spotify started getting in the business of providing personalized music recommendations in its service.social network companies have introduced a number of different recommendation avenues. Twitter, for example, introduced their Who to Follow recommendation algorithm to recommend new social connections. LinkedIn

[18] https://www.coursera.org/
[19] https://www.udemy.com/
[20] https://www.skillshare.com/
[21] https://news.google.com/topstories?hl=en-US&gl=US&ceid=US:en
[22] https://www.apple.com/itunes/

used a Survival Analysis approach to understand how likely a user is to change jobs. Google has also published work in recommendation systems for some of their social networks such as Orkut. Yahoo! has also worked on personalizing aspects such as comments on social sites.

2.7 Industrial Recommendation System Implementations

2.7.1 Netflix Recommendation System

Netflix was founded in 1997 and initially provided a service to rent DVDs via email. In the following years, Netflix moved into a video streaming provider with approximately 182 million users across the world[**netflix**]. Nowadays, Netflix additionally evolved into a movie and series production company. Due to its business model of providing its customers with flexible monthly subscriptions to access their content and the growing number of competitors in the field of video streaming, Netflix aims to create high user engagement and retention. One major cornerstone in this regard is to provide a customer with the content he or she is most interested in. In the past, customers went into their video rental store of trust and may be asked the owner for suggestions. Nowadays, the situation changed, customers are not willing to spend much time searching for content on their own but rather receive interesting content automatically. What sounds like a simple task to do, indeed, is challenging because of the following reasons. First, the amount of possible movies and series to choose from is extended continuously and overwhelming list. And second, the number of customers using the service is increasing from year to year. In this regard, Netflix's success is highly dependent on an accurate and well-engineered recommendation system.
In its first implementation, Netflix used Apache Chukwa [23] to process the massive amount of event data generated by the customers

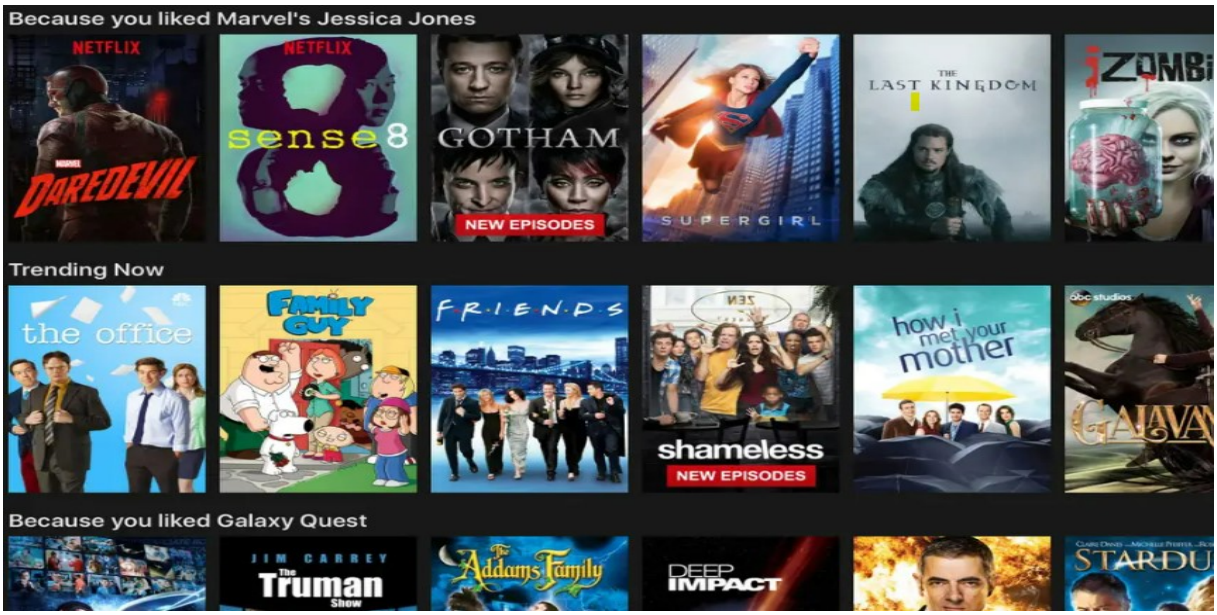

Figure 2.3: Netflix Recomendation System

[23] https://chukwa.apache.org/

2.7.2 Amazon Recommendation System

was also one of the pioneers in recommendation systems, especially in the commercial setting. During the early years, it was one of the few retailers that had the foresight to realize the usefulness of this technology. Originally founded as a book e-retailer, the business expanded to virtually all forms of products. Consequently, Amazon.com now sells virtually all categories of products such as books, CDs, software, electronics, and so on. The recommendations in Amazon are provided on the basis of explicitly provided ratings, buying behavior, and browsing behavior. The ratings in Amazon are specified on a 5 point scale, with the lowest rating being 1 star, and the highest rating being 5 star. The customer-specific buying and browsing data can be easily collected when users are logged in with an account authentication mechanism supported by Amazon.

Figure 2.4: Amazon Recommendation System

2.7.3 Google News Personalization

System the Google News personalization system is able to recommend news to users based on their history of clicks. The clicks are associated with specific users based on identification mechanisms enabled by Gmail accounts. In this case, news articles are treated as items. The act of a user clicking on a news article can be viewed as a positive rating for that article.

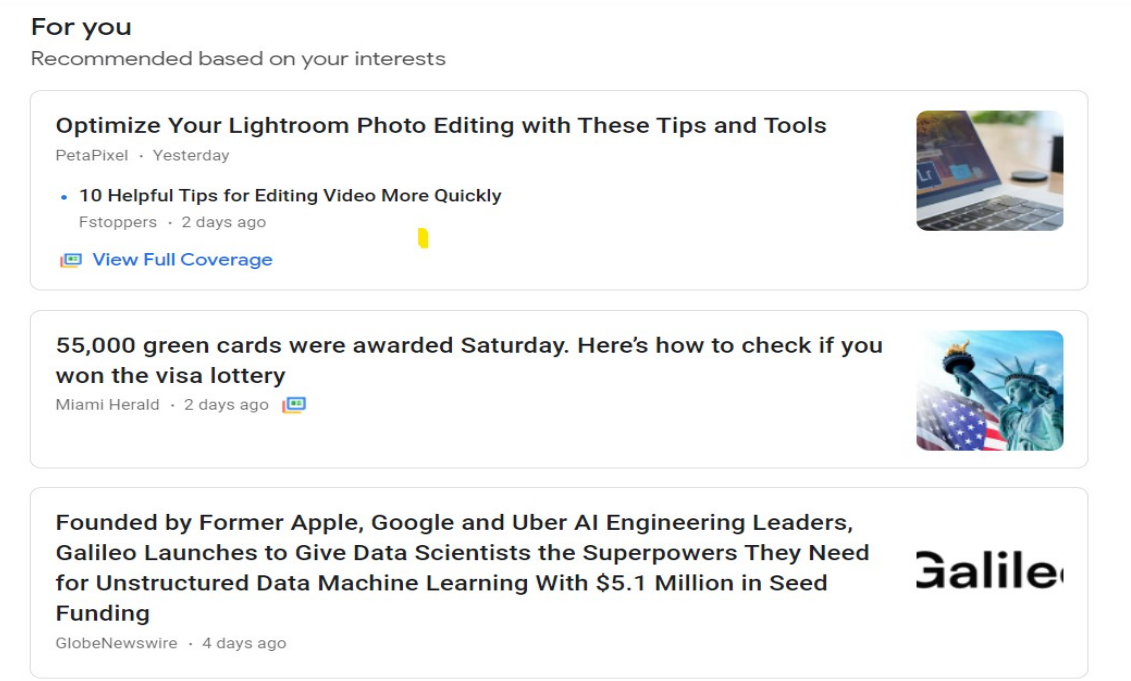

Figure 2.5: Google Recommendation System

2.7.4 Facebook Friend Recommendations

Social networking sites often recommend potential friends to users in order to increase the number of social connections at the site. Facebook is one such example of a social networking Web site. This kind of recommendation has slightly different goals than a product recommendation. While a product recommendation directly increases the profit of the merchant by facilitating product sales, an increase in the number of social connections improves the experience of a user at a social network. This, in turn, encourages the growth of the social network. Social networks are heavily dependent on the growth of the network to increase their advertising revenues. Therefore, the recommendation of potential friends (or links) enables better growth and connectivity of the network. This problem is also referred to as link prediction in the field of social network analysis. Such forms of recommendations are based on structural relationships rather than rating data.

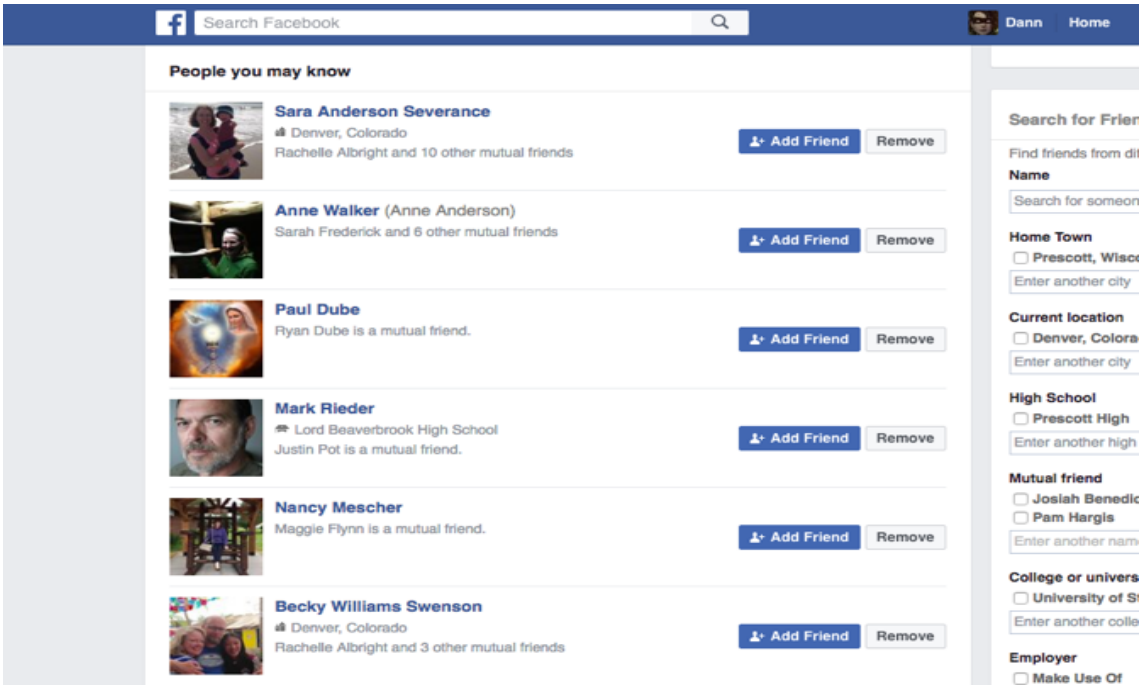

Figure 2.6: Google Recommendation System

2.8 Models of Recommendation Systems

There can be two types of recommendations **Personalized and Non-Personalized Recommendation Systems**.
Personalized Recommendation takes into consideration users previous history for rating and predicting items. On the other hand non-personalized recommendation systems recommend what is popular and relevant to all the users which can be a list of top-10 items for every new user.

2.8.1 Non-Personalized Recommendation System

In the case of Non-Personalized Recommendation System, the websites give a general item preference based on the feature evaluation of the items by different types of users. One such non-personal recommendation is the news suggestion by online news agencies.
The non-personal recommendations do not have a direct impact on the users' buying habits and personal interests, so such types of systems have less concern in recommendation system research.
There is only one type of Non- Personalized Recommendation System method. Popularity-based recommendation systems .

1. **Popularity-Based Recommendation Systems** As the name suggests Popularity based recommendation system works with the trend. It basically uses the items which are in trend right now, the bestselling product or most popular product present in the market.

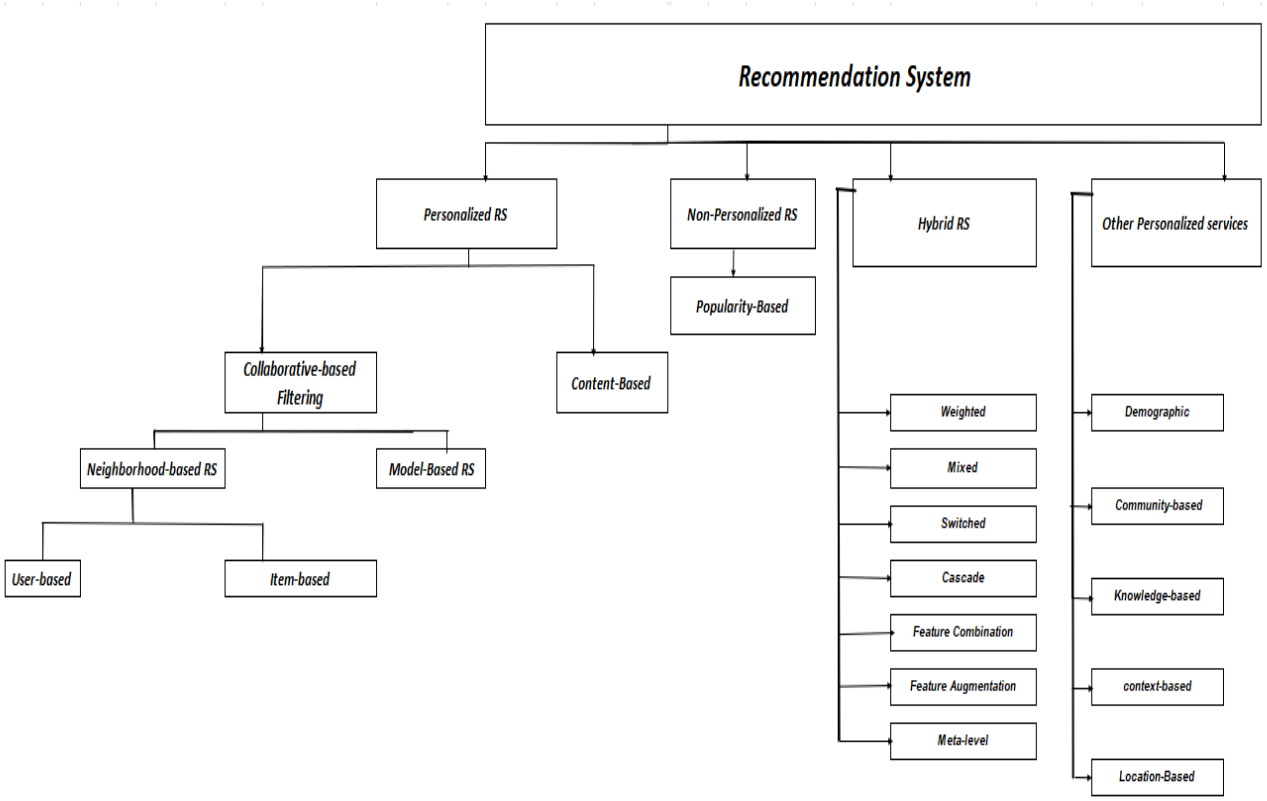

Figure 2.7: Recommendation Systems Models

2.8.2 Personalized Recommendation System

The personalized recommendation system relies on a single-user perspective, and here the suggestions are based on the individual user activity. The activity can be users rating for similar items, users' buying habits, etc. Personalized recommendation also takes similar user activity into account for better suggestions. The online shopping sides use personalized recommendation methods to suggest different items to different users depending on their business.

1. **Collaborative-Based filtering:**

   Collaborative Filtering is an information filtering model that first appeared in the 1990s , the process of filtering or evaluating items through the opinions of other users. Collaborative Filtering is a model that constructs a user's preference database using the user's evaluation data to predict items that fit the user's taste, and then uses it for recommendation. Collaborative filtering There are two types of methods that are commonly used in collaborative filtering,Neighborhood-based Collaborative Filtering (Memory-Based) and Model-Based Collaborative Filtering

   (a) **Model-Based Filtering methods**
   The recommendation system uses different machine learning and Artificial intelligence algorithms, for better prediction, taking into account the user's history, product features, and behaviors of other customers. Many of these

cannot be computed directly, and have to be estimated using mathematical models. The model-based recommendation deals with the learning algorithms that examine the data and suggest the item. Different researchers have suggested a different mechanism to filter the suggestion. These mechanisms use nearest neighbor clustering, Bayesian probabilistic model, and neural networks for proper classification and refinement of information before providing suggestions to the customers. the main Advantages of Model-Based Collaborative Filtering are

- It helps to resolve the problem of sparsity and scalability.
- accuracy is better.

in addition the disadvantages of Model-Based are

- Implementation cost is high.
- There is trade-off between scalability and efficiency of model.
- Due to dimensionality reduction methods, it may loss valuable information

(b) **Neighborhood-based methods** Neighborhood-based approach are also referred to as Memory-based collaborative filtering algorithms. These were among the earliest collaborative filtering algorithms, in which the ratings of user-item combinations are predicted on the basis of their neighborhoods. the Neighborhood-based approach is a Non-Parametric approach. The two methods that can be used to do this are called User-Based and item-based recommendation.

Advantages of Neighborhood-based methods the main advantages of Neighborhood-based methods are:

- **Simplicity:**
  Neighborhood-based methods are intuitive and relatively simple to implement. In their simplest form, only one parameter (the number of neighbors used in the prediction) requires tuning.
- **Justifiability:**
  Such methods also provide a concise and intuitive justification for the computed predictions. For example, in the item-based recommendation, the list of neighbor items, as well as the ratings given by the user to these items, can be presented to the user as a justification for the recommendation. This can help the user better understand the recommendation and its relevance, and could serve as a basis for an interactive system where users can select the neighbors for which greater importance should be given in the recommendation
- **Efficiency:**
  One of the strong points of neighborhood-based systems is their efficiency. Unlike most model-based systems, they require no costly training phases, which need to be carried out at frequent intervals in large commercial applications. These systems may require pre-computing nearest neighbors in an offline step, which is typically much cheaper than model training, providing near-instantaneous recommendations. Moreover, storing these nearest neighbors requires very little memory,

making such approaches scalable to applications having millions of users and items.

- **Stability:**
  Another useful property of recommendation systems based on this approach is that they are little affected by the constant addition of users, items, and ratings, which are typically observed in large commercial applications. For instance, once item similarities have been computed, an item-based system can readily make recommendations to new users, without having to re-train the system. Moreover, once a few ratings have been entered for a new item, only the similarities between this item and the ones already in the system need to be computed.

Disadvantages of Neighborhood-based

- These systems are fully dependent on the rankings provided by the users.
- These systems are not able to handle the sparse data which result in performance degradation.
- Recommendation cannot be made for new customers and new products.
- System is not scalable.

2. **Content-Based:**
The system learns to recommend items that are similar to the ones that the user liked in the past. The similarity of items is calculated based on the features associated with the compared items. the similarity in the rating history of the users In content-based recommendation systems, the descriptive attributes of items are used to make recommendations. The term "content" refers to these descriptions.
In Content-Based methods, the ratings and user behavior of users are combined with the content information available in the items. For example, consider a situation where John has rated the movie Terminator highly, but we do not have access to the ratings of other users. However, the item description of Terminator contains similar genre keywords as other science fiction movies, such as Alien and Predator. In such cases, these movies can be recommended to John. In Content-Based methods, the item descriptions, which are labeled with ratings, are used as training data to create a user-specific classification or regression modeling problem. For each user, the training documents correspond to the descriptions of the items she has bought or rated. The class (or dependent) variable corresponds to the specified ratings or buying behavior. These training documents are used to create a classification or regression model, which is specific to the user at hand (or active user). This user-specific model is used to predict whether the corresponding individual will like an item for which her rating or user behavior is unknown.
Advantages of Content-Based Recommendation technique such as :

- **User Independence**
  Content-Based Recommendation system builds a user profile only based upon the rating or purchase done by the user in the past. No neighbor is considered for building the profile of the user who has the same interest as of user.
- **Transparency**
  Explanation facility of a content-based recommendation system is transparent to the user which means it provides explanations of the recommendations.
- **New Item**
  It does not suffer from the first-rate problem which means if an item is not rated by any user, it is still able to recommend that item to the user.

And also disadvantages of content-Based Recommendation are:

- **Limited Content Analysis**
  One of the shortcomings of a content-based recommendation system is limited content associated with the item in terms of a number of features and types. Proper differentiation cannot be done between the item's user likes and items user dislikes if available data is insufficient. Representation sometimes is able to capture only certain aspects of user choice but not all. For example, Web pages, feature extraction techniques from the text completely overlook visual qualities and additional multimedia information.
- **Over-Specialization**
  Content-based recommendation system does not have any essential method

to explore something unpredicted. The system can recommend only those items which result in a high score while matching with the user profile

- **New User**
  To make a recommendation system to learn about user preferences, sufficient ratings need to be collected. System is not able to provide reliable recommendations to the new users as no past data is available

3. **Hybrid Recommendation Systems:**
   Hybrid Recommendation System This method has the potential to tackle both content and collaborative data in its term and combine the efforts of both to produce a system from their strategic advantage.

   Both filtering models feature limitations because the Content-Based Filtering model relies on metadata about the user's item, and Collaborative Filtering relies on the user's item rating data.data. A Hybrid recommendation model was proposed to solve the limitations of both recommendation filtering models and to improve the recommendation performance. The Hybrid recommendation model is divided iton seven types: Weighted Hybridization, Switching Hybridization, Cascade Hybridization, Mixed Hybridization ,Feature Combination, Feature-Augmentation, and Meta-level.

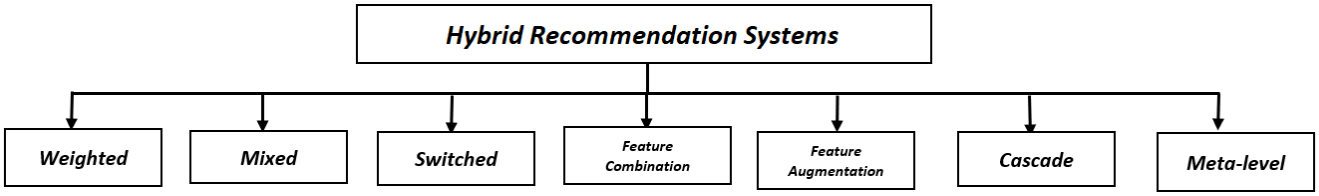

Figure 2.8: Hybrid Recommendation System

(a) **Weighted:**
The score of different recommendation components are combined numerically.

(b) **Mixed:**
Recommendations from different recommenders are presented together.

(c) **Switched:**
In this, system switches among various recommending methods depending on some conditions. For instance, a Colaborative-Based filtering method may shift to the content-based recommendation if the collaborative filtering method doesn't offer sufficient reliable results.

(d) **Feature Combination:**
Features derived from different knowledge sources are combined together and given to a single recommendation algorithm.

(e) **Feature Augmentation:** One recommendation technique is used to compute a feature or set of features, which is then part of the input to the next technique.

(f) **Cascade:**
Recommenders are given strict priority, with the lower priority ones breaking ties in the scoring of the higher ones.

(g) **Meta-level:**
One recommendation technique is applied and produces some sort of model, which is then the input used by the next technique.

The Hybrid Recommendation model is mainly designed to solve the sparsity Problem, the main goal of most studies dealing with the Hybrid recommendation model is to compensate for the lack of rating data by integrating the information of the Content-Based Filtering and Collaborative Filtering models.

4. **Demographic:** This type of system recommends items based on the demographic profile of the user, such as age, gender,Location, education, etc. for classifying groups of users,For example, if it is given in a site that 70% of people of your age use this product and getting benefits. Many commercial sites and industries have adopted this strategy as it is not too complex and the implementation is also simple. In a Demographic-Based Recommendation System, proper market research is needed along with the cultural value of citizens in the specified region, accompanied by a short survey report to accumulate data for categorization.

5. **Community-based recommendation system:** The main idea in the community-based recommendation is to recommend to the user based on the preference of his friends, following the epigram ''Tell me who your friends are, and I will tell you who you are".

6. **Contextual recommendation system:** In this kind of recommendation system, the results of recommendation vary according to the context of the user. For example, in the temporal context, the clothes recommended in summer vary totally from those recommended in winter. For a social context, an example recommending a restaurant for a Saturday night with friends,which would vary from a restaurant for a lunch during the week with co-workers

7. **Knowledge-based system:** recommend items based on specific domain knowledge about how certain item features meet users' needs and preferences a. Knowledge-Based Recommendation Systems are particularly useful in the context of items that are not purchased very often. Examples include items such as real estate, automobiles, tourism requests, financial services, or expensive luxury goods. In such cases, sufficient ratings may not be available for the recommendation process. As the items are bought rarely, and with different types of detailed options, it is difficult to obtain a sufficient number of ratings for a specific instantiation of the item at hand. A particular item may have attributes associated with it that correspond to its various properties, and a user may be interested only in items with specific properties. For example, cars may have several makes, models, colors, engine options, and interior options, and user interests may be regulated by a very specific combination of these options. Thus, in these cases, the item domain tends to be complex in terms of its varied properties, and it is hard to associate sufficient ratings with the large number of combinations at hand. Such cases can be addressed with knowledge-based recommendation systems.

8. **Location-Based Recommendation:** Systems With the increasing popularity of GPS-enabled mobile phones, consumers are often interested in location-based recommendations. For example, a traveling user may wish to determine the closest restaurant based on her previous history of ratings for other restaurants. In general, the recommendation of places always has a location aspect built into it. An example of such a system is Foursquare2. which recommends various types of places such as restaurants or nightlife venues. There are two types of spatial locality that are common to such systems.

   (a) **User-Specific Locality:** The geographical location of a user has an important role in her preferences. For example, a user from Wisconsin might not have the same movie preferences as a user from New York. This type of locality is referred to as preference locality.

   (b) **Item-specific locality:** The geographical location of an item (e.g., restaurant) might have an impact on the relevance of the item, depending on the current location of the user. Users are generally not willing to travel very far from their current location. This type of locality is referred to as a travel locality.

   Location-based recommendation systems have witnessed an increasing interest in recent years because of the increasing prevalence of mobile phones and other GPS-enabled devices.

2.9 Evaluating Recommendation Systems

In order to evaluate the performance of the Recommendation Systems models, The quality of RSs may be measured in many ways, and different metrics can be applied.

The simplest way to evaluate the performance of a recommendation system is:

2.9.1 Accuracy

is one of the most fundamental measures through which recommendation systems are evaluated. Three types of accuracy may be considered:

Prediction-based metrics , Decision-based metrics , Rank-based metrics.

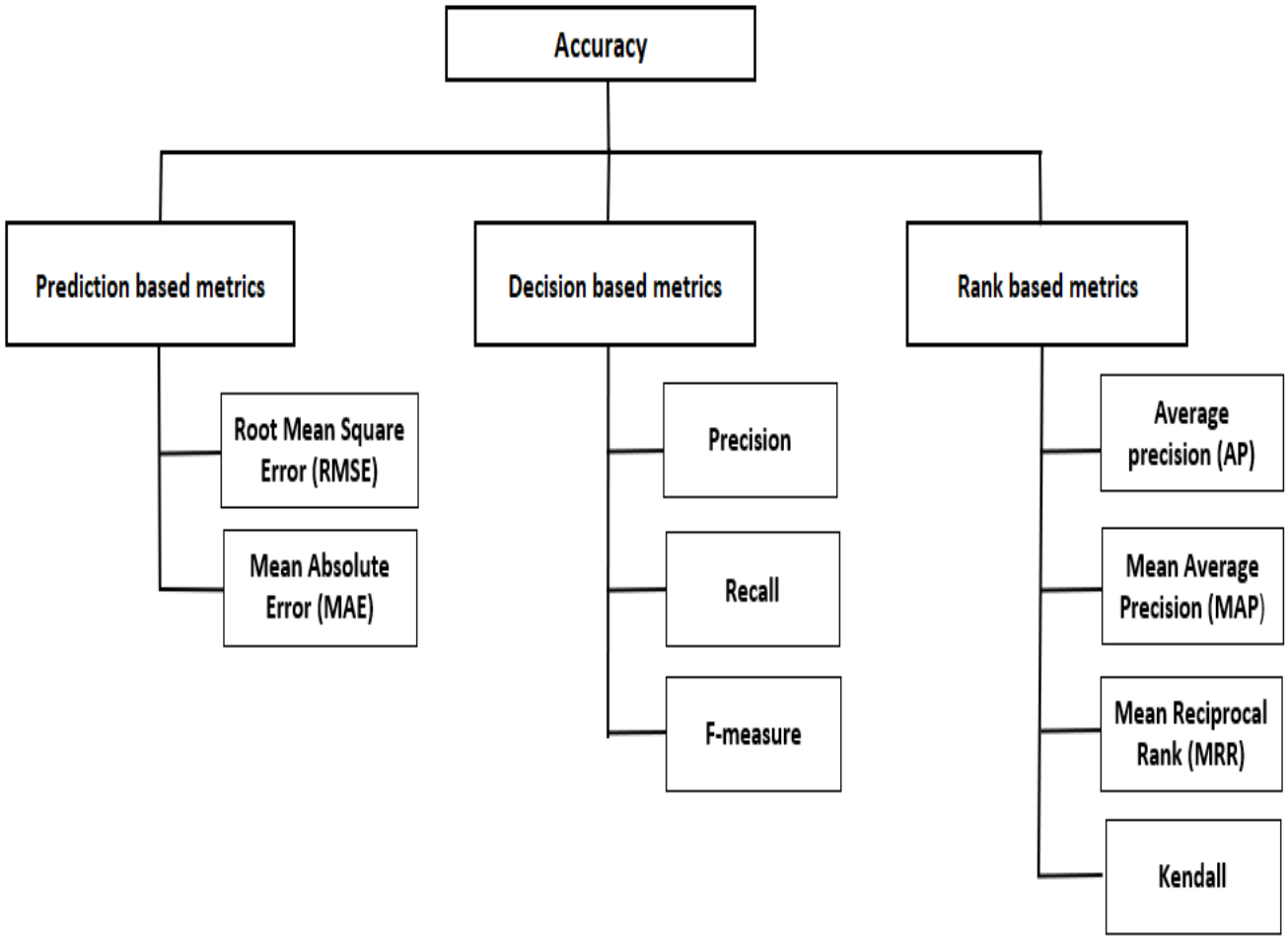

Figure 2.9: Accuracy metrics

Prediction Based Metrics

Used to measure the difference between predicted rating $\hat{r_{ui}}$ and the rating given by the user r_{ui}.

- **Root Mean Square Error (RMSE)**

  it evaluates the difference between the ratings predicted $\hat{r_{ui}}$ by the RS and the ratings given by the user r_{ui} . It also has become popular in the recommendation system field with the Netflix Challenge (Bell et al., 2007) , reducing the RMSE amounts to increase the relevance and precision of the recommendations.

  The RMSE is defined as follows :

$$RMSE = \sqrt{\frac{\sum_{(u,i,r)\epsilon R}(\hat{r_{ui}} - r_{ui})^2}{|R|}}$$

  Where R be the set of ratings ,normally a Test Set.

  (Bell et al., 2007) R. Bell, Y. Koren, et C. Volinsky., 2007. The bellkor 2008 solution to the netflix prize. The Netflix Prize.

- **Mean Absolute Error (MAE)**

  The Mean Absolute Error (MAE) Similar to $RMSE$ is a metric that is used to detect the accuracy of the system by comparing the predicted ratings against the actual ratings of the items.The average absolute difference between the estimated and the user's true rating is termed as the mean absolute error. The relationship between the metrics measure and the system's performance is inverse, which implies better performance of the system ,The lower the MAE,the more accuracy the recommendation system predicts rating.

$$MAE = \frac{\sum_{(u,i,r)\epsilon R}|\hat{r_{ui}} - r_{ui}|}{|R|}$$

  Where R denotes the amount of user u rated items.,normally a Test Set,r_{ui} determines the actual rating that user u rates item i and $\hat{r_{ui}}$ denotes the predicted rating of item i for user u .

Decision based metrics

Decision-based metrics evaluates the top-N recommendations for a user. There are four different cases to take into account:

- **True Positive (TP)** The system recommends an item the user is interested in.
- **False Positive (FP)** The system recommends an item the user is not interested in.
- **True Negative (TN)** The system does not recommend an item the user is not interested in.
- **False negative (FN)** The system does not recommend an item the user is interested in.

	Relevant	**Not Relevant**
Recommended	**True Positive**	**False Positive**
Not Recommended	**False Negative**	**True Negative**

Table 2.1: Confusion Matrix

- **Precision**

  Precision defines the segment of suggested items that are identical to the users' preferences in the testing dataset,It is also known as positive predictive value and measures the relevancy of the result. Assuming that, in the system, items ratings belong to 3–5 are positive ratings and 1 and 2 are negative ratings. Thus, when the item's real rating exists within 3–5 and the item's predicted rating is 3–5, then we can conclude that it is like the true positive (TP), whereas, when the item's real rating belongs to 1 and 2, but the item's predicted rating belongs to 3–5, then we can declare this as false positive (FP). Thus, the precision is computed as the ratio of the number of relevant recommended items to the number of all the recommended items,Higher the precision better is a recommendation .

$$Precision = \frac{TP}{TP + FP}$$

  Where **TP+FP** refers to the total suggested items .

- **Recall**

  The recall is the mean quantity of items of the testing dataset that exists among the ranked list from the training dataset. It also called sensitivity and measures the amount of truly relevant results that are identified. Based on the above precision's hypothesis, an item's real rating exists within 3–5 and the item's predicted rating belongs to 3–5, then, we can call it a true positive (TP), on the other hand, when the item's predicted rating belongs to 1–2, then it is classified

as a false negative (FN) ,the equation of the recall computation as the ratio of number of relevant recommended items to the total number of relevant items.

$$Recall = \frac{TP}{TP + FN}$$

- **F-measure:**

  It has been seen that, when the precision and recall cannot anticipate decent results, then $F_\beta - measure$ is used as a weighted harmonic mean of the precision and recall to ensure better evaluation of the test system
  The general formula of F-measure is given as follows :

$$F_\beta - measure = \frac{(\beta^2 + 1)(Precision.Recall))}{\beta^2.Precision + Recall}$$

  β is a parameter that controls a balance between Precision and Recall.

  The most common $F - measure$ is $F_1 - measure$,where $\beta = 1$.
  $F_1 - measure$ the standard F-Measure , the harmonic mean of precision and recall.

$$F_1 - measure = \frac{2 \times (Precision + Recall)}{(Precision + Recall)}$$

Rank-based metrics

A rank metric measures the ability of a RSs to estimate the correct order of items concerning the user's preference, which is called the measurement of rank correlation in statistics.
We need to be careful with these metrics as it makes no sense to compute them if a user appears only once in the test dataset. Rank-based metrics are sometimes hard to interpret

Average precision (AP)

Average Precision (AP) equals:

$$AP = \frac{\sum_{k=1}^{n}(P(k) \times rel(k))}{NumberofRelevantItems}$$

Where $P(k)$ is Precision at top-k,and $rel(k)$ is an indicator function equaling 1 if the item at rank is a relevant item, and zero otherwise, n is the number of items in the recommendation list.

Mean Average Precision (MAP)

Mean Average Precision (MAP) is defined as the precision (i.e. the percentage of relevant items among the first k recommendations) at the position k in the ranked results

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}$$

where Q is the whole number of the users involved in this recommendation system.

Mean Reciprocal Rank (MRR)

The mean reciprocal rank (MRR) is a popular evaluation metric that is used produce a list of ranked items for queries by calculate the average of reciprocal of the rank in which the first correct item was retrieved at each prediction. It will match the recommended list with the predicted recommended list. It will give a higher score if the match is found at the top of the list. The MRR is computed as follows:

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

Where $rank_i$ is the rank of ith item.

Kendall– τ

Kendall Rank correlation is a non-parametric test that measures the strength of dependence between two variables. If we consider two samples, a and b, where each sample size is n, we know that the total number of pairings with a b is $n(n-1)/2$. The following formula is used to calculate the value of Kendall rank correlation:

$$\tau = \frac{n_c - n_d}{\frac{1}{2}n(n-1)}$$

Where n_c Number of concordant. n_d Number of discordant.

2.9.2 Non Accuracy Metrics

Coverage: As the prediction accuracy of a recommendation system, especially in collaborative filtering systems, in many cases grows with the amount of data, some algorithms may provide recommendations with high quality, but only for a small portion of the items where they have huge amounts of data. There are two types of coverage, which are referred to as user-space coverage and item-space coverage, respectively.

Novelty: The novelty of a recommendation system evaluates the likelihood of a recommendation system to give recommendations to the user that they are not aware of, or that they have not seen before]. Unseen recommendations often increase the ability of the user to discover important insights into their likes and dislikes that they did not know previously. This is more important than discovering items that they were already aware of but have not rated.

Confidence : in the recommendation can be defined as the system's trust in its recommendations or predictions. The estimation of ratings is an inexact process that can vary significantly with the specific training data at hand. Furthermore, the algorithmic methodology might also have a significant impact on the predicted ratings. This always leads to uncertainty in the user about the accuracy of the predictions. Many recommendation systems may report ratings together with confidence estimates. For example, a confidence interval on the range of predicted ratings may be provided. In general, recommendation systems that can accurately recommend smaller confidence intervals are more desirable because they bolster the user's trust in the system. For two algorithms that use the same method for reporting confidence, it is possible to measure how well the predicted error matches these confidence intervals. For example, if two recommendation systems provide 95% confidence intervals for each rating, one can measure the absolute width of the intervals reported by the two algorithms. The algorithm with the smaller confidence interval width will win as long as both algorithms are correct (i.e., within the specified intervals) at least 95% of the time on the hidden ratings. If one of the algorithms falls below the required 95% accuracy, then it automatically loses. Unfortunately, if one system uses 95% confidence intervals and another uses 99% confidence intervals, it is not possible to meaningfully compare them. Therefore, it is possible to use such systems only by setting the same level of confidence in both cases.

Trust: in the recommendation system is How much the user trusts the recommendation, this aspect varies between users. Some users trust recommenders that offer them new items, while others prefer having some of their preferred items in the recommendation list in order to trust the system.

Diversity : The notion of diversity implies that the set of proposed recommendations within a single recommended list should be as diverse as possible, usually users prefer systems with high diversity as they can explore new horizons of items. Consider the case where three restaurants are recommended to a user in the list of top-3 items. If all three restaurants are of a particular cuisine (eg. French, Thai, Italian, Indian, and Chinese) and contain similar dishes, then there is little diversity in the recommendations. If the user dislikes the top choice, then there is a good chance that she might dislike all of them. Presenting different types of restaurants can often increase the chance that the user might select one of them.

Serendipity : is a measure of the level of surprise in successful recommendations. In other words, recommendations need to be unexpected, as serendipitous recommen-

dation helps the user find a surprisingly interesting item he might not have otherwise discovered. for example, Consider the case where a particular user frequently orders Indian restaurants. The recommendation of a new Pakistani restaurant to that user might be novel if that user has not eaten at that restaurant earlier. However, such a recommendation is not serendipitous, because it is well known that Indian and Pakistani food is almost identical. On the other hand, if the recommendation system suggests a new Ethiopian restaurant to the user, then such a recommendation is serendipitous because it is less obvious.

Robustness and Stability : The robustness is the ability to make good recommendations in the presence of fake information ,As more people rely on recommendation systems to guide them through the item space, influencing the system to change the rating of an item may be profitable to an interested party, For example, an owner of a hotel may wish to boost the rating for their hotel. This can be done by injecting fake user profiles that rate the hotel positively, or by injecting fake users that rate the competitors negatively. Such attempts to influence the recommendation are typically called attacks. An attack occurs when an agent tries to influence the behavior of the recommendation system. To measure the actual influence of an attack on the outcome of a recommendation system, the measures "robustness" and "stability" have been proposed by O'Mahony et al. (2004). Robustness measures the shift in the overall accuracy before and after an attack; the stability measure expresses the attack-induced change of the ratings predicted for the attacked items.

Scalability : In recent years, the sizes of the data sets continue to increase over time. As a result, it has become increasingly essential to design recommendation systems that can perform effectively and efficiently in the presence of large amounts of data. Scalability is typically measured by experimenting with growing data sets, showing how the speed and resource consumption behaves as the task scales up, A variety of measures are used for determining the scalability. Such as Training time: Most recommendation systems require a training phase, which is separate from the testing phase, Therefore, as long as the training time is of the order of a few hours, it is quite acceptable in most real settings. And Prediction time Once a model has been trained, it is used to determine the top recommendations for a particular user. Furthermore,the computation increases as the number of users increases. User-based methods work fine for thousands of users, but scalability gets to be a problem when we have a million users.
This is where the importance of scalability has become particularly great in recent years because of the increasing importance of the "big data" paradigm.

Utility : Many e-commerce websites employ a recommendation system in order to improve their revenue by, e.g., enhancing cross-sell. In such cases the recommendation systems can be judged by the revenue that it generates for the website, Utility can be measured cleanly from the perspective of the recommendation system owner.

Privacy: Providing personal data to the recommendation systems may increase the system performance but may lead to problems of data privacy and security. Users are reluctant to feed data into recommendation systems that suffer from the data privacy issue

2.10 Challenges in Recommendation Systems

This section explores the main challenges that face the Recommendation Systems.

2.10.1 Cold-Start

Cold starts can be classified into two distinct subsets: item cold starts and user cold starts. Whenever a new item is displayed on an the web application [24] , it goes through the item cold start, and this means that there are no rating due to the lack of user interaction. If there are not enough user interactions, RSs can't retrieve the user preferences, which leads to the RSs being unable to display any recommendation. The cold-start behavior occurs when a user creates an account for the first time and does not have any item preferences or history available to base recommendations. The solution to the problem can be provided by incorporating the global relevance where the system can retrieve the item based on its popularity.

2.10.2 Sparsity

The sparsity problem in recommendation systems remains that only a few values in the user-item matrix are known, while most items are missing. In many large-scale applications, both the number of items and the number of users are large, which leads to the sparsity problem, where the user-item matrix becomes a sparse matrix[25]. in which most of the rating are zero . Generally, where the majority of the users do not rate most of the items, consequently, the rating matrix becomes very sparse. Due to this, the data sparsity problem arises that declines the chances of finding a set of users with similar ratings.

2.10.3 Scalability

To provide a recommendation is challenging when the recommendation systems deals with huge datasets. Since some recommendation algorithms work extremely well with small datasets, but its efficiency may not work for large datasets.

[24] **A web application** (or web app) is application software that runs on a web browser.
[25] **Sparse matrix** is a matrix in which most of the elements are zero

Chapter 3

Neighborhood Based Recommendation System

The collaborative filtering approach recommends items based on the opinions of other users. Typically, by computing the similarity of users, a set of "nearest neighbor" users whose known preferences correlate significantly with a given user . Preferences for unseen items are predicted for the user based on a combination of the preferences known from the nearest neighbors. Thus, in this approach, users share their preferences regarding each item that they consume so that other users of the system can better decide which items to consume (Herlocker et al. 1999). The collaborative filtering approach is the most successful and widely adopted recommendation technique to date.

As mentioned, the collaborative filtering approach utilizes user preferences to generate recommendations. Several different techniques have been proposed for collaborative filtering recommendations, including neighborhood-based, model-based ,latent factors , Due to space limitation, we will only review the neighborhood-based collaborative filtering techniques since they are the most prevalent algorithms used in collaborative filtering for recommendation.
The neighborhood-based CF algorithm use the entire user-item database to generate a prediction. Every user is part of a group of people with similar interests. The neighborhood-based CF algorithm, a prevalent memory-based CF algorithm, uses the following steps:

- **Dimension Reduction:**
  Transform the original user item matrix into a lower dimensional space to address the sparsity and scalability problems.

- **Neighborhood Similarity:**

  the similarities between all other users and the active user to form a proximity-based neighborhood with a number of like-minded users for the active user.

- **Produce a prediction $\hat{r_{ui}}$ of user u to item i.**

- **Select the neighborhood of the active user.**

3.1 Types of Neighborhood-based collaborative filtering

There are two main type of Neighborhood-based collaborative filtering

3.1.1 User-based neighborhood collaborative filtering:

User-based neighborhood collaborative filtering is the most successful method for building recommendation systems to date, and is extensively used in many commercial, method first seek who shared the same rating pattern with the target user and then use the ratings of the similar users to estimate the predictions and then recommendation.
This method calculate the rating for a yet unrated item of the active user, average the ratings of the nearest neighbors about this particular item. In order to generate more accurate predictions, rating values of neighbor are assigned with weights according their similarity to the target user.

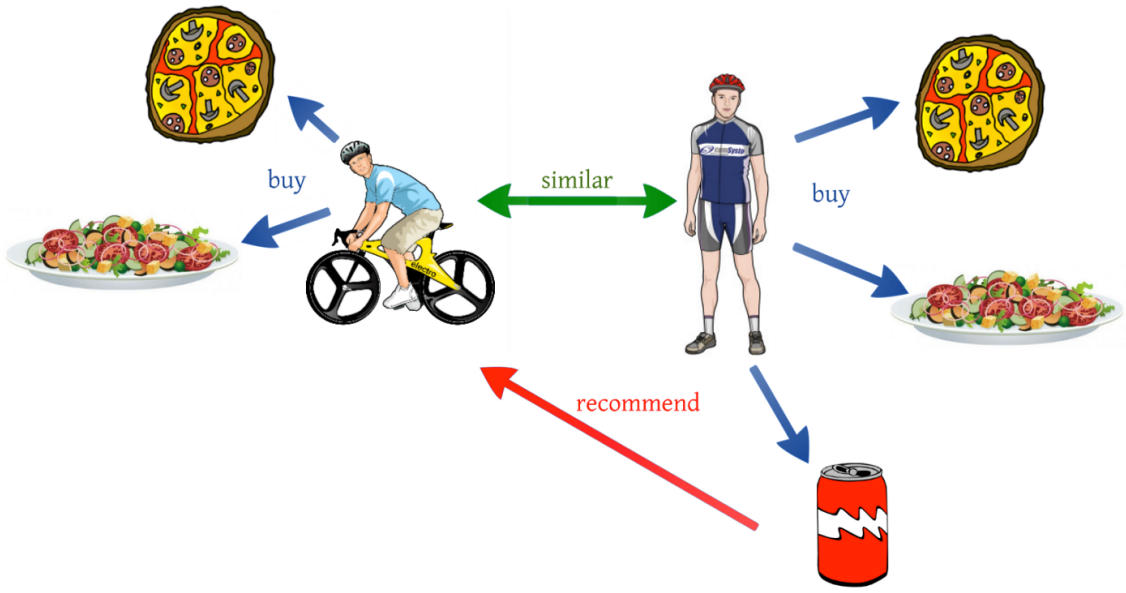

Figure 3.1: User-based neighborhood collaborative filtering

The main idea behind user-based filtering is that if we are able to find users that have bought similar items in the past, they are more likely to buy similar items in the future too. This method for generating the more precise prediction, weights allocate to the values of neighbor based their similarity to the active user.

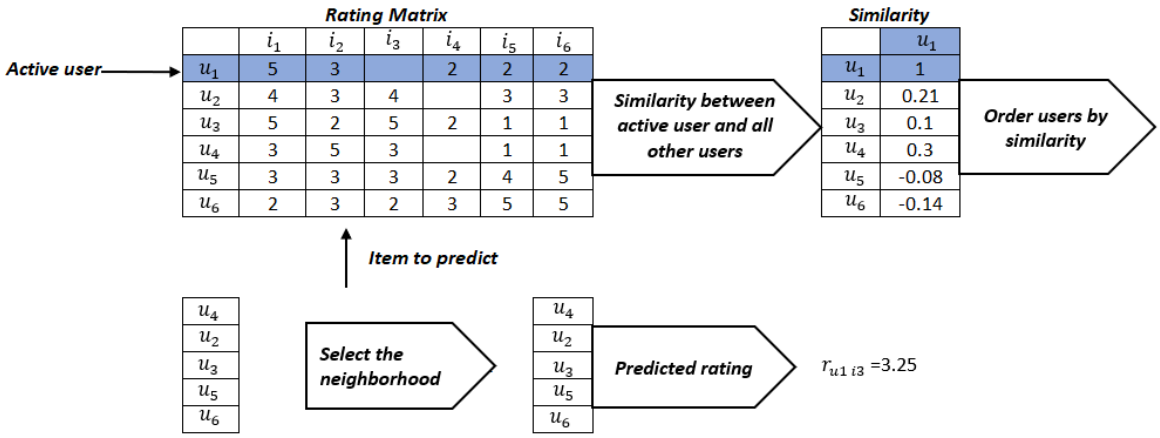

Figure 3.2: The user-Based pipeline

3.1.2 Item-based neighborhood collaborative filtering:

Item-based neighborhood collaborative filtering is the transpose of user-based nearest neighbor algorithms that produce predictions based on similarities between items. An item-based method exploits the similarity among the items.

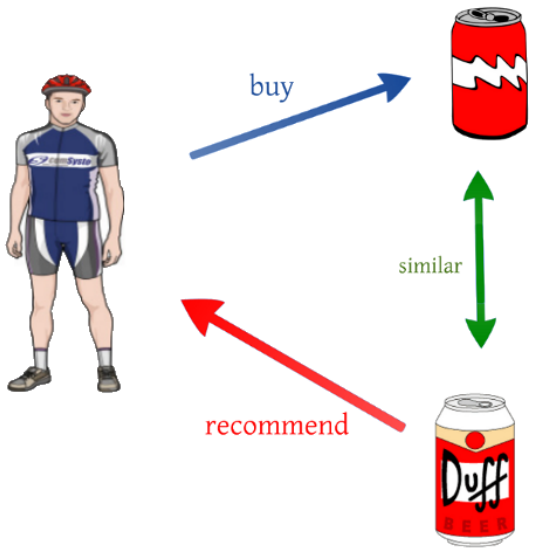

Figure 3.3: The item-Based pipeline

This method looks into the set of items that a user has rated to (the ones the user already buys) and computes the similarity among the target item (to decide whether is worth to recommend it to the user or not).

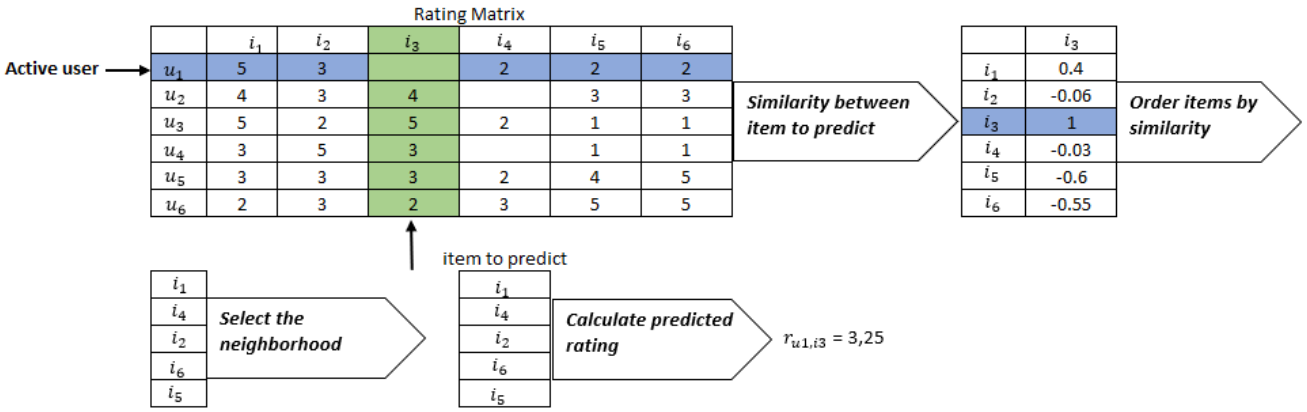

Figure 3.4: The item-Based pipeline

3.2 User-based VS Item-based Recommendation

When choosing between the implementation of a user-based and an item-based neighborhood recommender system, five criteria should be considered:

- **Accuracy:** The accuracy of neighborhood recommendation methods depends mostly on the ratio between the number of users and items in the system.As the similarity between two users in user-based methods, which determines the neighbors of a user, is normally obtained by comparing the ratings made by these users on the same items. an item-based method usually computes the similarity between two items by comparing ratings made by the same user on these items. In general, In cases where the number of users is much greater than the number of items, such as large commercial systems like Amazon.com, item-based methods can therefore produce more accurate recommendations Likewise, systems that have fewer users than items, e.g , a research paper recommender with thousands of users but hundreds of thousands of articles to recommend, may benefit more from user-based neighborhood methods.

- **Efficiency:** The memory and computational efficiency of recommender systems also depends on the ratio between the number of users and items. Thus, when the number of users exceeds the number of items, as is it most often the case, item-based recommendation approaches require much less memory and time to compute the similarity weights (training phase) than user-based ones, making them more scalable.

- **Stability:** The choice between a user-based and an item-based approach also depends on the frequency and amount of change in the users and items of the system. If the list of available items is fairly static in comparison to the users of the system, an item-based method may be preferable since the item similarity weights could then be computed at infrequent time intervals while still being able to recommend items to new users.

- **Justifiability:** An advantage of item-based methods is that they can easily be used to justify a recommendation. Hence, the list of neighbor items used in the

prediction, as well as their similarity weights, can be presented to the user as an explanation of the recommendation. By modifying the list of neighbors and/or their weights, it then becomes possible for the user to participate interactively in the recommendation process. User-based methods, however, are less amenable to this process because the active user does not know the other users serving as neighbors in the recommendation.

- **Serendipity:** In item-based methods, the rating predicted for an item is based on the ratings given to similar items. Consequently, recommender systems using this approach will tend to recommend to a user items that are related to those usually appreciated by this user. user-based approaches are more likely to make serendipitous recommendations This is particularly true if the recommendation is made with a small number of nearest-neighbors.

3.3 Rating Matrix

User/Item	i_1	i_2	...	i_j	...	i_n
u_1	r_{11}	r_{12}	...	r_{1j}	...	r_{1n}
u_2	r_{21}	r_{22}	...	r_{2j}	...	r_{2n}
u_3	r_{31}	r_{32}	...	r_{3j}	...	r_{3n}
.	.	.	...	.	...	.
.	.	.	...	.	...	.
.	.	.	...	.	...	.
u_i	r_{i1}	r_{i2}	...	r_{ij}		r_{in}
.	.	.	...	.	...	.
.	.	.	...	.	...	.
.	.	.	...	.	...	.
u_m	r_{m1}	r_{m2}	....	r_{mj}	...	r_{mn}

Table 3.1: **User-Item Rating Matrix**

Let R be a user–item matrix of size $m \times n$, where n is the total number of the items , while m is the tottal number of the users, while each row represents a user and each column represents one single item , each cell represents the rating r_{ui} given by user u to item i or null if the user did not rate the item yet, In most of the cases, this matrix is sparse because each user does not normally rate all the items in the data set.

3.4 Explicit and Implicit Ratings

For the purpose of gathering the preference of users, the system shall collect ratings from the user's interaction. The ratings can be either explicit or implicit.
The explicit ratings, the most prominent form of such feedback in the literature is the user-provided ratings, e.g., on a [1-5] scale often displayed as "stars" . On the implicit ratings, we do not ask users to give any ratings, and we just observe their behaviorthe implicit rating can be gathered using the user's feedback and interaction,An example of this is keeping track of what a user clicks on in the online New York Times.
Another implicit rating is what the users buy. Nonetheless, explicit ratings are not always readily available as many users may be reluctant to rate items. Implicit ratings are often readily available since it is mainly concerned with modeling implicit behavior such as user clicks. Many recommendation systems are centered on implicit ratings, which indirectly reflect users' opinions through observing user behavior.
There are diverse forms of implicit ratings on the Web pages: purchase history, browsing history, watches, and even mouse movements, the clicks on the link to a page, time spent looking at a page, repeated visits referring a page to others. also of example on Music players: what the person plays skipping tunes number of times a tune is played, and others like,save, share. IMDB also collects ratings ranging from one to ten stars for movies, and YouTube provides the thumbs-up and thumbs-down buttons for users to show their preferences.
It is apparent that gathering explicit ratings requires users to indicate their interests proactively. For example, a user that purchased many books by the same author probably likes that author. Note that implicit ratings are inherently noisy. We can only guess their preferences and true motives. A user who watches a movie does not necessarily indicate a positive view of that movie. Explicit ratings struggle mainly from the users who are lazy and do not rate items, and also, they may lie or give only partial information.

3.5 Similarity measures in collaborative filtering:

Similarity computation between items or users is a critical step in Neighbhood-based collaborative filtering algorithms.,

For item-based CF algorithms, the basic idea of the similarity computation between item i and item j is first to work on the users who have rated both of these items and then to apply a similarity computation to determine the similarity s_{ij}.
For a user-based CF algorithm, we first calculate the similarity,s_{uv} between the users u and v who have both rated the same items. There are many different methods to compute similarity or weight between users or items.

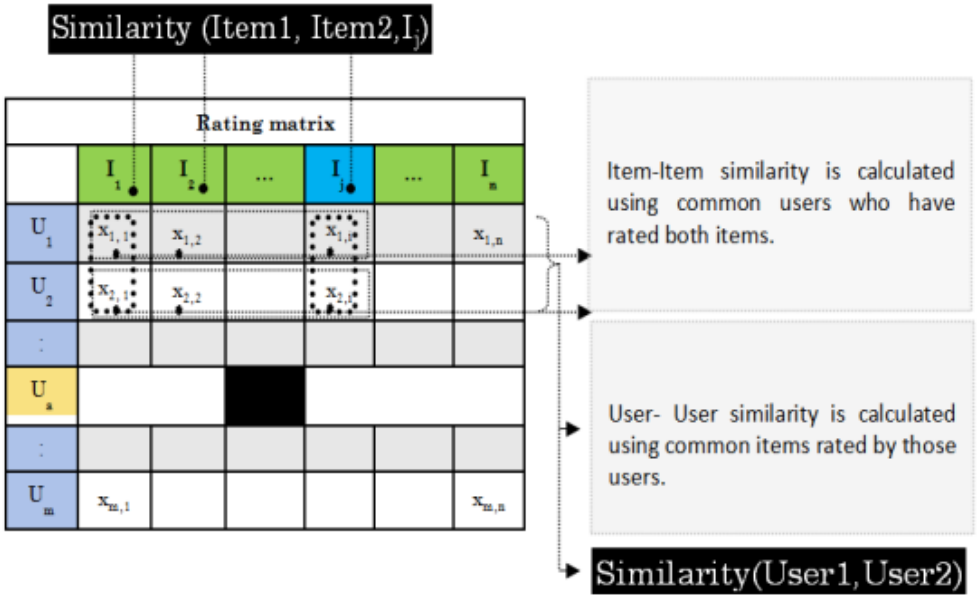

Figure 3.5: Item and user based similarity Neighborhood-based CF

3.5.1 Cosine similarity:

Usually cosine similarity metric is used for estimate the similarity between two objects ,This technique presents a user as a vector of ratings rated by himself and an item as a vector of ratings rated by the set of users . The cosine between two vectors representing two users (or items) indicates the similarity value between each other. A value close to 1 indicates that it exists a strong correlation between the two variables. A value close to 0 indicates that there is no correlation (independent variables). the cosine measure for users and items, calculating the Cosine Vector (CV) (or Vector Space) similarity between these vectors indicate the distance of them to each other Billsus and Pazzani (1998, 2000) and Lang (1995):

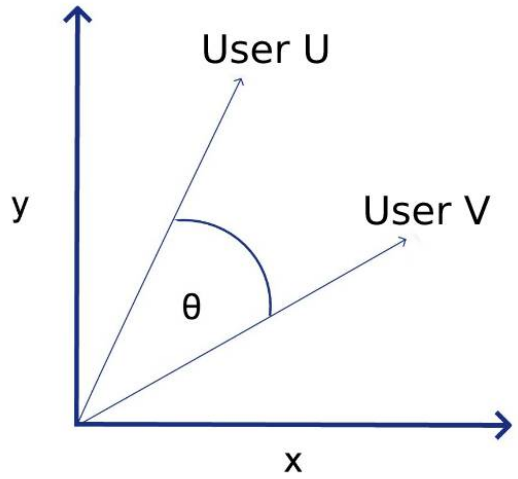

Figure 3.6: User Cosine Similarity Measure

Cosine similarity is measure by looking at the angles between the rating vectors, the smaller the angle between the users' vectors, the more similar the users.

$$cosine(u,v) = \frac{\sum_{i \epsilon I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{u \epsilon I_u} r_{ui}^2} \sqrt{\sum_{u \epsilon I_v} r_{vi}^2}}$$

where I_u and I_v denote the sets of items rated by users u and v, respectively, and I_{uv} denotes the set of items commonly rated by both u and v. r_{ui} and r_{vi} are the ratings values on item i given by users u and v, respectively

For example this is the rating matrix ,

	i_1	i_2
u_1	3	5
u_2	4	1
u_3	2	5

we will look at content is to see the rows of the rating matrix as vectors in space, and then look at the angle between them. the we plot the data iton the coordinate system (This works for more than two items as well)

From the angles between the vectors, it's easy to see that u_1 and u_3 are much closer than u_2 and u_1 when it comes to ratings. Because of that, you can assume that the preferences of u_3 and u_1 are more similar.

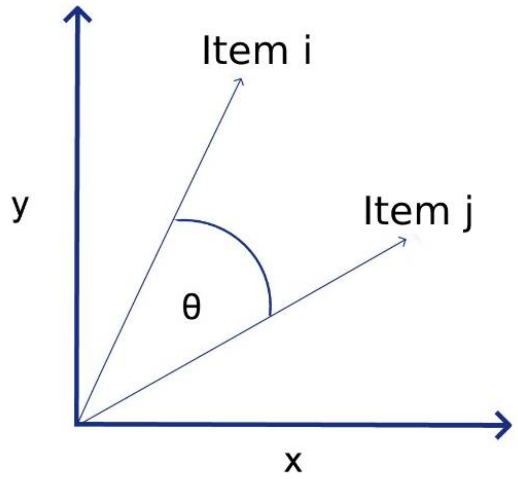

Figure 3.7: Item Cosine Similarity Measure

Cosine item similarity

$$cosine(i,j) = \frac{\sum_{u \epsilon U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \epsilon U_i} r_{ui}^2} \sqrt{\sum_{u \epsilon U_j} r_{uj}^2}}$$

Where U_i and U_j denote the sets of users who rated the items i and j, respectively, and U_{ij} represents the set of users who rated both items i and j. r_{ui} and r_{uj} are the rating values assigned by the same user u on items i and j, respectively

A shortcoming of this measure is that it does not examine the differences in the mean and variance of the ratings made by user's u and v.

3.5.2 Adjusted cosine vector:

The Adjusted cosine measure calculates the correlation value between two users.

$$ACosine(u,v) = \frac{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_i})(r_{vi} - \bar{r_i})}{\sqrt{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_i})^2} \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - \bar{r_i})^2}}$$

Where I_{uv} denotes the set of items commonly rated by both u and v. $\bar{r_i}$ denotes the average ratings on i. r_{ui} and r_{vi} denote, respectively, the ratings of the user u and v on the item i.

$$ACosine(i,j) = \frac{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_u})(r_{uj} - \bar{r_u})}{\sqrt{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{u \epsilon U_{ij}} (r_{uj} - \bar{r_u})^2}}$$

Where U_{ij} represents the set of users who rated both items i and j. $\bar{r_u}$ denotes the average ratings by u. We mention that r_{ui} and r_{uj} are the ratings of user u on items i and j, respectively.

3.5.3 Pearson correlation:

This measure was proposed by Karl Pearson (Pearson, 1895) to measure linear relationships and became widely used in statistics field.
Pearson's correlation, measures the linear correlation between two vectors of ratings. PCC formula returns a value between -1 and 1, where:

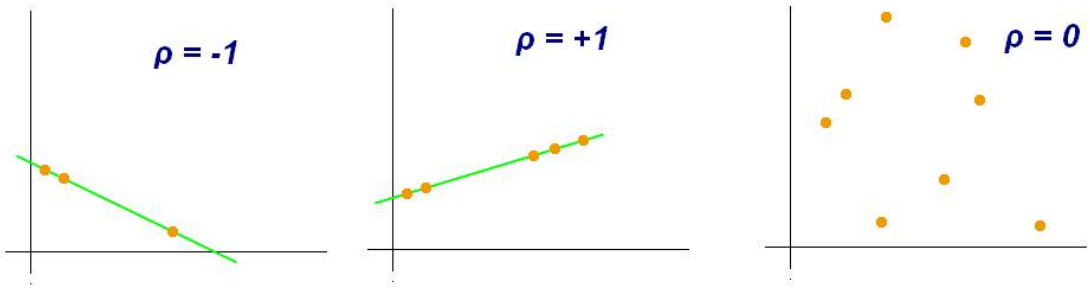

Figure 3.8: Pearsons correlation

1 indicates a strong positive correlation,

-1 indicates a strong negative correlation

0 indicates no correlation at all

Pearson's correlation (PC) is a well-known metric that compares ratings where the effects of mean and variance have been eliminated is the Pearson Correlation (PC) similarity calculates the similarity between two users u and v.

$$PCC(u,v) = \frac{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - \bar{r_v})^2}}$$

Where I_{uv} denotes the set of items commonly rated by both u and v. $\bar{r_u}$ and $\bar{r_v}$ denote the average ratings of the users u and v on item i in I_{uv} , respectively. r_{ui} and r_{vi} are ratings of users u and v on the same item i. this Formula calculates the similarity between two items i and j:

$$PCC(i,j) = \frac{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_i})(r_{uj} - \bar{r_j})}{\sqrt{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_i})^2} \sqrt{\sum_{u \epsilon U_{ij}} (r_{uj} - \bar{r_j})^2}}$$

let U_{ij} denote the set of users who provided a rating for both i and j , i.e $u(i,j) = u(i) \cap u(j)$, U_{ij} represents the set of users who rated both items i and j, $\bar{r_i}$ and $\bar{r_j}$ denote the average ratings on i and j in U_{ij}, respectively. r_{ui} and r_{uj} are ratings of user u on items i and j.

3.5.4 The Jaccard coefficient:

The Jaccard index, denoted by J,computes the similarity of two sets.The Jaccard coefficient between two finite set is defined as the cardinality of the intersection divided by the cardinality of the union. it measures the ratio of the number of elements shared

between the two sets to the total number of elements in both sets. J index takes a zvalue between 0 and 1, the closer the index to 1, the more similar the two vectors. calculates the Jaccard index for users similarity of two vectors u and v, while u and v can be users (set of ratings assigned by the same user) .

$$J(u,v) = \frac{|u \cap v|}{|u \cup v|}$$

Calculates the Jaccard index for items similarity of two vectors i and j, while i and j can be items (set of ratings assigned by the same item) .

$$J(i,j) = \frac{|i \cap j|}{|i \cup j|}$$

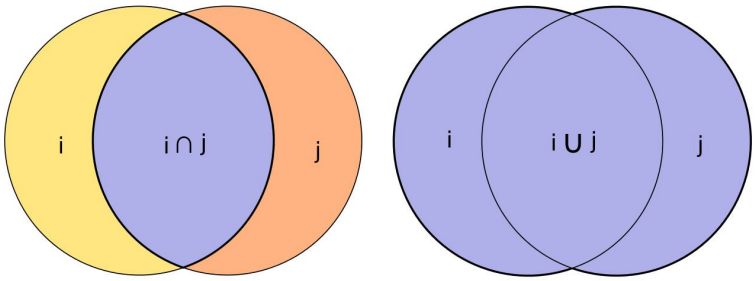

Figure 3.9: Jaccard coefficient

3.5.5 Conditional probability-based similarity:

Karypis (2001) proposed the conditional probability-based metric as a similarity metric for item-based CF Top-N recommendations. The similarity between two items i and j is simply the probability of purchasing (rating) one given that the other has already been purchased. Thus the probability of purchasing a given that b has been purchased is determined as the number of users that purchased both items divided by the total number of users that purchased b. Note that this metric gives asymmetric similarities since $(P(i|j) \notin P(j|i))$. The similarity of i to j is given in Eq. (11) as:

$$Sim(i,j) = P(i|j) = \frac{freq(i,j)}{freq(j)}$$

According to the Deshpande and Karypis (2004), one of the shortcomings of an asymmetric metric is that each item tends to have high conditional probabilities with regard to the most favored items. To allay this shortcoming, the following form of the conditional probability is presented in Deshpande and Karypis (2004):

$$Sim(a,b) = P(a|b) = \frac{\sum_{\forall u: R_{u,b>0}} R_{u,b}}{freq(a), (freq(b)^{\alpha})}$$

where, $\alpha \epsilon [0,1]$ and Freq (a) indicates the number of users that have a transaction on item i in the training data and R (u, b) is the (u, b) element in the normalized user-item matrix.

3.5.6 Kendall's tau

As Spearman's rank correlation coefficient, Kendall's Tau (τ) evaluates statistical monotone relationships based on the ranks of the data. The value returned by τ ranges from -1 (as the rank of one variable increases the other one decreases) to 1 (the ranks of both variables increase together), while 0 indicates that is no relationship between the two variables.
This measure is mainly based on counting the concordant pairs (ordered in the same way) and discordant pairs (ordereddifferently). Kendall's tau for computing the association strength between two vectors of ratings is defined as:

$$\tau = \frac{c - d}{c + d}$$

Where c is the number of concordant pairs and d is the number of discordant pairs.

3.5.7 Euclidean distance

The Euclidean distance denoted by d, from a user u to a user v (or from an item i to an item j) is the length of a line segment between the two users (or items) in the Euclidean space. In practical terms, each user is represented by its Cartesian coordinates with respect to the basis of items (the same thing for an item which is represented with respect to the basis of users) and the distance between two users (or two items) is the absolute value of the numerical difference of their coordinates. The Euclidean distance (d) formula representing the correlation between two users u and v is as follows:

$$d(u, v) = \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - r_{ui})^2}$$

where I_{uv} denotes the set of items commonly rated by both u and v, r_{ui} and r_{vi} denote the rating of the user u and v, respectively, on the item i. This formula provides the Euclidean distance between two items i and j.

$$d(i, j) = \sqrt{\sum_{u \epsilon U_{IJ}} (r_{uj} - r_{ui})^2}$$

where U_{ij} represents the set of users who rated both items i and j,r_{ui} and r_{uj} denote the rating of the user u on items i and j, respectively. The Euclidean distance should be normalized to become a similarity measure.
The Euclidean similarity (ES) measures for users and items, respectively

$$ES(u, v) = \frac{1}{1 + d(u, v)}$$

Between two items i and j

$$ES(i, j) = \frac{1}{1 + d(i, j)}$$

3.5.8 Manhattan distance

The Manhattan distance, also known as city blocks and taxicab, between two vectors is equal to the one-norm of the distance between the vectors . To adapt this measure to the RS domain, we need to represent a user by its Cartesian coordinates with respect to the basis of items (the same thing for an item). The Manhattan distance (d1) between two users u and v is as follows:

$$d_1(u,v) = \sum_{i \epsilon I_{uv}} (|r_{vi} - r_{ui}|)$$

where I_{uv} denotes the set of items commonly rated by both u and v, r_{ui} and r_{vi} denote the rating of the user u and v, respectively, on the item i. the Euclidean distance between two items i and j.

$$d_1(i,j) = \sum_{u \epsilon U_{ij}} (|r_{uj} - r_{ui}|)$$

Where U_{ij} represents the set of users who rated both items i and j, r_{ui} and r_{uj} denote the rating of the user u on items i and j, respectively.
In order to convert the Manhattan distance into a similarity measure (MS)for users and items ,respectively.

$$MS(u,v) = \frac{1}{1 + d_1(u,v)}$$

$$MS(i,j) = \frac{1}{1 + d_1(i,j)}$$

3.6 User-based Rating Prediction

User-based neighborhood recommendation methods predict the rating r_{ui} of a user u for a new item i using the ratings given to i by users most similar to u, called nearest-neighbors. Suppose we have for each user vu a value S_{uv} representing the similarity between u and v.

The k-nearest-neighbors (k-NN) of u is one of the simplest forms of machine learning algorithms mostly used for classification, denoted by $N(u)$, are the k users v with the highest similarity S_{uv} to u. However, only the users who have rated item i can be used in the prediction of r_{ui} and we instead consider the k users most similar to u that have rated i. We note set of neighbors as $N_i(u)$ The rating r_{ui} can be estimated as the average rating given to i by these neighbors :

$$\widehat{r_{ui}} = \frac{1}{|N_i(u)|} \sum_{v \epsilon N_i(u)} r_{vi}$$

As this equation does not take into account the fact that the neighbors can have different levels of similarity. A common solution to this problem is to weigh the contribution of each neighbor by its similarity to u. However, if these weights do not sum to 1, the predicted ratings can be well outside the range of allowed values. Consequently, it is customary to normalize these weights, such that the predicted rating becomes

$$\widehat{r_{ui}} = \frac{\sum_{v \epsilon N_i(u)} S_{uv} r_{vi}}{\sum_{v \epsilon N_i(u)} |S_{uv}|}$$

$|s_{uv}|$ is used instead of s_{uv} v because negative weights can produce ratings outside the allowed range.

3.7 Item-based Rating Prediction

Item-based approaches look at ratings given to similar items. Denote by $N_u(i)$ the items rated by user u most similar to item i. The predicted rating of u for i is obtained as a weighted average of the ratings given by u to the items of $N_u(i)$:

$$\widehat{r_{ui}} = \frac{\sum_{j \epsilon N_u(i)} S_{ij} r_{uj}}{\sum_{j \epsilon N_u(i)} |S_{ij}|}$$

3.8 Rating Normalization

Assigning a rating r_{ui} to an item i for user u , each user has its own personal scale, some users might be reluctant to give high/low scores to items they liked/disliked. Two of the most popular rating normalization schemes that have been proposed to convert individual ratings to a more universal scale are mean-centering and Z-score

- **Mean-centering** In user-based recommendation, a raw rating r_{ui} is transformation to a mean-centered one $h(r_{ui})$ by subtracting to r_{ui} the average $\bar{r_u}$ of the ratings given by user u to the items in I_u.

  $$h(r_{ui}) = r_{ui} - \bar{r_u}$$

  Using this approach the user-based prediction of a rating r_{ui} is obtained as

  $$\widehat{r_{ui}} = \bar{r_u} + \frac{\sum_{v \epsilon N_i(u)} s_{uv}(r_{vi} - \bar{r_v})}{\sum_{v \epsilon N_i(u)} |s_{uv}|}$$

  The Mean-centering normalization technique is most often used in item-based, where The item-mean-centered normalization of r_{ui} is given by

  $$h(r_{ui}) = r_{ui} - \bar{r_i}$$

  $\bar{r_i}$ corresponds to the mean rating given to item i by user in U_i

  Using this approach the item-based prediction of a rating r_{ui} is obtained as:

  $$r_{ui} = \bar{r_i} + \frac{\sum_{j \epsilon N_u(i)} s_{ij}(r_{uj} - \bar{r_j})}{\sum_{j \epsilon N_u(i)} |s_{ij}|}$$

- **Z-score normalization** This is very similar to the above method, just that we also devide by the standard deviation in this case. This method provides a slight lift in the accuracy when compared to mean centering. in addition , z-score is the signed number of deviations from mean indicating that a datum is above the mean if positive and below the mean otherwise.

  In user-based methods, the normalization of a rating r_{ui} divides the user-mean-centered rating by the standard deviation σ_u of the ratings given by user u:

  $$h(r_{ui}) = \frac{r_{ui} - \bar{r_u}}{\sigma_u}$$

  A user-based prediction of rating r_{ui} using g this normalization approach would therefore be obtained as :

  $$\widehat{r_{ui}} = \bar{r_u} + \sigma_u \frac{\sum_{v \epsilon N_i(u)} s_{uv}(r_{vi} - \bar{r_v})/\sigma_v}{\sum_{v \epsilon N_i(u)} |s_{uv}|}$$

  The z-score normalization of r_{ui} in item-based methods divides the item- mean-centered rating by the standard deviation of ratings given to item i:

$$h(r_{ui}) = \frac{r_{ui} - \bar{r_i}}{\sigma_i}$$

The item-based prediction of rating r_{ui} would then be

$$\widehat{r_{ui}} = \bar{r_i} + \sigma_i \frac{\sum_{j \epsilon N_u(i)} s_{ij}(r_{uj} - \bar{r_j})/\sigma_j}{\sum_{j \epsilon N_u(i)} |s_{ij}|}$$

3.9 Neighborhood Selection

The number of nearest-neighbors to select have a serious impact on the quality of the recommender system.We call it a neighborhood because we talk about items with a small distance between them,In this section, I'll cover three ways to define and calculate such a neighborhood: Top-N filtering , Threshold filtering, and Negative filtering.

3.9.1 Top-N Recommendation:

For each user or item, only a list of the N nearest-neighbors and their respective similarity weight is kept. To avoid problems with efficiency or accuracy, N should be chosen carefully. Thus, if N is too large, an excessive amount of memory will be required to store the neighborhood lists and predicting ratings will be slow. On the other hand, selecting a too small value for N may reduce the coverage of the recommendation method, which causes some items to be never recommended.

User-Based Top-N Recommendation

User-based recommendation systems compute the top-N recommended items for that user as follows. First they identify the k most similar user in the database.

Once this set of the k most similar have been discovered, then we identify the set C of items purchased by the group as well as their frequency. Using this set, user-based techniques then recommend the N most frequent items in C that are not already in the set of items that have already been purchased by the user for which we want to compute the top-N recommendations.

Item-Based Top-N Recommendation

Item-based top-N recommendation algorithms that use item-to-item similarity to compute the relations between the items,for each item , the k most similar items are computed, and their corresponding similarities are recorded.Now, for each user that has purchased a set T of items that have already been purchased by the customer for which we want to compute the top-N r recommendations items as follows.

First, we identify the set C of candidate recommended items by taking the union of the k most similar items for each item $i \epsilon T$, and removing from the union any items that are already in T Then, for each item $c \epsilon C$ we compute its similarity to the set T as the sum of the similarities between all the items $i \epsilon T$ and c, using only the k most similar items of i.
Finally, the items in C are sorted in non-increasing order with respect to that similarity, and the first N items are selected as the top-N recommended set.

3.9.2 Threshold filtering:

Instead of keeping a fixed number of nearest-neighbors, this approach keeps all the neighbors whose similarity weight has a magnitude greater than a given threshold s_{min}. While this is more flexible than the previous filtering technique, as only the most significant neighbors are kept, the right value of s_{min} may be difficult to determine.

3.9.3 Negative filtering:

In general, negative rating correlations are less reliable than positive ones. Intuitively, this is because strong positive correlation between two users is a good indicator of their belonging to a common group However, although negative correlation may indicate membership to different groups, it does not tell how different these groups are, or whether these groups are compatible for other categories of items. While negative correlations provide no significant improvement in the prediction accuracy,

The three filtering approaches are not mutually exclusive and can be combined to fit the needs of the recommendation systems.

3.10 Dimensionality Reduction

The dimension Reduction phase transforms the original user item matrix into a lower dimensional space to address the sparsity and scalability problem often encountered in collaborative filtering recommendation scenarios.

The original representation of the input data to a collaborative filtering system is an $n \times m$ user item matrix $R_{n \times m}$, This representation may potentially pose sparsity and scalability problems for collaborative filtering systems (Sarwar et al. 2000).
In practice, when a large set of items is available, users may have rated or chosen a very low percentage of items, resulting in a very sparse user preference matrix.
As a consequence, a collaborative filtering recommendation system may be unable to make any recommendations for a paricular user.
When building collaborative filtering,The user-item matrix or the rating matrix usualy contain very large numbers of users or items , we are dealing with millions of users rating millions of items.

This may be problematic during data exploration and analysis processes.
Dimensionality reduction is one of the oldest approaches used to respond to this problem,To improve performance, speed up calculations, and avoid the curse of dimensionality.

The aim of dimensionality reduction is to select or extract an optimal subset from the matrix.

The main aims of dimensionality reduction are as follows:

- Easy visualize and understand data.
- Reduce the required storage space.
- Reduce learning and use times.

3.11 Matrix factorization techniques

Matrix factorization is a subset of dimensionality reduction in which a data user-item matrix is reduced to the product of many low-rank matrices.

Recommendation systems mainly base their suggestions on rating data of items, which are often placed in a matrix with one representing users and the other representing items of interest, the ratings are given explicitly by users creating a sparse user–item rating matrix, because an individual user is likely to rate only a small fraction of the items that belong to the item set, i.e, many values in the rating matrix are null since all users do not rate all items

and due to the high-dimensional data of the users and items,, MF was proposed to deal with these problems. Matrix factorization is defined as the decomposition of the initial matrix in two or more smaller metrics to reduce required storage space and processing time.

Another challenging issue with this user–item rating matrix is scalability of data (i.e., the large number of possible registered users or inserted items), which may affect the time performance of a recommendation algorithm We can deal with all the mentioned challenges by applying matrix decomposition methods (also known as factorization methods).

Matrix factorization techniques gained popularity during the Netflix challenge, because they are fast and accurate.

Matrix Factorization decompose matrix into a product of Three matrices, it's a powerful technique to find the hidden structure behind the data.
The main popular three methods are:

- Singular Value Decomposition (SVD)
- Principal Component Analysis (PCA)
- Probabilistic Matrix Factorization (PMF)

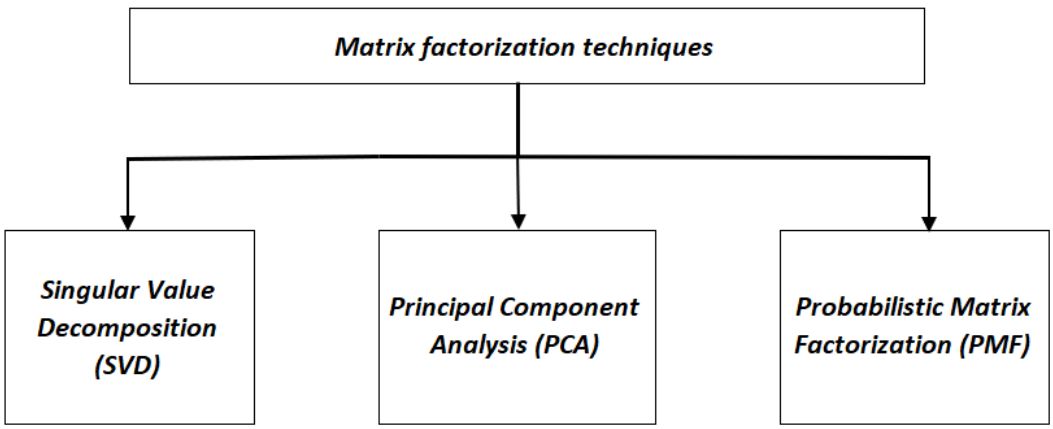

Figure 3.10: Matrix factorization techniques

3.11.1 Singular Value Decomposition (SVD)

is the most popular matrix factorization methods that can be applied in recommendeation systems to solve sparsity and scalability problem ,The Singular Value Decomposition (SVD) used as dimensionality reduction technique also to produce a low-dimensional representation of the original user-item matrix .

It is nothing more than decomposing vectors into orthogonal axes by transformating data from a high-dimensional space into a low-dimensional space . The key issue in an SVD decomposition is to find a lower dimensional feature space, thus can be expressed in the following way.

is so that the low-dimensional

$$SVD(A) = U \times S \times V^T$$

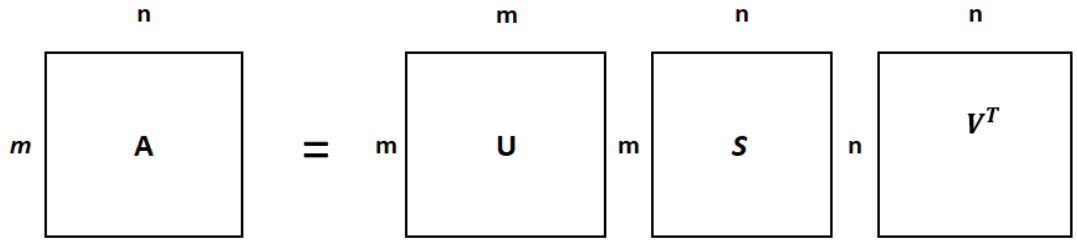

Figure 3.11: Singular Value Decomposition (SVD)

Where A is the $m \times n$ matrix with rank r, U and V are $m \times m$ and $n \times n$ are orthogonal matrices respectively. S is a diagonal matrix $m \times n$ singular orthogonal matrix with non-negative elements and

the $m \times m$ matrix U is called orthogonal where $U^T U = I_{m \times m}$, Also matrix V is orthogonal where $V^T V = I_{n \times n}$.

The initial r diagonal elements of S , $(s_1, s_2, s_3, ..s_r)$ are called the singular values of matrix A have the property that $s_i > 0$ and $s_1 >= s_2 >= s_3 >= ... >= s_r$.

Accordingly, the first r columns of U are called the left singular vectors (the eigenvictors of AA^T) and The first r columns of V are called the right singular vectors (the eigenvectors of the $A^T A$).

If we focus only on these r nonzero singular values, the effective dimensions of the SVD matrices U, S and V will become $m \times r$,$r \times r$ and $r \times n$ respectively.

In SVD, particularly useful in the case of Recommendation Systems, is that it can provide the best low-rank approximation of the original matrix A. By retaining the first $k << r$ singular values of S and discarding the rest, which can be translated as keeping the k largest singular values, based on the fact that the elements in S are sorted.

The resulting diagonal matrix S is termed S_k. Matrices U and V should be also reduced accordingly. U_k is produced by removing $r - k$ columns from matrix U. V_k is produced by removing $r - k$ rows from matrix V . Matrix A_k which is defined as:

$$A_k = U_k \times S_k \times V_k^T$$

Stands for the closest linear approximation of the original matrix A with reduced rank k. Once this transformation is completed, users and items can be represented as points in the k-dimensional space.

[B.Sarwar et al. 2001] Applied the Matrix Factorization model SVD to reduce the dimensionality of a rating matrix. Using the MovieLens dataset, they selected 943 users to form a 943 x 1682 matrix that is 95.4% sparse (each user's on average rates 5% of the 1682 movies). They first fill missing values using user and movie rating averages and then apply SVD. For this large dataset Item-Based technique provide optimal accuracy with significantly faster and high-quality online recommendations than the user-user (k-Nearest-Neighbor) method.

We define the original user-item matrix R, of size mn, which includes the ratings of m users on n items r_{ij}, In order to eliminate all missing rating r_{ui} we replace them by Computing the average of a $\bar{r_u}$ then we normalized matrix R. Then the normalized matrix R

$$R = U \times S \times V^T$$

Perform the dimensionality reduction step by keeping only k diagonal entries from matrix S to obtain a $k \times k$ matrix, S_k. Similarly, matrices U_k and V_k of size $m \times k$ and $k \times n$ are generated.
The "reduced" user-item matrix, R, is obtained by

$$R = U_k \times S_k \times V_K$$

3.11.2 Principal Component Analysis (PCA)

Principal component analysis (PCA) is also the powerful technique of dimensionality reduction that transforms high-dimensions data into lower-dimensions while retaining as much information as possible and is a particular realization of the matrix factorization approach.
PCA allow to obtain an ordered list of components that account for the largest amount of the variance from the data in terms of least square errors, The amount of variance captured by the first component is larger than the amount of variance on the second component and so on. We can reduce the dimensionality of the data by neglecting those components. As PCA is a statistical technique, which is used to reduce the dimensionality of the data for computational efficiency consisting of many correlated variables. It retains the variation present in the dataset, up to the maximum extent, and uses the ideas of variances and co-variances.

PCA is a multivariate analysis technique that transforms a set of correlated variables into a set of orthogonal components. Although correlated components are required to reproduce the total variability of the original variables, much of this variability can be often explained by a small number k of the principal components (PC's). The k PC's can then replace the original variables, thus compressing the original data set into a smaller one that consists of the k PC's .

[Goldberg et al] Proposed an approach to use Principal Component Analysis (PCA) in the context of an online joke recommendation system. Their system, known as Eigentaste's, In Eigentaste they addressed sparseness using universal queries, which insure that all users rate a common set of k-items. So, resulting sub-matrix will be dense and directly compute the square symmetric correlation matrix and then did linear projection using Principle Component Analysis (PCA), a closely-related factor analysis technique first described by Pearson in 1901. Like SVD, PCA reduces dimensionality of matrix by optimally projecting highly correlated data along a smaller number of orthogonal dimensions.

3.11.3 Probabilistic Matrix Factorization (PMF)

Probabilistic Matrix Factorization (PMF) is a probabilistic linear model with Gaussian observation noise. The Probabilistic Matrix Factorization (PMF) models the user preference matrix as a product of two lower-rank user and item matrices. Suppose we have N users and M items . Let R_{ij} be the rating value of user i for item j , U_i and V_j represent D-dimensional user-specific and item-specific latent feature vectors respectively.

The conditional distribution over the observed ratings $R \epsilon \mathbb{R}^{N \times M}$ and the prior distributions over $U \epsilon \mathbb{R}^{D \times N}$ and $V \epsilon \mathbb{R}^{D \epsilon M}$ are given by :

$$P(R|U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} [N(R_{ij}|U_i^T V_j, \sigma^2)]^{I_{ij}}$$

Where $N(x|u, \sigma^2)$ is the probability density function of the Gaussian distribution with mean u and variance σ^2 ,I_{ij} is the indictor variable that is equal to 1 if user i rated item j and equal to 0 otherwise.

We also place zero-mean spherical Gaussian priors[1, 11] on user and item feature vectors:

$$P(U|\sigma_u^2) = \prod_{i=1}^{N} N(U_i|0, \sigma_U^2 I)$$

$$p(V|\sigma_V^2) = \prod_{j=1}^{M} N(V_j|0, \sigma_V I))$$

Chapter 4

Covariance Distance

The objective of a recommendation system is to provide users with personalized recommendations while selecting an item among a set of items (restaurants, books, etc...). Collaborative filtering is one of the most successful approaches for an online store to make personalized recommendations through its recommendation systems.
The neighborhood-based is one of the most successful and widely used methods of online product recommendation. The most critical component of this method is the mechanism of finding similarities between items or users in item-based neighbourhood or user-based neighbourhood respectively.
In addition, item-based neighborhood collaborative filtering and user-based neighborhood algorithms produce predictions based on similarities between items and users, respectively, using the user's ratings. The calculation of similarities metrics, Studies, and real-world implementations so far has relied on traditional distance and vector similarity measures, mainly although the most widely used similarity measures such as cosine, Pearson correlation, and adjusted cosine have been proved to be successful in many studies in terms of their effectiveness in the recommendation systems. those similarity metrics have been mentioned in the previous chapter. They are limited to be used, This chapter presents a new similarity measure that focuses on improving recommendation systems performance and effectiveness.....

Predefined similarity measures

The definitions of the mainly similarity measures that have been used for neighborhood-based recommendation systems are summarized in table .
The first one,The cosine measure looks at the angle between two vectors of ratings where a smaller angle is regarded as implying greater similarity. The second one,Adjusted Cosine Similarity that has proven to be more efficient for the item-based neighborhood [**B. Sarwar et al. 2001**]. The third one Pearson's correlation, measures the linear correlation between two vectors of ratings .

TABLE OF SIMILARITIES

Measure	User Similarity Measure	Item similarity Measure
Cosine	$cosine(u,v) = \frac{\sum_{i \epsilon I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{u \epsilon I_u} r_{ui}^2} \sqrt{\sum_{u \epsilon I_v} r_{vi}^2}}$	$cosine(i,j) = \frac{\sum_{u \epsilon U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \epsilon U_i} r_{ui}^2} \sqrt{\sum_{u \epsilon U_j} r_{uj}^2}}$
Adjusted cosine	$ACosine(u,v) = \frac{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_i})(r_{vi} - \bar{r_i})}{\sqrt{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_i})^2} \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - \bar{r_i})^2}}$	$ACosine(i,j) = \frac{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_u})(r_{uj} - \bar{r_u})}{\sqrt{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{u \epsilon U_{ij}} (r_{uj} - \bar{r_u})^2}}$
Pearson correlation	$PCC(u,v) = \frac{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_u})(r_{vi} - \bar{r_v})}{\sqrt{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - \bar{r_v})^2}}$	$PCC(i,j) = \frac{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_i})(r_{uj} - \bar{r_j})}{\sqrt{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_i})^2} \sqrt{\sum_{u \epsilon U_{ij}} (r_{uj} - \bar{r_j})^2}}$
The Jaccard coefficient	$J(u,v) = \frac{\lvert u \cap v \rvert}{\lvert u \cup v \rvert}$	$J(i,j) = \frac{\lvert i \cap j \rvert}{\lvert i \cup j \rvert}$
Euclidean distance	$d(u,v) = \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - r_{ui})^2}$	$d(i,j) = \sqrt{\sum_{u \epsilon U_{IJ}} (r_{uj} - r_{ui})^2}$
Manhattan distance	$d_1(u,v) = \sum_{i \epsilon I_{uv}} (\lvert r_{vi} - r_{ui} \rvert)$	$d_1(i,j) = \sum_{u \epsilon U_{ij}} (\lvert r_{uj} - r_{ui} \rvert)$

4.1 Distance Covariance

Classical dependence measures such as Pearson correlation, Spearman's ρ, and Kendall's τ can detect only monotonic or linear dependence. To overcome these limitations, [**Szekely et al.2007**]. proposed distance covariance and its derived correlation. The distance covariance is a weighted L2 distance between the joint characteristic function and the product of marginal distributions. This measure can detect the presence of a dependence structure when this ample size is large enough.

There has been a considerable recent interest in measuring dependence by employing the concept of distance covariance, a new and appealing measure of dependence for random variables. The distance covariance a powerful measure of dependence between sets of multivariate random variables based on the assumption that the data are $i.i.d$ (independent and identically distributed) . However, as this assumption is often violated in many practical problems,Rémillard (2009) proposed to extend the notion of distance covariance to the case of dependent data.
The distance covariance has the crucial feature that it equals zero if and only if the sets are mutually independent.
Hence it can be applied to multivariate data to detect arbitrary types of non-linear associations between sets of variables.

4.1.1 The Distance Covariance Function

The distance covariance function as a new measure of dependence between multivariate random vectors, say X and Y, of arbitrary, not necessarily equal in dimensions,let note p and q be the dimension of X and Y,respectively.
A crucial feature of the distance covariance is that it equals zero if and only if X and Y and are mutually independent.Hence the distance covariance is sensitive to arbitrary dependencies,this is in contrast to the classical covariance, which is generally capable of detecting only linear dependencies. where the distance covariance have been applied to multivariate data to detect arbitrary types of non-linear associations between sets of variables.

The definition of distance covariance relies on the joint characteristic function of X and Y,which is denoted by

$$\phi_{X,Y}(t,s) = E[\exp^{i(t'X + s'Y)}]$$

Where $(t,s) \epsilon R^{p+q}$, and $i^2 = -1$.

The corresponding marginal characteristic function of multivariate random vector X and Y are denoted respectively by

$$\phi_X(t) = E[\exp(it'X)]$$
$$\phi_Y(s) = E[\exp(is'Y)]$$

The distance covariance function is defined as the nonnegative square root of a weighted L_2 distance between the joint and the product of the marginal characteristic functions of two random vectors, namely

$$V^2(X,Y) = \int_{R^{p+q}} |\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s)|^2 \omega(t,s)\, dt\, ds \tag{4.1}$$

Where $\omega(.,.) : R^{p+q} \to R$ is a weight function for which the previous integral exists . The distance covariance function $V^2(X,Y)$ leads to the definition of the distance correlation function between X and Y , which is the positive square root of

$$R^2(X,Y) = \begin{cases} \frac{V^2(X,Y)}{\sqrt{V^2(X,X)V^2(Y,Y)}}, & V^2(X,X)V^2(Y,Y) > 0 \\ 0, & V^2(X,X)V^2(Y,Y) = 0. \end{cases}$$

The previous display shows that the distance correlation function is a coefficient analogous to coefficient. the coefficient measures the linear relationship between X and Y and can be zero even when the variables are dependent,

The choice of the weight function $\omega(.,.)$ is crucial. Choosing the non-integrable weight function of the form

$$\omega(t,s) = (c_p c_q |t|_p^{1+p} |s|_q^{1+q})^{-1}$$

where $c_d = \pi^{(1+d)/2}/\Gamma((1+d)/2)$,with $\Gamma(.)$ denoting the Gamma function.
$R^2(X,Y)$ is scale and rotation invariant. In particular $\omega(t,s)$ is the only weight function such that $V^2(X,Y)$ is rigid motion invariant and scale equivariant However, the choice of $\omega(t,s)$ yields to computational advantages.

Denote by $|x|_p$ The Euclidean norm [1] of x in R^p.

The distance covariance $V^2(X,Y)$ between X and Y will be defined as the following equation :

$$V^2(X,Y) = \int_{R^{p+q}} \frac{|\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s)|^2}{c_p c_q |t|_p^{1+p} |s|_q^{1+q}}\, dt\, ds$$

As the $V^2(X,Y)$ is nonnegative, it follows that $V^2(X,Y) >= 0$ Further more $V^2(X,Y) < \infty$ whenever X and Y have finite first moments.
An advantage of this equation is that it directly implies one of the most important properties of the distance covariance, the characterization of independence. For all X and Y ,$V^2(X,Y) = 0$ if and only in X and Y are independent.

PROOF
If X and Y are independent then $\phi_{X,Y}(t,s) = \phi_X(s)\phi_Y(t)$ for all s and f ,hence

[1] The Euclidean norm (L_2 norm) of a vector $x \epsilon R^p$ is defined by $|x|_p = \sqrt{x_1^2 + x_2^2 + x_3^2 + ... + x_p^2}$

$V^2(X,Y) = 0$
If X and Y are not independent then the functions $\phi_{X,Y}(t,s)$ and $\phi_X(s)\phi_Y(t)$ are not identical. Since characteristic functions are continuous then there exists an open set $A \subseteq R^p \times R^q$ such that $|\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s)|^2 > 0$ for all $(s,t)\epsilon A$, hence in the equation $V^2(X,Y) > 0$.

The main properties of $V^2(X,Y)$,$R^2(X,Y)$, when $\omega(t,s)$ is used, are given by the following:

- If $E(|X|_p + |Y|_q) < \infty$, then the distance correlation,$R(X,Y)$, satisfies $0 <= R(X,Y) <= 1$,and $R(X,Y) = 0$ if and only if X and Y are independent.
- $R(X,Y)$ is invariant under orthogonal transformations of the form $(X,Y) \rightarrow (\alpha_1 + b_1 C_1 X, \alpha_2 + b_2 C_2 Y)$, where α_1, α_2 are arbitrary vectors,b_1, b_2 are arbitrary nonzero numbers and ,C_1, C_2 are arbitrary orthogonal matrices.
- If $p = q = 1$ and X and Y have standard normal distributions with $r = Cov(X,Y)$,then $R(X,Y) <= |r|$ and

$$R^2(X,Y) = \frac{r \arcsin r + \sqrt{1-r^2} - r \arcsin r/2 - \sqrt{4-r^2} + 1}{1 + \pi/3 - \sqrt{3}}$$

- When $E(|X|_p^2 + |X|_q^2) < \infty$ then

  $V^2(X,Y) = E|X - X'|_p|Y - Y'|_q + E|X - X'|_p|Y - Y''|_q - 2E|X - X'|_p|Y - Y''|_q$

  with (X', Y') and (X'', Y'') being independent copies of (X,Y)

Theorem Suppose that $(X_1, Y_1), ..., (X_4, Y_4)$ are independent, identically distributed (i.i.d.), Then

$$V^2(X,Y) = E[|X_1 - X_2|.|Y_1 - Y_2| - 2|X_1 - X_2|.|Y_1 - Y_3| + |X_1 - X_2|.|Y_3 - Y_4|] \tag{4.2}$$

Proof

Note that $< ., . >$ the corresponding inner product.

$|\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s)|^2$

$= (\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s))\overline{(\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s))}$

$= E[e^{i<t,X_1-X_2>+i<s,Y_1-Y_2>} - 2e^{i<t,X_1-X_2>+i<s,Y_1-Y_3>} + e^{i<t,X_1-X_2>+i<s,Y_3-Y_4>}]$

As this last expression is real,any term of the form e^{iz}, $z\epsilon R$ can be replaced by $cosz$

$$V^2(X,Y) = \int_{R^{p+q}} \frac{|\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s)|^2}{c_p c_q |t|_p^{1+p} |s|_q^{1+q}}\, dt\, ds$$

$$c_p c_q V^2(X,Y) = \int_{R^{p+q}} \frac{A_{12}(t,s) - 2A_{13}(t,s) + A_{34}(t,s)}{|t|_p^{1+p} |s|_q^{1+q}}\, dt\, ds \tag{4.3}$$

Where, for each (j,k),

$$A_{jk}(t,s) = E(cos(< t, X_1 - X_2 > + < s, Y_j - Y_k >)) \tag{4.4}$$

Replacing t by $-t$ in (4.3) we also obtain

$$c_p c_q V^2(X,Y) = \int_{R^{p+q}} \frac{A_{12}(t,-s) - 2A_{13}(t,-s) + A_{34}(t,-s)}{|t|_p^{1+p} |s|_q^{1+q}}\, dt\, ds \tag{4.5}$$

and by adding (4.3) and (4.6) we find that

$$c_p c_q V^2(X,Y) = \int_{R^{p+q}} \frac{B_{12}(t,s) - 2B_{13}(t,s) + B_{34}(t,s)}{|t|_p^{1+p} |s|_q^{1+q}}\, dt\, ds \tag{4.6}$$

where for each (j,k),

$$B_{jk}(t,s) = \frac{1}{2}(A_{jk}(t,s) + A_{jk}(t,-s)) \tag{4.7}$$

On applying to $A_{jk}(t,s)$ (4.4) and $B_{jk}(t,s)$ (4.7) the trigonometric identity,

$$\cos(x+y) + \cos(x-y) = 2\cos x \cos y$$

We deduce that

$$B_{jk}(t,s) = E(cos(< t, X_1 - X_2 >< s, Y_j - Y_k >)) \tag{4.8}$$

For $j,k\epsilon 1,2,3,4$ we apply to Eq (4.8)

$$\begin{aligned}\cos < t, X_1 - X_2 > \cos < s, Y_j - Y_k >= \\ (1 - \cos < t, X_1 - X_2 >)(1 - \cos < s, Y_j - Y_k >) \\ -1 + \cos < t, X_1 - X_2 > + \cos < s, Y_j - Y_k >\end{aligned} \tag{4.9}$$

Then we obtain

$$\begin{aligned}c_p c_q V^2(X,Y) = \int_{R^{p+q}} (E[(1 - \cos < t, X_1 - X_2 >)(1 - \cos < s, Y_1 - Y_2 >) \\ -2E[(1 - \cos < t, X_1 - X_2 >)(1 - \cos < s, Y_1 - Y_3 >)] + E[(1 - \cos < t, X_1 - X_2 >) \\ (1 - \cos < s, Y_3 - Y_4 >)]) \frac{dtds}{|t|_p^{1+p}|s|_q^{1+q}}\end{aligned} \tag{4.10}$$

Which is obtained by decomposing all summands on the right-hand side using Eq.(4.9) and observing that all terms which are not of the form $E(\cos < t, X_i - X - j > \cos < s, Y_l - Y_k >)$ cancel each other. By applying the Fubini-Tonelli Theorem and the linearity of expectation and integration, we obtain

$$\begin{aligned}c_p c_q V^2(X,Y) = E \int_{R^{p+q}} [(1 - \cos < t, X_1 - X_2 >)(1 - \cos < s, Y_1 - Y_2 >) \\ -2(1 - \cos < t, X_1 - X_2 >)(1 - \cos < s, Y_1 - Y_3 >) \\ +(1 - \cos < t, X_1 - X_2 >)(1 - \cos < s, Y_3 - Y_4 >)] \frac{dtds}{|t|_p^{1+p}|s|_q^{1+q}}\end{aligned} \tag{4.11}$$

4.1.2 Estimation of distance covariance

The empirical distance covariance and correlation measures are functions of the double centred distance matrices of the samples Suppose that (X_i, Y_i) , $i = 1, ..., n$ is a random sample from the joint distribution of the random vectors X and Y. Based on this sample, consider the $n \times m$ Euclidean pairwise distance matrices with elements

$$(a_{ij}) = (|X_i - X_j|_p)$$

and

$$(bij) = (|Y_i - Y_j|_q)$$

These matrices are double centred so that their row and means are equal to zero. In other words, let

$$A_{ij} = a_{ij} - \bar{a_{i.}} - \bar{a_{.j}} + \bar{a_{..}}$$

$$B_{ij} = b_{ij} - \bar{b_{i.}} - \bar{b_{.j}} + \bar{b_{..}}$$

where $\bar{a_{i.}} = (\sum_{j=1}^{n} a_{ij})/n, \bar{a_{.j}} = (\sum_{i=1}^{n} a_{ij})/n, \bar{a_{..}} = (\sum_{i,j=1}^{n} a_{ij})/n^2$

Similarly, we define the quantities $b_{i.}$, $b_{.j}$ and $b_{..}$. The sample distance covariance is defined by the square root of the statistic

$$\hat{V}^2(X,Y) = \frac{1}{n^2}\sum_{i,j=1}^{n} A_{ij}B_{ij}$$

We compute the squared sample distance correlation $\hat{R}^2(X,Y)$

$$\hat{R}^2(X,Y) = \begin{cases} \frac{\hat{V}^2(X,Y)}{\sqrt{\hat{V}^2(X,X)\hat{V}^2(Y,Y)}}, & \hat{V}^2(X,X)\hat{V}^2(Y,Y) > 0 \\ 0, & \hat{V}^2(X,X)\hat{V}^2(Y,Y) = 0. \end{cases}$$

Furthermore, note that by multiplying out $\hat{V}^2(X,Y)$ we obtain the alternative expression

$$\hat{V}^2(X,Y) = \frac{1}{n^2}\sum_{i,j=1}^{n} a_{ij}b_{ij} + \frac{1}{n^4}\sum_{i,j=1}^{n} a_{ij}\sum_{i,j=1}^{n} b_{ij} - \frac{2}{n^3}\sum_{i,j,k=1}^{n} a_{ij}b_{jk}$$

The main properties of sample distance covariance and correlation functions are the following:

- The estimators $\hat{V}^2(X,Y)$ and $\hat{R}^2(X,Y)$ are both strongly consistent, that is, they both converge almost surely to their population $V^2(X,Y)$ and $R^2(X,Y)$ respectively, as n tends to infinity, provided that $E(|X|_P) < \infty$ and $E(|Y|_q) < \infty$
- $\hat{V}^2(X,Y) >= 0$ the equality holds when X and Y are independent.
- $0 <= \hat{R}(X,Y) <= 1$.
- $\hat{R}(X, \alpha + bXC) = 1$,where ,α is a vector ,b is a nonzero real number and C is an orthogonal matrix.

$\hat{V}^2(X,Y)$ is not an unbiased estimator of $V^2(X,Y)$, where we have

$$E[\hat{V}^2(X,Y)] = \frac{(n-1)(n-2)^2}{n^3}V^2(X,Y) + \frac{2(n-1)^2}{n^3}\gamma - \frac{(n-1)(n-2)}{n^3}\alpha\beta$$

Where

$$\alpha = E|X - X'|_p$$

,

$$\beta = E|Y - Y'|_q$$

and

$$\gamma = E[|X - X'|_p|Y - Y'|_q]$$

X' (respectively Y')denotes an independent of X (respectively Y). An unbiased estimator of squared distance covariance proposed by Székely and Rizzo (2014) is given by

$$\hat{V}_U(X,Y) = \frac{1}{n(n-3)}\sum_{i\neq j} \tilde{A}_{ij}\tilde{B}_{ij}$$

for $n > 3$, where the so-called U-centred matrices, $\tilde{A}_{ij}$ have the additional property that $E[\tilde{A}_{ij}] = 0$ for all i, j and are defined by

$$\tilde{A}_{ij} = \begin{cases} a_{ij} - \frac{1}{n-2}\sum_{l=1}^{n} a_{il} - \frac{1}{n-2}\sum_{k=1}^{n} a_{kj} + \frac{1}{(n-1)(n-2)}\sum_{k,l=1}^{n} a_{kl}, & i \neq j \\ 0, & i = j \end{cases}$$

4.1.3 Asymptotic Distribution

Under the null hypothesis of independence between the random vectors X and Y.

$$n\hat{V}^2(X,Y) \xrightarrow{n\to\infty} \to \sum_{j=1}^{\infty} \lambda_j Z_j \tag{4.12}$$

In distribution, where Z_j are independent standard normal variables and λ_j are eigenvalues that depend on the joint distribution of the random vectors (X,Y), provided that $E[|X|_P] < \infty$ and $E[|Y|_q] < \infty$.

Moreover, the quadratic form

$$Q := \sum_{j=1}^{\infty} \lambda_j Z_j$$

Satisfies $E[Q] = E[|X - X'|_p|Y - Y''|_q]$ On the other hand, if X and Y are not independent, then

$$n\hat{V}^2(X,Y) \xrightarrow{n\to\infty} \infty \tag{4.13}$$

The equations (4.12) and (4.13) imply a test for independence based on the statistic nV^2.
While Székely et al. (2007) derive an asymptotic test with asymptotic significance level at most α they point out that this significance level can be quite conservative for many distributions.

4.1.4 Generalisations and Modifications of the Distance Covariance

The α distance covariance function, $V^{(\alpha)}(X,Y)$ is a generalisation of the distance covariance function (4.1) for a constant α that lies in the interval [0,2], when $\omega(t,s)$ is used.

$$V^{2(\alpha)}(X,Y) = \frac{1}{C_{(p,\alpha)}C_{(q,\alpha)}} \int_{R^{p+q}} \frac{|\phi_{(X,Y)}(t,s) - \phi_X(t)\phi_Y(s)|^2}{|t|_p^{\alpha+p}|s|_q^{\alpha+q}}\, dt\, ds$$

Where $C(p,\alpha)$ and $C(q,\alpha)$ are given by

$$C(d,\alpha) = \frac{2\pi^{d/2}\Gamma((1-\alpha/2))}{\alpha 2^{\alpha}\Gamma((d+\alpha)/2)}$$

The estimator of $V^{2(\alpha)}(X,Y)$ is computed by defining a distance matrix with elements $a_{ij} = |X_i - X_j|_p^{\alpha}$ and $b_{ij} = |Y_i - Y_j|_q^{\alpha}$

$V^{2(\alpha)}(X,Y)$ Extends the theory of distance covariance because $V^2(X,Y)$ is obtained a special case when $\alpha = 1$.

4.2 The Distance Correlation Coefficient in High Dimensions

As stated in asymptotic Tests,distance covariance provides a test for independence for random X,Y of arbitrary dimension p and q , respectively. Distance correlation is often used when needed as measure for independence than applying a formal test.While this appears to work well for univariate random variables, some authors have reported that the direct interpretation of distance correlation is questionable in high dimensions. [**Dueck et al.(2014)**] showed that for fixed q and standard multivariate normal random vectors $(X, Y)\epsilon R^{p+q}$.

$$\lim_{p\to\infty} R(X, Y) = 0 \tag{4.14}$$

irrespective of the dependence structure between X and Y On the other hand, Székely et al.2013) proved that for fixed n,

$$\lim_{p,q\to\infty} \widehat{R}(X, Y) = 1 \tag{4.15}$$

Provided that the coordinates of X and Y are i.i.d. and the second moments of X and Y exist. Comparison of (4.14) and (4.15) reveals that there are actually two different effects, which complicate the interpretation of distance correlation in high dimension.
Equation (4.14) shows that the population distance correlation is close to 0 when comparing a low-dimensional and a high-dimensional vector.
Equation (4.15) shows that the sample distance correlation is close to 1 when comparing two high-dimensional vectors. These two problems may occur simultaneously, when comparing two high-dimensional vectors with substantially distinct dimensions. In particular,(4.14) shows that we can find sequences $(p_k)_{k\epsilon N}$, $(q_k)_{k\epsilon N}$ such that $p_k \to \infty$,$q_k \to \infty$ and

$$\lim_{k\to\infty} R(X, Y) = 0$$

$$\lim_{k\to\infty} \widehat{R}(X, Y) = 1$$

Where for any k, X and Y are p_k and q_k-dimensional standard normal vectors and n is fixed. When $p = q$, the population distance correlation $R(X, Y)$ can attain both the values 0 (X and Y independent) and 1 ($Y = \alpha + \beta CX$ with $= \alpha\epsilon R^p, b\epsilon R$,
C orthogonal)

Equation (4.15) we can apply an unbiased estimator for the distance covariance as described in Székely et al. (2013). This has the further advantage that transformation of this estimator converges to a Student-t distribution with $n(n-3)/2$ degrees of freedom, which enables a distance correlation $t-test$ for independence in high dimensions. When p and q differ (but are both high-dimensional), the t-test may still be applied, however, a direct interpretability is critical (recall (4.14)).

4.3 Implimetation of distance covariance in recommendation systems

Distance covariance is recently introduced dependency measures between random vectors , the implimetation of distance covariance in user-based neighborhood or item-based neighborhood recommendation systems . where we had used to compute the estimators of the squared distance covariance $\hat{V}^2(i,j)$ in item-based and $\hat{V}^2(u,v)$ in user-based .

4.3.1 Distance covariance in Item-Based Neighborhood

The empirical distance covariance for Item-based neighborhood $\hat{V}^2(i,j)$ with a simple form. Suppose that we have n observations of r_i denotes the rating vecteur of item i,r_j denotes the rating vecteur of item j :

$$\hat{V}^2(r_i,r_j) = \frac{1}{n^2}\sum_{u,v=1}^{n} a_{uv}b_{uv} + \frac{1}{n^4}\sum_{u,v=1}^{n} a_{uv}\sum_{u,v=1}^{n} b_{uv} - \frac{2}{n^3}\sum_{u,v,k=1}^{n} a_{uv}b_{uk}$$

(r_{ui}, r_{uj}) , $i = 1, ..., n$ and $j = 1, ..., n$ is a random sample from the joint distribution of the random vectors r_i and r_j. Based on this sample, consider the $n \times m$ Euclidean pairwise distance matrices with elements

$$(a_{uv}) = (|r_{ui} - r_{vi}|_p)$$

and

$$(b_{uv}) = (|r_{uj} - r_{vj}|_q)$$

These matrices are double centred so that their row and means are equal to zero. In other words, let

$$A_{uv} = a_{uv} - \bar{a_{u.}} - \bar{a_{.v}} + \bar{a_{..}}$$
$$B_{uv} = b_{uv} - \bar{b_{u.}} - \bar{b_{.v}} + \bar{b_{..}}$$

where $\bar{a_{u.}} = (\sum_{v=1}^{n} a_{uv})/n, \bar{a_{.v}} = (\sum_{u=1}^{n} a_{uv})/n, \bar{a_{..}} = (\sum_{u,v=1}^{n} a_{uv})/n^2$

Similarly, we define the quantities $b_{u.}$, $b_{.v}$ and $b_{..}$. The sample distance covariance is defined by the square root of the statistic

$$\hat{V}^2(r_i,r_j) = \frac{1}{n^2}\sum_{u,v=1}^{n} A_{uv}B_{uv}$$

$$\hat{V}^2(r_i,r_j) = \frac{1}{n^2}\sum_{u,v=1}^{n} a_{uv}b_{uv} + \frac{1}{n^4}\sum_{u,v=1}^{n} a_{uv}\sum_{u,v=1}^{n} b_{uv} - \frac{2}{n^3}\sum_{u,v,k=1}^{n} a_{uv}b_{uk}$$

4.3.2 Distance covariance in User-Based Neighborhood

The empirical distance covariance for User-based neighborhood $\hat{V}^2(u,v)$ with a simple form. Suppose that we have m observations of r_u denotes the rating vecteur of user u,r_v denotes the rating vecteur of user v :

$$\hat{V}^2(r_u, r_v) = \frac{1}{m^2}\sum_{i,j=1}^{m} a_{ij}b_{ij} + \frac{1}{m^4}\sum_{i,j=1}^{n} a_{ij}\sum_{i,j=1}^{m} b_{ij} - \frac{2}{m^3}\sum_{i,j,k=1}^{m} a_{ij}b_{ik}$$

(r_{ui}, r_{vi}) , $u = 1, ..., m$ and $v = 1, ..., m$ is a random sample from the joint distribution of the random vectors r_u and r_v. Based on this sample, consider the $n \times m$ Euclidean pairwise distance matrices with elements

$$(a_{ij}) = (|r_{ui} - r_{uj}|_p)$$

and

$$(b_{ij}) = (|r_{vi} - r_{vj}|_q)$$

These matrices are double centred so that their row and means are equal to zero. In other words, let

$$A_{ij} = a_{ij} - \bar{a_{i.}} - \bar{a_{.j}} + \bar{a_{..}}$$
$$B_{ij} = b_{ij} - \bar{b_{i.}} - \bar{b_{.j}} + \bar{b_{..}}$$

where $\bar{a_{i.}} = (\sum_{j=1}^{m} a_{ij})/m, \bar{a_{.j}} = (\sum_{i=1}^{m} a_{ij})/m, \bar{a_{..}} = (\sum_{i,j=1}^{m} a_{ij})/m^2$

Similarly, we define the quantities $b_{i.}$, $b_{.j}$ and $b_{..}$. The sample distance covariance is defined by the square root of the statistic

$$\hat{V}^2(r_u, r_v) = \frac{1}{m^2}\sum_{i,j=1}^{n} A_{ij}B_{ij}$$

$$\hat{V}^2(r_u, r_v) = \frac{1}{m^2}\sum_{i,j=1}^{m} a_{ij}b_{ij} + \frac{1}{m^4}\sum_{i,j=1}^{m} a_{ij}\sum_{i,j=1}^{n} b_{ij} - \frac{2}{m^3}\sum_{i,j,k=1}^{n} a_{ij}b_{ik}$$

Chapter 5

Implementation of Restaurant Recommendation Systems

5.1 Introduction

Recommendation systems help users deal with the overload data by recommending items based on thier preferences. Currently, recommendation systems play an important role in most commercial applications. These systems help increase both users' satisfaction and content providers' revenue by capturing users' personalized preferences. The central concept of recommendation systems is personalized prediction, to which the most common approach is Neighborhood collaborative filtering, which employs users' interactions and interests to define their preferences among a considerable number of items.

Nowadays, eating out at restaurants has becomes one of the needs of today's society due to the a lifestyle .The large number of restaurant choices and also lack of information about the restaurant become an obstacle to people's needs in choosing a restaurant. A restaurant is a business that sells various menus of food and drinks to its users.The bigger the business, the more menus are offered. Consequently, choosing a good appropriate restaurant to order from it can be challenging and time consuming for the users since the choice of the available restaurants provided is overwhelming.However, today, users want to get a restaurant with a good reputation and fit their tastes, so that restaurant ratings from other users are required in the restaurant recommendation process.

Recommendation systems help users to tackle the problem of having to find restaurants suit their preference by generating a set of a personalized list of recommendations that might be of interest them by learning through their previous rating. In this case, A restaurant recommendation system is becoming a necessity following the trend of eating out at the restaurants. The task of such a system is to generate a **top-N** list of restaurants that may be of interest to a user, in which the user previous rating behavior, we propose an YASSIR EXPRESS android and web application where we applied our recommendation system models.

We implement widely approach used in the recommendation systems user-Neighborhood collaborative filtering and the Item-Neighborhood collaborative filtering approachs that employs the users or items similarities using diffrent metrics to generate the list of recommendations to a target user. we have implemented Also popularity recommendation systems and location recommendation system.

5.2 YASSIR EXPRESS

5.2.1 The Application

Released on 2018, YASSIR EXPRESS it's application developed by the company **YASSIR**, bringing together meal delivery services from restaurants partner, Yassir dark stores, and other services such as dry cleaning.Available on APP STORE about the smartphones under IOS, and Google Play for Smartphones under Android.
YASSIR EXPRESS offers a service by which E-consumers will be put in touch with E-Suppliers for the purchase of products/services/Meals remotely presented on the application, and benefit from delivery to the destination of their choice.In this context, it is recalled that Yassir Express acts as an intermediary, its role is limited to hosting offers of products/services/meals from E-suppliers and putting them in contact with E-consumers. The opportunities behind a recommendation system for an YASSIR EXPRESS are several. Indeed it would allow a better user experience and They provide an effective way of driving traffic ,Increase Average Order Value whitch increase the company revenue.

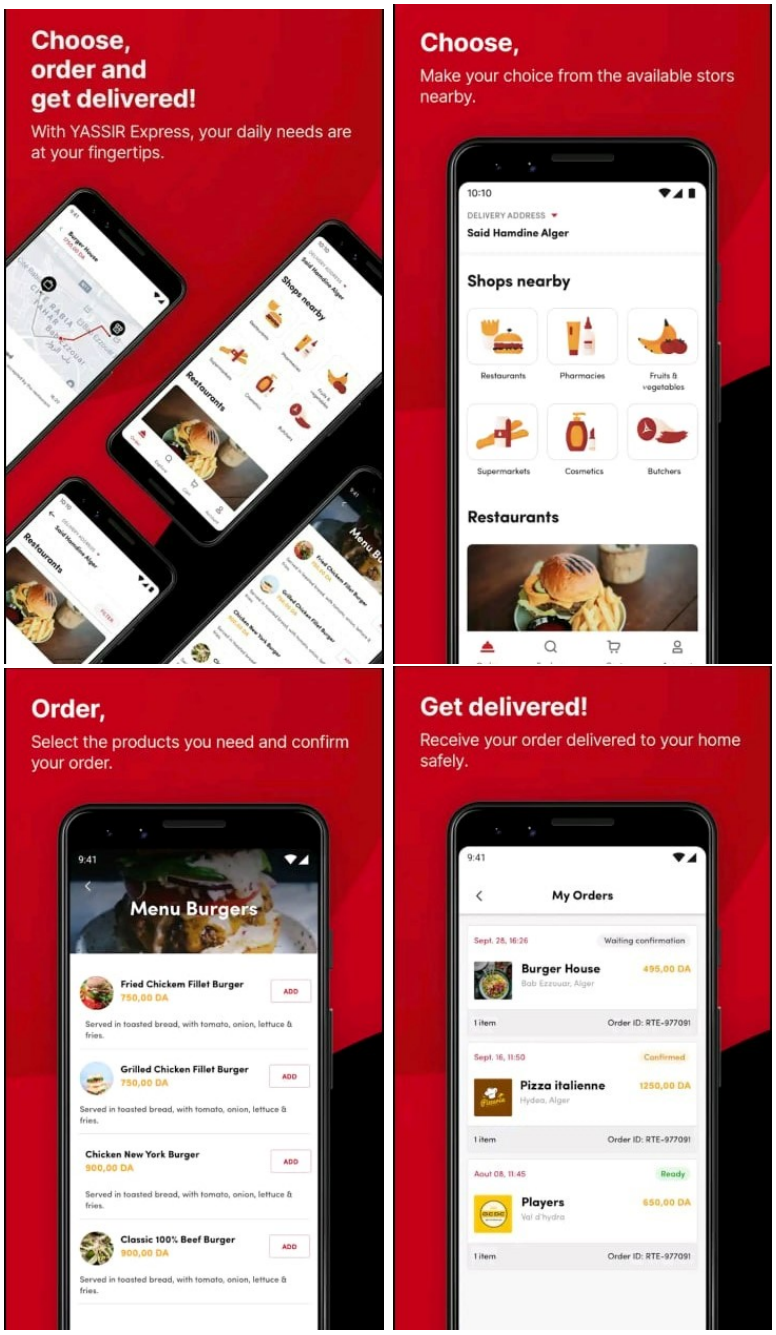

Figure 5.1: YASSIR EXPRESS Application

5.2.2 Flow

In this section we will see more in detail how the app works, trying to follow the user from switching on the app to choosing a restaurant.

The user journey starts at the home page where the user add his/her location position , Then the user can choose between a chose or selected set of restaurants,based on the restaurant near to his location,next the user can pick the avalibe dish the from the he have select ,As we can see on the next figure 5.2.2

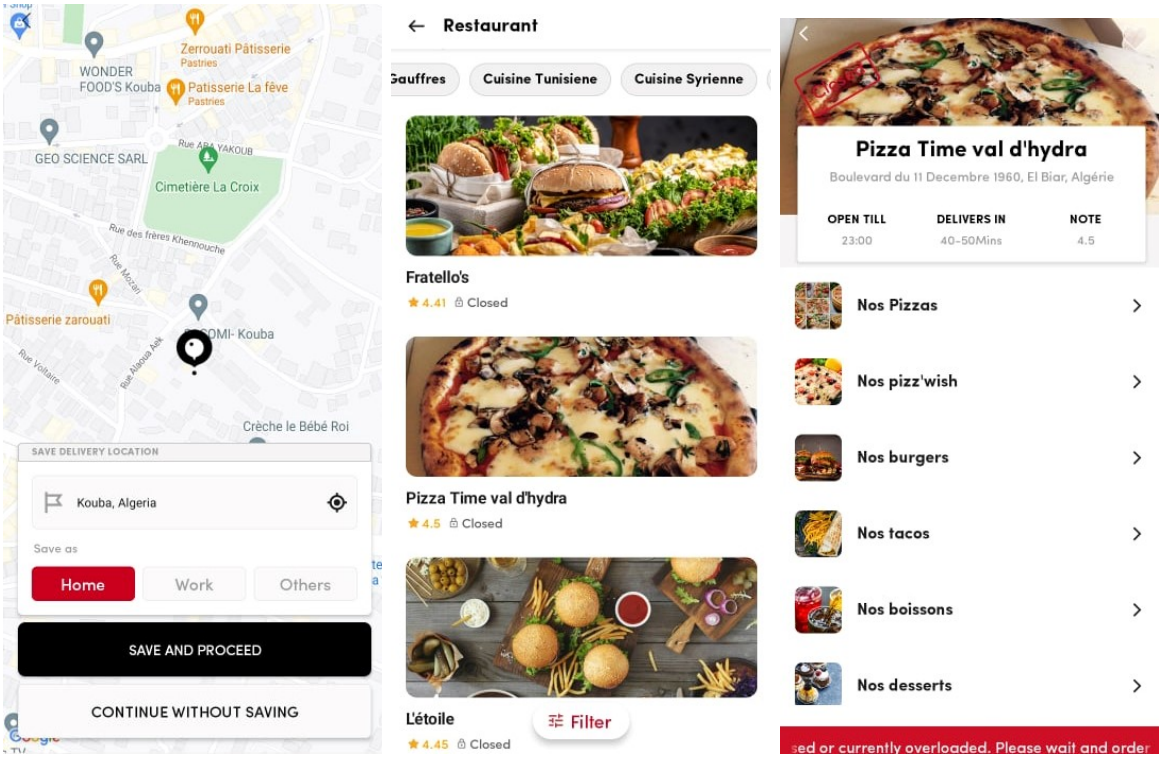

Figure 5.2: YASSIR EXPRESS User Experience

5.2.3 Frameworks and libraries variety

Python Python is a high-level, interpreted, general-purpose programming language. Its design philosophy emphasizes code readability with the use of significant indentation. Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly procedural), object-oriented and functional programming. It is often described as a "batteries included" language due to its comprehensive standard library Python is a programming language that is preferred for programming due to its vast features, applicability, and simplicity. The Python is the best fits machine learning due to its independent platform and its popularity in the programming community.

Google Colab Google Colaboratory, or "Google Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code in other words,Colab is web IDE for python through the browser and is especially well suited to machine learning, data analysis and education. Colab is a hosted Jupyter notebook service that requires no setup to use, while providing access free of charge to computing resources including GPUs.

Python Libraries Python Libraries are vital in the preparation of a suitable programming environment, it's offer a reliable environment that reduces software development time significantly. A library basically includes a prewritten code that developers can use to speed up coding when working on complex projects. Here is the main the Python Libraries that we have used on our project.

- **pandas** is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables.
- **NumPy** is a package that defines a multi-dimensional array object and associated fast math functions that operate on it. It also provides simple routines for linear algebra and fft and sophisticated random-number generation.
- **Seaborn** is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
- **Scikit-learn** (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language.It features various classification, regression and clustering algorithms including support-vector machines, random forests, gradient boosting, k-means .
- **Plotly** is Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms...

- **Surprise** is a Python scikit for building and analyzing recommender systems .
- **SciPy** in Python is an open-source library used for solving mathematical, scientific, engineering, and technical problems. It allows users to manipulate the data and visualize the data using a wide range of high-level Python commands.
- **Matplotlib** is a comprehensive library for creating static, animated, and interactive visualizations in Python.

5.2.4 Dataset

The dataset used for training the recommendation system was the YASSIR EXPRESS , it composed of one tables with all the information about users and restaurants, collected by the webapplication.

YASSIE EXPRESS comprises **104358** users with information on **1069** restaurants(items) each type of data in the dataset was available in the form of two csv files,then we did the merge. .
For the raison of anonymity, we are obligated not to disclose the dataset's real numbers, names and times, and values.

Dataset	*Users*	*Items*
Yassir Express	*104358*	*1069*

Table 5.1: **Table summarizes the YASSIR EXPRESS dataset**

A datasets contains information about the users and the restaurants where all the main information about restaurants are collected, for instance:

- **order id**: the ID of the order.
- **user id**: the ID of the user.
- **restaurant**: the ID of the restaurant.
- **request date:** the request time of the order.
- **delivery date:** the delivery time of the order.
- **pick up date:** the pick up date date of the order.
- **restaurant accepted date:** the time that the restaurant accepted the order.
- **restaurant address lat:** the restaurant latitude address .
- **restaurant address lng:** the restaurant longitude address.
- **commune res:** the restaurant commune address.
- **wilaya res:** the restaurant wilaya address.

5.2.5 Experimental Setup

We followed the process described in Figure 1 with the aim to apply an experimental comparative evaluation result between the similarity metrics using different evaluation metrics. In Step 1, we selected the Yassir Express dataset s. In In step 2, we did the loading of the data, the data Preparation then exploration. In Step 3, we implemented recommendation models. Finally, in Step 4, we evaluated the recommendation models using RMSE, MSE, and MAE metrics.

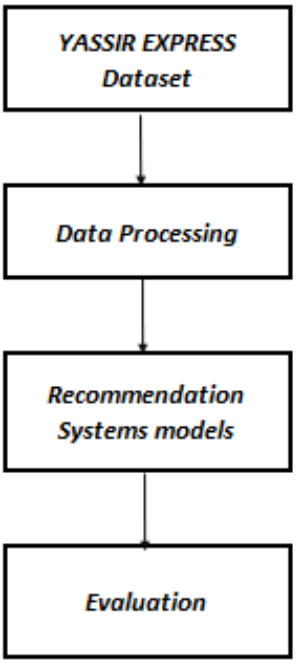

Figure 5.3: Steps performed to compare the recommendations systems

5.3 Data Processing

5.3.1 Data Loading

The data loading from two CSV files,the resulting format is a Pandas dataframe structure, which can be considered a table representation of the data.

5.3.2 Data Preparation

This module's functions provide easy manipulation and filtering of the data build on the Python library and its dataframe structure. First, we removed the missing values (NAN) ,then we did the transformation of the different data structures into appropriate data typesas time and the string datatype. For instance, convert user and item information into numerical types and dates represented by strings into a corresponding date.

5.3.3 Data Exploration

To explore our dataset, we will see how many unique cities and states are in the dataset, and then we will display most of the ten cities and states with the largest number of restaurants.

```
data['wilaya_res'].value_counts().sort_values(ascending=False)
data['commune_res'].value_counts().sort_values(ascending=False)
```

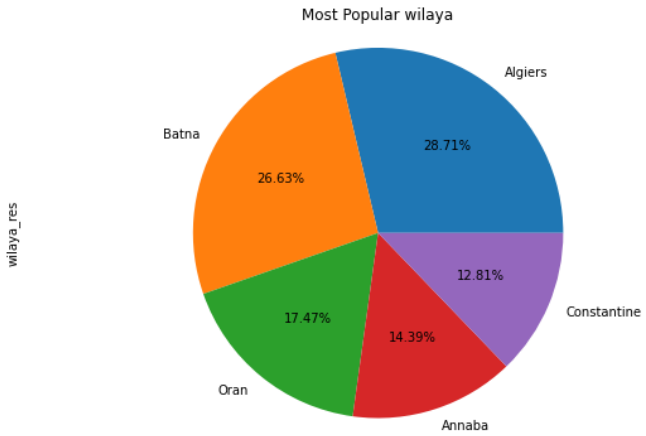

Figure 5.4: The states with the largest number of restaurants

As you can see that Algiers is the state that have the largest number of restaurants.

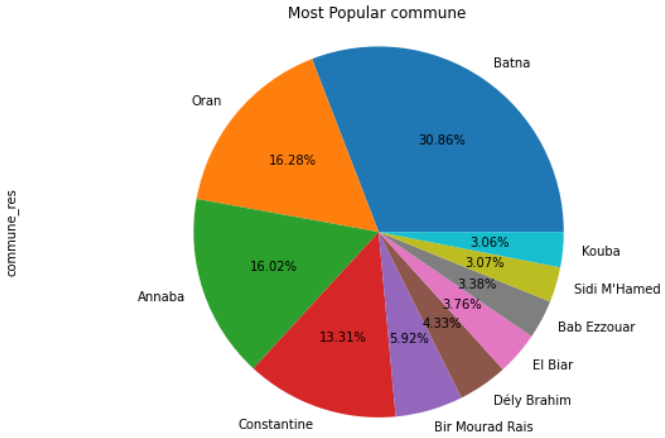

Figure 5.5: The commune with the largest number of restaurants

As you can see that Batna is the commune that have the largest number of restaurants.

The User Rating

The YASSIR EXPRESS dataset doesn't contain any user rating ,this problem occurs when the users are lazy or don't rate items. due to this situation we defined an implicit rating ,as we have previously mentioned the the implicit rating can be gathered using the user's interaction to deduce the user's preferences.

The implicit rating function that outputs is the rating r_{ui} [1-5] of the user u to item i .

We have considered duration per minute between the delivery time and request time as the implicit rating that we will work with. In contrast, the delivery duration is the most crucial feature that the users focus on in the food delivery services. Moreover, the users do not order from a restaurant that takes a long time to deliver.
Furthermore, there are other features like the taste and flavor of the delivered dishes, hygiene, price, etc. as our dataset doesn't contain any of those information's we can't add this features on our implicit rating computation.

```
data.delivery_date = pd.to_datetime(data.delivery_date)

data.request_date = pd.to_datetime(data.request_date)

data["diff"] = (data["delivery_date"] - data["request_date"])
.astype('timedelta64[m]')
```

We applied the k-means clustering algorithm to cluster the delivery duration by selecting the k=5 values the number of clusters, then we have created the rating scalar by assing for each cluster his own rating value.

```
X1 = data[['diff']].iloc[: , :].values
kmeans= KMeans(n_clusters = 5, random_state = 42)
# Compute k-means clustering
kmeans.fit(X1)
# Compute cluster centers and predict cluster index for each sample.
pred = kmeans.predict(X1)
pred
data['Cluster'] = pd.DataFrame(pred, columns=['cluster'] )

# Create a Rating scalar
data.loc[data['Cluster'] == 0 , 'rate_service'] = 2
data.loc[data['Cluster'] == 1 , 'rate_service'] = 4
data.loc[data['Cluster'] == 2 , 'rate_service'] = 1
data.loc[data['Cluster'] == 3 , 'rate_service'] = 3
data.loc[data['Cluster'] == 4 , 'rate_service'] = 5
```

We plot the distribution of the ratings (that we have created). where the total of each rating number is calculated. This plot shows that a rating range of 2 to 4 is available for this dataset and most restaurants have a rating of 1.

```
fig, ax = plt.subplots(figsize=(12,10))
sns.countplot(data['rate_service'], ax=ax)
plt.title('Rating distribuation')
plt.show()
```

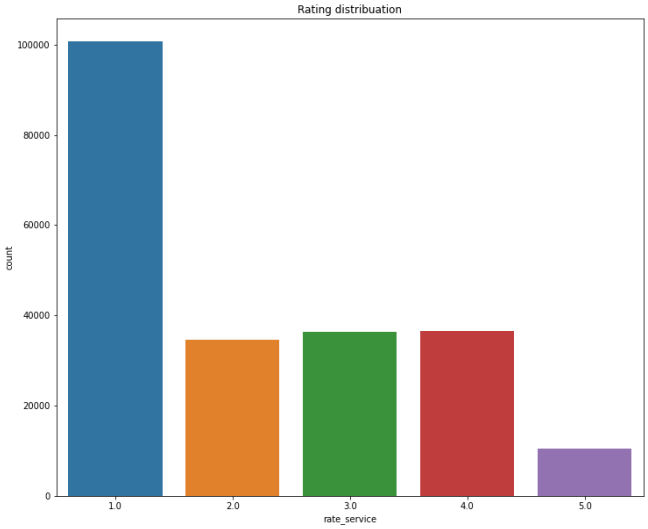

Figure 5.6: The Rating Distribuation

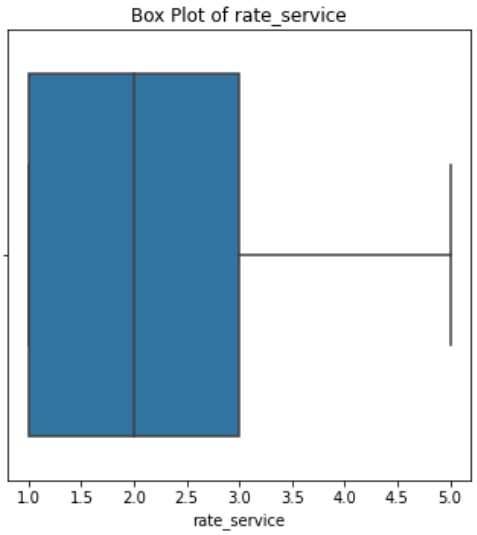

Figure 5.7: Box Plot Of The Rating

As we can visualize from the box plot of the rating distribuation isright-skewed where the 25% of the rating are between 2 and 1 while 75% of our rating are from 2

to 5 and the minimum is 1 , the maximum 5,the median is the average value from a set of rating shown by the line that divides the box into two parts it's equal to 2.

5.4 The Recommendations Systems Models

Recommendation systems are one of the most commonly used practical systems in data science. In this section, we will focus on collaborative filtering, where the focus is on similarities between users.

Depending on In this section we will describe in detail the recommendeations that we built. In the first section we will describe the preprocessing and the decisions we took as to build all the models. Then we will describe base models.

we use user-based collaborative filtering method. If the user wants to find a restaurant recommended by another user, then the system will searching similarity of preferences the target user and all existing users by calculating the similarity between users and the similarity of the user attributes. Then the system search the neighbors who have biggest similarity with the target user, so that restaurants that have been given a rating by neighbors will be recommended to target users who have not rated the restaurant.

5.4.1 Evaluation Metrics

Before applying a recommendation systems in real life applications, it is important to measure the effectiveness of a RSs.

To evaluate the performance of the RS, we intend to utilize each time a different similarity measure. Therefore, we evaluate the performance of the RS using 3 well-known evaluation metrics which are widely used for evaluating regression models :

- ***Root Mean Square Error (RMSE)***

  it evaluates the difference between the ratings predicted $\hat{r_{ui}}$ by the RS and the ratings given by the user r_{ui} .

$$RMSE = \sqrt{\frac{\sum_{(u,i,r)\epsilon R}(\hat{r_{ui}} - r_{ui})^2}{|R|}}$$

- ***Mean Absolute Error (MAE)***:

  it evaluates the difference between the ratings predicted by the RS and the ratings given by the users. It returns a positive value.

$$MAE = \frac{\sum_{(u,i,r)\epsilon R}|\hat{r_{ui}} - r_{ui}|}{|R|}$$

  Where R denotes the amount of user u rated items,normally a Test Set,r_{ui} determines the actual rating that user u rates item i and $\hat{r_{ui}}$ denotes the predicted rating of item i for user u . The lower the MAE,the more accuracy the recommendation system predicts rating.

- ***Mean squared error (MSE)***

  The MSE is used to evaluate recommendation systems by computing the average value of all absolute value differences between the predicted ratings $\hat{r_{ui}}$ and the user's true ratings r_{ui}. The MSE can be computed using the following equation

$$MSE = \frac{\sum_{(u,i,r)\epsilon R}(\hat{r_{ui}} - r_{ui})^2}{|R|}$$

5.4.2 Popularity Based Recommendation Model

Popularity based recommendation system uses the items that are in trend right now ,It is a very fundamental type of recommendation systems which gives recommendations based on the popularity of the item ,As mentioned in section 2.8.1. For example, the most popular item will be recommended first.It ranks items based on its popularity i.e. the rating count. If a item is highly rated then it is most likely to be ranked higher and hence will be recommended. As it is based on the items popularity,that is why it's can't be personalized and hence same set of items will be recommended for all the users.The implemented Popularity recommendation systems we selected 70% of the dataset as the training set and the remaining 30% as the testing test.

The Recommendation Based on the Popularity Model

```
#Use the popularity based recommendation system model
#To make recommendations to user_id=265
popularity_rss= pr.recommend(265)
popularity_rss
```

	userid	itemid	rating_count	Rank
446	265	452	12369	1.0
367	265	373	6687	2.0
442	265	448	4183	3.0
293	265	299	4087	4.0
146	265	149	3215	5.0
355	265	361	3076	6.0
643	265	652	3065	7.0
632	265	641	2816	8.0
478	265	484	2724	9.0
257	265	262	2177	10.0

Figure 5.8: The Ten Recommendation Based on the Popularity Model

This table represents the ten recommended items based on the popularity model for the userid 265. Our model depends on the Rank variable, which ranks the items based on the highest rating count, the rating count variable counts the number of users that have order from the itemid (the restaurant) .

In this section, we will plot the evaluation metrics results. The evaluation metrics are **RMSE,MSE, and MAE ,Recall ,Precision**.

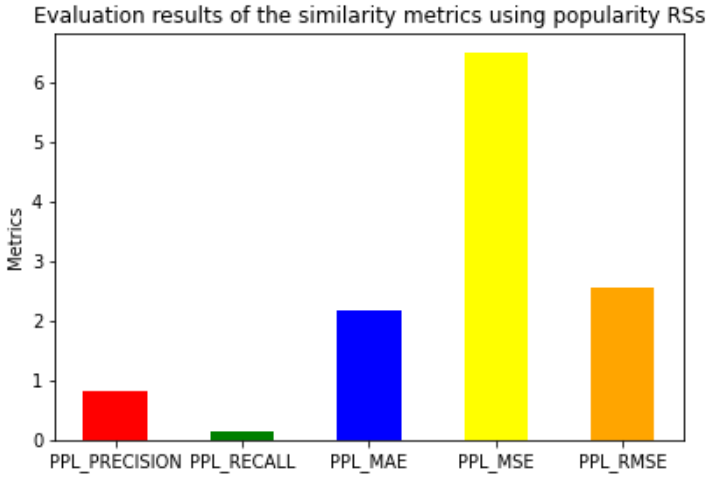

Figure 5.9: The Ten Recommendation Based on the Popularity Model

Model	RMSE	MSE	MAE	Recall	Precision
Popularity RSs	**2.54954**	**6.50016**	***2.19050***	***0.16666***	***0.83334***

Table 5.2: **The Evaluation Metrics Results on the Popularity RSs**

5.5 Neighborhood Collaborative Filtering

Today, recommendation systems play a vital role in the acceleration of searches by internet users to find what they are interested in. Among the strategies proposed for recommendation systems, Neighborhood collaborative filtering (also known as memory-based systems) has received due attention regarding its simplicity and efficiency. Neighbourhood-based collaborative filtering models directly exploit the feedback from the users.

We can distinguish either user-based or item-based approaches within Neighbourhood-based, the critical factor for the success of this strategy returns to the similarity calculation methods that affect the accuracy of its recommendations In this section, we will implement the user-Neighborhood and the item-Neighborhood collaborative filtering. We carried out an experimental comparative evaluation result between the similarity metrics using different evaluation metrics, MSE, RMSE, and MAE.

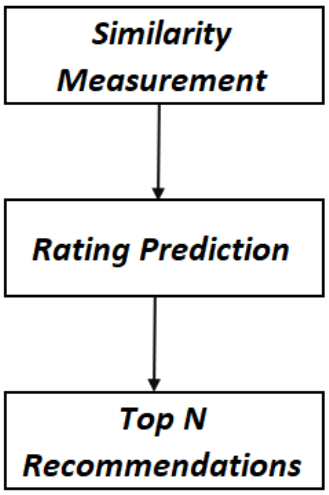

Figure 5.10: Flowchart Of The Neighborhood Collaborative Filtering

5.5.1 User Neighborhood Collaborative Filtering

User Neighborhood CF is based on the main idea that users who have an interest in same items and similar ratings will thus have similar preferences,while the recommendations based on using the feedback from similar users (known as neighbours) , where they give a similar rating behavior, the recommendation system is able to predict if a user might be interested in an unseen item. In the user-based recommendation we need to find **k** most similar users and make a prediction of a target user ,k-NN computes a specific neighbourhood for each user consisting of the top k users with higher similarity to the target user, the the Top-N recommendation.

The process of a typical User Neighborhood is generally divided into three steps:

1. **Similarity measurement** (between users) The recommendation system computes the similarities between the active user and other users who have rated the same items. Neighborhood Similar users are regrouped in a subset namely the « neighborhood » of the active user. .

2. **Rating prediction** (using ratings of similar users) User-based prediction of a rating r_{ui} is obtained as:

$$\widehat{r_{ui}} = \bar{r_u} + \frac{\sum_{v \epsilon N_i(u)} s_{uv}(r_{vi} - \bar{r_v})}{\sum_{v \epsilon N_i(u)} |s_{uv}|}$$

3. **The Top-N recommendations** are obtained by choosing the N items which provide most satisfaction to the user according to prediction by decreasing order of predicted rating.3.9.1

We applied an experimental comparative evaluation result between the similar items as the critical step in the proposed user-based recommendation is to determine the similarity between the users, In implementing this model, applied different similarity metrics as the Cosine, Euclidean Distance, Manhattan Distance using different evaluation metrics, MSE, RMSE, and MAE.

5.5.2 Item Neighborhood Collaborative Filtering

Item k-nearest neighborhood collaborative filtering method that predicts the user preference based on the similarity with the k nearest items. while the item-based to determine the similarities among the various items and then used them to identify the set of items to be recommended.first the method used to compute the similarity between the items, and then combine these similarities in order to compute the similarity between a set of items and a candidate recommender item.

The process of a typical Item Neighborhood is generally divided into three steps as the user Neighborhood :

1. **Similarity computation** (among the items). The recommendation system computes the similarities between the items , Neighborhood Similar items are regrouped in a subset namely the « neighborhood » of items. .

2. **Rating prediction** (using ratings of similar items). item-based prediction of a rating r_{ui} :

$$r_{ui} = \bar{r_i} + \frac{\sum_{j \epsilon N_u(i)} s_{ij}(r_{uj} - \bar{r_j})}{\sum_{j \epsilon N_u(i)} |s_{ij}|}$$

3. **Top-N recommendation**. Are obtained by choosing the N items which provide most satisfaction to the user according to prediction by decreasing order of predicted rating. 3.9.1

We applied an experimental comparative evaluation result between the similar items as the critical step in the proposed item-based recommendation is to determine the similarity between the items. In implementing this model, applied different similarity metrics as the Cosine, Adjusted Cosine, Euclidean, Manhattan Distance, and Distance Covariance Distance.

5.5.3 Similarity Metrics

In this section, the similarity metrics used in neighborhood-based collaborative filtering are described.

Cosine This metric first transfers inputs to the vector space and then utilizes the angle between two rate vectors as the degree of the similarity of two users,or items.

Adjusted Cosine This metric is extensions of the Cosine metric which are defined similar to Pearson correlation. The Adjusted cosine measure calculates the correlation value between two users.

Euclidean Distance The euclidean distance compute the correlation between two users (or two items) is the absolute value of the numerical difference of their coordinates.

Manhattan Distance The Manhattan Distance it's the one norm of the distance the measure the similarity between two users (or two items)

Covariance Distance Since we want to have a similarity measure, we take its complement to one. Hence, the more ratings are close, the more the distance is short, the more the similarity tends to 1. And therefore the more users or items are considered to be alike , Covariance Distance It's new similarity metrics that have been used for the first time in the Recommendation systems Domaine. Distance covariance is recently introduced. dependency measures between random vectors, where we will use the empirical distance covariance to compute the similarity between the two users(or two items)

Measure	User Similarity Measure	Item similarity Measure
Cosine	$cosine(u,v) = \frac{\sum_{i \epsilon I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{u \epsilon I_u} r_{ui}^2} \sqrt{\sum_{u \epsilon I_v} r_{vi}^2}}$	$cosine(i,j) = \frac{\sum_{u \epsilon U_{ij}} r_{ui} r_{uj}}{\sqrt{\sum_{u \epsilon U_i} r_{ui}^2} \sqrt{\sum_{u \epsilon U_j} r_{uj}^2}}$
Adjusted cosine	$ACosine(u,v) = \frac{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_i})(r_{vi} - \bar{r_i})}{\sqrt{\sum_{i \epsilon I_{uv}} (r_{ui} - \bar{r_i})^2} \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - \bar{r_i})^2}}$	$ACosine(i,j) = \frac{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_u})(r_{uj} - \bar{r_u})}{\sqrt{\sum_{u \epsilon U_{ij}} (r_{ui} - \bar{r_u})^2} \sqrt{\sum_{u \epsilon U_{ij}} (r_{uj} - \bar{r_u})^2}}$
Covariance Distance	$\hat{V}^2(u,v) = \frac{1}{m^2} \sum_{i,j=1}^{m} a_{ij} b_{ij} + \frac{1}{m^4} \sum_{i,j=1}^{n} a_{ij} \sum_{i,j=1}^{m} b_{ij} - \frac{2}{m^3} \sum_{i,j,k=1}^{m} a_{ij} b_{ik}$	$\hat{V}^2(i,j) = \frac{1}{n^2} \sum_{u,v=1}^{n} a_{uv} b_{uv} + \frac{1}{n^4} \sum_{u,v=1}^{n} a_{uv} \sum_{u,v=1}^{n} b_{uv} - \frac{2}{n^3} \sum_{u,v,k=1}^{n} a_{uv} b_{uk}$
Euclidean distance	$d(u,v) = \sqrt{\sum_{i \epsilon I_{uv}} (r_{vi} - r_{ui})^2}$	$d(i,j) = \sqrt{\sum_{u \epsilon U_{IJ}} (r_{uj} - r_{ui})^2}$
Manhattan distance	$d_1(u,v) = \sum_{i \epsilon I_{uv}} (\lvert r_{vi} - r_{ui} \rvert)$	$d_1(i,j) = \sum_{u \epsilon U_{ij}} (\lvert r_{uj} - r_{ui} \rvert)$

Table 5.3: **Table Of The Used Similarities Metrics**

Important: Notice that due to the computational limitations (RAM Saturation problem) of the large number of the users in our dataset, we limited our user neighborhood CF implementation model by applying only the Euclidean distance, Cosine, and Manhattan Distance similarity metrics.

Implementation of the User Neighborhood

- We identify the set G_u of k most similar users. G_u is the group users similar to the active user u. The similarity between two users u and v can be measured by the similarity metrics.
- Find the set C of candidate items, purchased by the group and not purchased by the active user u.
- Candidate items have to be the most frequent items purchased by the group. Aggregate ratings of users in G_u to make predictions for user u on items he has not already purchased.
- The Top-N recommendations are obtained by choosing the N items which provide most satisfaction to the user according to prediction.

Step 1. Identify G_u ***the set of*** k ***users similar to an active user*** u

Defined the function the compute the User-Item matrix R.
Each row of the rating matrix R represents ratings of user u on all items of the database. Missing ratings are replaced with 0, Uth row of the rating matrix R. Ratings of user u on all items in the database,

```
def ratings_matrix(data1):
return csr_matrix(pd.crosstab(data1.userid, data1.itemid, data1.rating,
 aggfunc=sum).fillna(0).values)

R = ratings_matrix(data1)
```

We create a nearest neighbors model with sklearn through the function *createmodel()*. This function creates and fit a nearest neighbors model with user's ratings, *n neighbors* =51 neighbors, define the number of neighbors to return.With $k = 50$ neighbors, $|G_u| = 51$ as G_u contains 50 similar users added to the active user u. That is why *n neighbors*=51.

```
def create_model(rating_matrix, metric):
model = NearestNeighbors(metric=metric, n_neighbors=51, algorithm='brute')
model.fit(rating_matrix)
return model
```

Then we defined *nearestneighbors()*function that returns the knn users for each user, Where the input parameters of the function are :

-*ratingmatrix* : rating matrix of shape (nbusers, nbitems)
-*model* : nearest neighbors model
And the output of the function are :
-*similarities* : Distances of the neighbors from the referenced user.
-*neighbors* : Neighbors of the referenced user in decreasing order of similarities.

```
def nearest_neighbors(rating_matrix, model):
similarities, neighbors = model.kneighbors(rating_matrix)
return similarities[:, 1:], neighbors[:, 1:]
```

Similarities and Neighbors using Cosine Similarity metric

```
model_cos = create_model(rating_matrix=R,metric='cosine')
similarities_cos, neighbors_cos = nearest_neighbors(R, model_cos)

similarities_cos, neighbors_cos
```

```
(array([[0.0513167, 0.0513167, 0.0513167, ..., 0.0513167, 0.0513167,
         0.0513167],
        [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
         0.       ],
        [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
         0.       ],
        ...,
        [1.       , 1.       , 1.       , ..., 1.       , 1.       ,
         1.       ],
        [0.       , 0.       , 0.       , ..., 0.       , 0.       ,
         0.       ],
        [1.       , 1.       , 1.       , ..., 1.       , 1.       ,
         1.       ]]), array([[33754, 27427, 40713, ..., 46026,  1712,  1064],
        [ 8749, 43319, 33058, ..., 53374, 27442, 39403],
        [56230, 39487, 30542, ..., 48147, 30951, 48002],
        ...,
        [46377, 46376, 46375, ..., 46382, 46392, 46406],
        [69029, 43427, 36259, ..., 54814, 37488, 56931],
        [46377, 46376, 46375, ..., 46382, 46392, 46406]]))
```

Figure 5.11: Similarities and Neighbors using Cosine Similarity metric

Similarities and Neighbors using Euclidean Distance Similarity metric

```
model_euclidean= create_model(rating_matrix=R,metric='euclidean')
similarities_euclidean, neighbors_euclidean = nearest_neighbors(R,model_euclidean)

similarities_euclidean, neighbors_euclidean
```

Similarities and Neighbors using Manhattan Distance Similarity metric

```
model_manhattan= create_model(rating_matrix=R,metric='manhattan')
similarities_manhattan, neighbors_manhattan  = nearest_neighbors(R,model_manhattan)

similarities_manhattan, neighbors_manhattan
```

Step 2. Find candidate items

The set C of candidate items are the most frequent ones purchased by users in G_u for an active user u and not purchased by u.

Function *findcandidateitems()* find items purchased by these similar users as well as their frequency. Note that the frequency of the items in the set C can be computed by just counting the actual occurrence frequency of that items.

1.Guitems : frequent items of G_u in decreasing order of frequency. 2.activeitems : items already purchased by the active user. 3.candidates : frequent items of G_u not purchased by the active user u.

Find candidate items with Cosine similarity metric

```
def find_candidate_items_cos(userid):
"""
Find candidate items for an active user
:param userid : active user
:param neighbors : users similar to the active user
:return candidates : top 50 of candidate items
"""
user_neighbors = neighbors_cos[userid]
activities = data1.loc[data1.userid.isin(user_neighbors)]

# sort items in decreasing order of frequency
frequency = activities.groupby('itemid')['rating'].count().reset_index(name='count'
Gu_items = frequency.itemid
active_items = data1.loc[data1.userid == userid].itemid.to_list()
candidates_cos = np.setdiff1d(Gu_items, active_items, assume_unique=True)[:50]

return candidates_cos
```

We applied the same function to get the ***Find candidate items with Euclidean Distance similarity metric*** and ***Find candidate items with Manhattan Distance similarity metric***

Step 3. Rating prediction

Predict the score of u on a candidate item i

Rating prediction with the cosine similarity metric predict what score userid would have given to itemid. **:param** - userid : user id for which we want to make prediction. - itemid : item id on which we want to make prediction. **:return** - rhat : predicted rating of user userid on item itemid.

```
def predict_cos(userid, itemid):

user_similarities = similarities_cos[userid]
user_neighbors = neighbors_cos[userid]
# get mean rating of user userid
user_mean = mean[userid]
```

```
# find users who rated item 'itemid'
iratings = np_ratings[np_ratings[:, 1].astype('int') == itemid]

# find similar users to 'userid' who rated item 'itemid'
suri = iratings[np.isin(iratings[:, 0], user_neighbors)]

# similar users who rated current item (surci)
normalized_ratings = suri[:,4]
indexes = [np.where(user_neighbors == uid)[0][0] for uid in suri[:, 0].astype('int'
sims = user_similarities[indexes]

num = np.dot(normalized_ratings, sims)
den = np.sum(np.abs(sims))

if num == 0 or den == 0:
return user_mean

r_hat_cos= user_mean + np.dot(normalized_ratings, sims) / np.sum(np.abs(sims))

return r_hat_cos
```

We applied the same function to get the ***Find candidate items with Euclidean Distance similarity metric*** and ***Find candidate items with Manhattan Distance similarity metric***.

Rating prediction with the cosine similarity metric

```
def predict_cos(userid, itemid):
        """
        predict what score userid would have given to itemid.
        :param
        - userid : user id for which we want to make prediction
        - itemid : item id on which we want to make prediction
        :return
        - r_hat : predicted rating of user userid on item itemid
        """
        user_similarities = similarities_cos[userid]
        user_neighbors = neighbors_cos[userid]
        # get mean rating of user userid
        user_mean = mean[userid]

        # find users who rated item 'itemid'
        iratings = np_ratings[np_ratings[:, 1].astype('int') == itemid]

        # find similar users to 'userid' who rated item 'itemid'
        suri = iratings[np.isin(iratings[:, 0], user_neighbors)]

        # similar users who rated current item (surci)
```

```
        normalized_ratings = suri[:,4]
        indexes =
        [np.where(user_neighbors == uid)[0][0] for uid in
        suri[:, 0].astype('int')]
        sims = user_similarities[indexes]

        num = np.dot(normalized_ratings, sims)
        den = np.sum(np.abs(sims))

        if num == 0 or den == 0:
        return user_mean

        r_hat_cos=
        user_mean + np.dot(normalized_ratings, sims) / np.sum(np.abs(sims))

        return r_hat_cos
```

We applied the same function to get the ***Find candidate items with Euclidean Distance similarity metric*** and ***Find candidate items with Manhattan Distance similarity metric***.

Step 4. Top-N recommendation

Function user2userRecommendation() reads predictions for a given user and return the list of items in decreasing order of predicted rating.

```
def user2userRecommendation_cos(userid):
saved_predictions = 'predictions_cos.csv'

predictions = pd.read_csv(saved_predictions, sep=',',
names=['userid', 'itemid', 'predicted_rating'])
predictions = predictions[predictions.userid==userid]
df = predictions.sort_values(by=['predicted_rating'], ascending=False)

df.userid = df.userid.tolist()
df.itemid =df.itemid.tolist()

df = pd.merge(df, data2, on='itemid', how='inner')
df = df.drop_duplicates(subset=['itemid'])
return df
```

We applied the same function to get the **user2userRecommendation with Euclidean Distance similarity metric** and **user2userRecommendation with Manhattan Distance similarity metric**.

Let us make top recommendation for a given user.

```
user2userRecommendation_cos(244)
```

	userid	itemid	predicted_rating
0	244	358	1.439946
2221	244	84	1.267857
2322	244	668	0.625000
2444	244	662	0.125000
2675	244	409	0.101636

Figure 5.12: TOP Recommendation using Cosine similarity metric

```
user2userRecommendation_euclidean(244)
```

	userid	itemid	predicted_rating
0	244	477	2.725000
234	244	206	1.925000
445	244	358	1.553571
2666	244	625	1.375000
2677	244	309	1.325000

Figure 5.13: TOP Recommendation using Euclidean Distance similarity metric

```
user2userRecommendation_manhattan(244)
```

	userid	itemid	predicted_rating
0	244	477	2.725000
234	244	358	1.553571
2455	244	309	1.325000
2559	244	119	0.125000
2612	244	662	0.125000
2843	244	444	0.125000

Figure 5.14: TOP Recommendation using Manhattan Distance similarity metric

Evaluation of user-based recommendation system with the Manhattan distance similarity metric

We split the dataset with a ratio of 75%, 25% into training set and test set ,We use the training set to obtain the training model, and then apply the model to test set to test the performance of our models.

```
# get examples as tuples of userids and itemids and labels from normalize ratings
raw_examples, raw_labels = get_examples(data1, labels_column='rating')

# train test split
(x_train, x_test), (y_train, y_test) = train_test_split(examples=raw_examples,
 labels=raw_labels)

def evaluate_cos(x_test, y_test):

preds = list(predict_cos(u,i) for (u,i) in x_test)

CFU_MAE_cos = np.sum(np.absolute(y_test - np.array(preds))) / x_test.shape[0]
print('>> CFU_MAE_cos: ' + str(CFU_MAE_cos))

CFU_MSE_cos = mean_squared_error(y_test ,np.array(preds))
print('>> CFU_MSE CFU_MSE_cos: ' + str(CFU_MSE_cos))


CFU_RMSE_cos = math.sqrt(CFU_MSE_cos)
print('>> CFU_RMSE_cos' + str(CFU_RMSE_cos))

evaluate_cos.list1=[CFU_MAE_cos,CFU_MSE_cos,CFU_RMSE_cos]

return evaluate_cos.list1

evaluate_cos(x_test, y_test)
```

We applied the same function to evaluate the ***User-based model using Euclidean Distance similarity metric*** and ***User-based model using Manhattan Distance similarity metric***.

	Similarity Measure	RMSE	MAE	MSE
User-Based	Euclidean Distance	*0.90187*	*0.64881*	*0.81338*
	Manhattan Distance	*0.89382*	*0.64353*	*0.79891*
	Cosine	*0.93452*	*0.69218*	*0.87333*

Table 5.4: **Evaluation Results of the Similarity Measures on User-Based**

5.5.4 Results

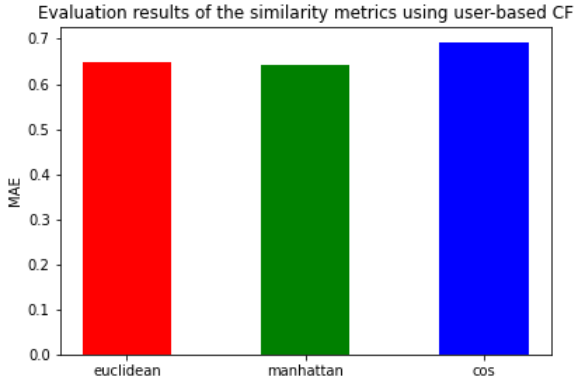

Figure 5.15: MAE Evaluation results of the similarity measures using user-based

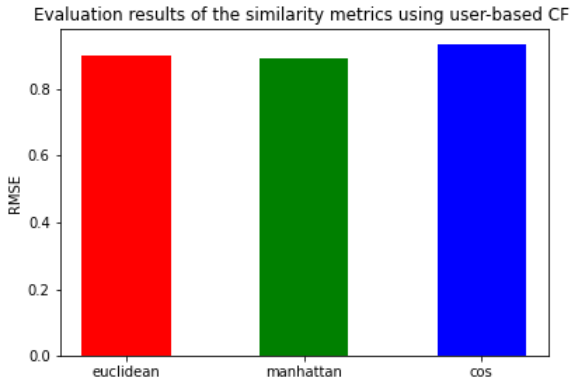

Figure 5.16: RMSE Evaluation results of the similarity measures using user-based

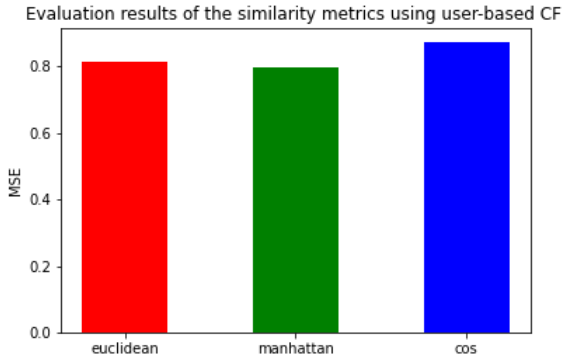

Figure 5.17: MSE Evaluation results of the similarity measures using user-based

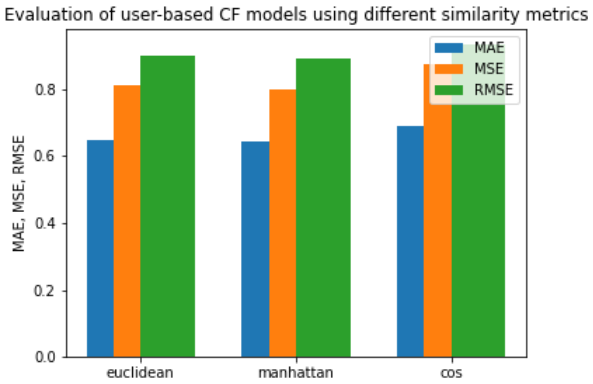

Figure 5.18: Performance Comparison Between The Evaluation results of the similarity measures using user-based

The system was applied to the three recommendation methods, which were evaluated using RMSA and MAE. In this section, we present the obtained findings. We perform user-based Neighborhood. The experimental study was carried out using different similarity measures, and After applying the user-based CF models, we obtained the results shown in 5.4 In the table,and the graphs we present the values reached by each similarity measures in terms of RMSE, MSE, MAE

As shown in Table 5.4, Manhattan distance similarity provides the best result for the 3 evaluation metrics RMSE, MSE, MAE while the lower those evaluation metrics values are the better our models accuracy.For instance, Manhattan distance gets the best values: 0.64353(MAE), 0.79891(MSE), 0.89382(RMSE)

The main conclusion that can be deduced from this part of the experimental study that Manhattan distance measure outscores the other measures if it is applied in the user-based approach ,on YASSIR EXPRESS dataset , while we may obtain different results by using another dataset .

Implementation of the Item Neighborhood

- First identify the k most similar items for each item in the catalogue and record the corresponding similarities. To compute similarity between two items ,we applied the Cosine, Euclidean Distance, Manhattan Distance ,Covariance distance ,Adjusted Cosine.
- Finding candidate items for user u,where the set I_u of items already rated by user u,the we take the union of similar items as C for all items in I_u , then we exclude from the set C all items in I_u, to avoid recommend to a user items he has already purchased.
- Find similarity between each candidate item and the set I_u.
- Rank candidate items according to their similarities to I_u.
- Top-N recommendations for a given user.

Similarities and Neighbors with Adjusted Cosine Similarity metric

```
def cosine(x, y):

return  dot(x, y)/(norm(x)*norm(y))


def adjusted_cosine_it(np_ratings, nb_items):
similarities_adjusted_cosine_it = np.zeros(shape=(nb_items, nb_items))
similarities_adjusted_cosine_it.fill(-1)

def _progress(count):
sys.stdout.write('\rComputing similarities. Progress status : %.1f%%' % (float(coun
sys.stdout.flush()

items = sorted(data1.itemid.unique())
for i in items[:-1]:
for j in items[i+1:]:
scores = np_ratings[(np_ratings[:, 1] == i) | (np_ratings[:, 1] == j), :]
vals, count = np.unique(scores[:,0], return_counts = True) # vals of users and coun
scores = scores[np.isin(scores[:,0], vals[count > 1]),:]

if scores.shape[0] > 2:
x = scores[scores[:, 1].astype('int') == i, 4]  # rating of item i
y = scores[scores[:, 1].astype('int') == j, 4]  #rating of item j

ml = max(len(x),len(y))
x = np.concatenate((x , np.zeros(ml-len(x))))
y = np.concatenate((y , np.zeros(ml-len(y))))
w = dis.cosine(x,y)

similarities_adjusted_cosine_it[i, j] = w
similarities_adjusted_cosine_it[j, i] = w
```

```
_progress(i)
_progress(nb_items)

# get neighbors by their neighbors in decreasing order of similarities
neighbors_adjusted_cosine_it = np.flip(np.argsort(similarities_adjusted_cosine_it),

# sort similarities in decreasing order
similarities_adjusted_cosine_it = np.flip(np.sort(similarities_adjusted_cosine_it),

return similarities_adjusted_cosine_it, neighbors_adjusted_cosine_it
```

now, we can call the adjustedcosine function to compute and save items similarities and neighbors based on the adjusted cosine metric.

```
similarities_adjusted_cosine_it, neighbors_adjusted_cosine_it = adjusted_cosine_it(
```

Similarities and Neighbors with covariance distance Similarity metric

```
def cov_dist(x, y):
        return dcor.distance_covariance(x, y, method='AVL')

def dist_it(np_ratings, nb_items):

similarities_cov_it = np.zeros(shape=(nb_items, nb_items))
similarities_cov_it.fill(-1)

def _progress(count):
sys.stdout.write('\rComputing similarities. Progress status : %.1f%%' % (float(coun
sys.stdout.flush()

items = sorted(data1.itemid.unique())
for i in items[:-1]:
for j in items[i+1:]:
scores = np_ratings[(np_ratings[:, 1] == i) | (np_ratings[:, 1] == j), :]
vals, count = np.unique(scores[:,0], return_counts = True)
scores = scores[np.isin(scores[:,0], vals[count > 1]),:]

if scores.shape[0] > 2:
x = scores[scores[:, 1].astype('int') == i, 2] # i items avec rating dyalo
y = scores[scores[:, 1].astype('int') == j, 2] # j items avec rating dyalo
#x=x[x != 0]
#y=y[y != 0]
ml = max(len(x),len(y))
x = np.concatenate((x , np.zeros(ml-len(x))))
y = np.concatenate((y , np.zeros(ml-len(y))))

w = dcor.distance_covariance(x, y)
```

```
similarities_cov_it[i, j] = w
similarities_cov_it[j, i] = w
_progress(i)
_progress(nb_items)

# get neighbors by their neighbors in decreasing order of similarities
neighbors_cov_it = np.flip(np.argsort(similarities_cov_it), axis=1)

# sort similarities in decreasing order
similarities_cov_it = np.flip(np.sort(similarities_cov_it), axis=1)

return similarities_cov_it , neighbors_cov_it


nb_items = data1.itemid.nunique()
similarities_cov_it , neighbors_cov_it=  dist_it(np_ratings, nb_items=nb_items)
```

We computed the similarities and the neighbors of the Cosine, Euclidean, and the manhattan similarity metrics in the same way we previously calculated them on the user-based.

Finding candidate items for user u using the covariance distance similarity metric

```
def candidate_items_cov_it(userid):
"""
:param userid : user id for which we wish to find candidate items
:return : I_u, candidates
"""

# 1. Finding the set I_u of items already rated by user userid
I_u = np_ratings[np_ratings[:, 0] == userid]
I_u = I_u[:, 1].astype('int')

# 2. Taking the union of similar items for all items in I_u to form the set of cand
c_cov = set()

for iid in I_u:
# add the neighbors of item iid in the set of candidate items
c_cov.update(neighbors_cov_it[iid])

c_cov = list(c_cov)
# 3. exclude from the set C all items in I_u.
candidates_cov_it = np.setdiff1d(c_cov, I_u, assume_unique=True)

return I_u, candidates_cov_it

# i_{u} number of items purchased by user u
        test_user = uencoder.transform([1])[0]
        i_u, u_candidates_cov_it  = candidate_items_cov_it(test_user)
```

Find similarity between each candidate item and the set I_u using Distance Covariance Similarity metric

```
def similarity_with_Iu_cov_it(c_cov, I_u):
"""
:param c : itemid of a candidate item
:param I_u : set of items already purchased by a given user
:return w : similarity between c and I_u
"""
w = 0
for iid in I_u :
# get similarity between itemid and c, if c is one of the k nearest neighbo
if c_cov in neighbors_cov_it[iid] :
w = w + similarities_cov_it[iid, neighbors_cov_it[iid] == c_cov][0]
return w
```

Rank candidate items according to their similarities using Covariance Distance Similarity metric

```
def rank_candidates_cov(candidates_cov_it, I_u):
        """
rank candidate items according to their similarities with i_u
:param candidates : list of candidate items
:param I_u : list of items purchased by the user
:return ranked_candidates : dataframe of

candidate items, ranked in descending order of similarities with I_u
        """

        # list of candidate items mapped to their
         corresponding similarities to I_u
        sims = [similarity_with_Iu_cov_it(c_cov, I_u) for c_cov in candidates_cov_i

        candidates_cov_it = iencoder.inverse_transform(candidates_cov_it )
        mapping = list(zip(candidates_cov_it , sims))

        ranked_candidates_cov = sorted(mapping,
         key=lambda couple:couple[1], reverse=True)

        return ranked_candidates_cov
```

Now that we defined all functions necessary to build our item to item top-N recommendation, let's define function that makes top- recommendations for a given user.

```
def topn_recommendation_cov(userid, N=50):

# find candidate items
I_u, candidates_cov_it = candidate_items_cov_it(userid)

# rank candidate items according to their similarities with I_u
```

```
ranked_candidates_cov =  rank_candidates_cov( candidates_cov_it, I_u)

# get the first N row of ranked_candidates to
 build the top N recommendation list
topn_cov = pd.DataFrame(ranked_candidates_cov[:N],
 columns=['itemid','similarity_with_Iu_cov_it'])

topn_cov = pd.merge(topn_cov, data2, on='itemid', how='inner')
topn_cov = topn_cov.drop_duplicates(subset=['itemid'])

return topn_cov
```

Before recommending the previous list to the user, we can go further and predict the ratings the user would have given to each of these items, sort them in descending order of prediction and return the reordered list as the new top N recommendation list.

Rating prediction using distance covariance similarity metric

```
def predict_cov(userid, itemid):
"""
Make rating prediction for user userid on item itemid
:param userid : id of the active user
:param itemid : id of the item for which we are making prediction
:return r_hat : predicted rating
"""

# Get items similar to item itemid with their corresponding similarities
item_neighbors = neighbors_cov_it[itemid]
item_similarities= similarities_cov_it[itemid]

# get ratings of user with id userid
uratings = np_ratings[np_ratings[:, 0].astype('int') == userid]

# similar items rated by item the user of i
siru = uratings[np.isin(uratings[:, 1], item_neighbors)]
scores = siru[:, 2]
indexes = [np.where(item_neighbors == iid)[0][0] for iid in siru[:,1].astype('int')
sims = item_similarities[indexes]

dot = np.dot(scores, sims)
som = np.sum(np.abs(sims))

if dot == 0 or som == 0:
return mean[userid]

return dot / som
```

Predict ratings the user would have given to the previous top-N recommended list with the Distance Covariance similarity metric

```
def topn_prediction_cov(userid):
"""
:return topn : initial topN recommendations returned by the function item2item_topN
:return topn_predict : topN recommendations reordered according to rating predictio
"""
# make top N recommendation for the active user
topn_cov = topn_recommendation_cov(userid)

# get list of items of the top N list
itemids = topn_cov.itemid.to_list()

predictions = []

# make prediction for each item in the top N list
for itemid in itemids:
r = predict_cov(userid, itemid)

predictions.append((itemid,r))

predictions = pd.DataFrame(predictions, columns=['itemid','prediction'])

# merge the predictions to topN_list and rearrange the list according to prediction
topn_predict_cov = pd.merge(topn_cov, predictions, on='itemid', how='inner')
topn_predict_cov = topn_predict_cov.sort_values(by=['prediction'], ascending=False)

return topn_cov, topn_predict_cov

topn_cov, topn_predict_cov = topn_prediction_cov(userid=test_user)
topn_predict_cov
```

	itemid	similarity_with_Iu_cov_it	prediction
8	373	0.088735	2.032342
5	342	0.094360	1.972241
23	583	0.072364	1.954189
48	587	0.060425	1.919935
17	116	0.075117	1.847081

Figure 5.19: Top recommendation using the Distance Covariance similarity metric

Evaluation of the item-based recommendation system with Covariance distance Similarity metric

```
# get examples as tuples of userids and itemids and labels from normalize ratings
raw_examples, raw_labels = get_examples(data1, labels_column='rating')

# train test split
(x_train, x_test), (y_train, y_test) = train_test_split(examples=raw_examples, labe
```

```
def evaluate_cov_it(x_test, y_test):
# print('Evaluate the model on {} test data ...'.format(x_test.shape[0]))
preds = list(predict_cov(u,i) for (u,i) in x_test)
CFI_MAE_cov = np.sum(np.absolute(y_test - np.array(preds))) / x_test.shape[0]
print('>> CFI_MAE_cov: ' + str(CFI_MAE_cov))
CFI_MSE_cov = mean_squared_error(y_test ,np.array(preds))
print('>> CFI_MSE_cov: ' + str( CFI_MSE_cov))
CFI_RMSE_cov = math.sqrt(CFI_MSE_cov)
print('>> CFI_RMSE_co ' + str(CFI_RMSE_cov))
evaluate_cov_it.list2=[CFI_MAE_cov,CFI_MSE_cov ,CFI_RMSE_cov]

return evaluate_cov_it.list2

evaluate_cov_it(x_test, y_test)
```

	Similarity Measure	RMSE	MAE	MSE
Item-Based	Euclidean Distance	*0.93329*	*0.676635*	*0.87103*
	Manhattan Distance	*0.93345*	*0.67690*	*0.87133*
	Cosine	*0.94761*	*0.67799*	*0.89796*
	Adjusted Cosine	2.40673	1.67612	5.79237
	Covariance Distance	3.14617	2.25624	9.89840

5.5.5 Results

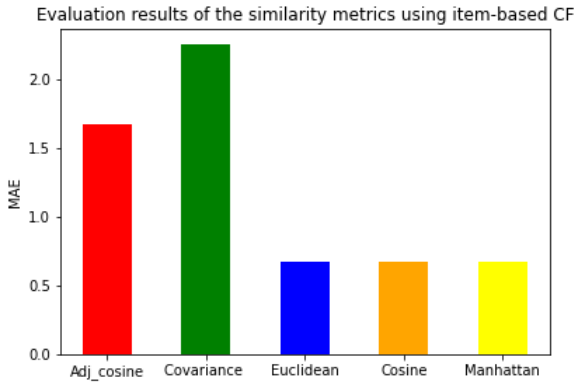

Figure 5.20: MAE Evaluation results of the similarity measures using item-based

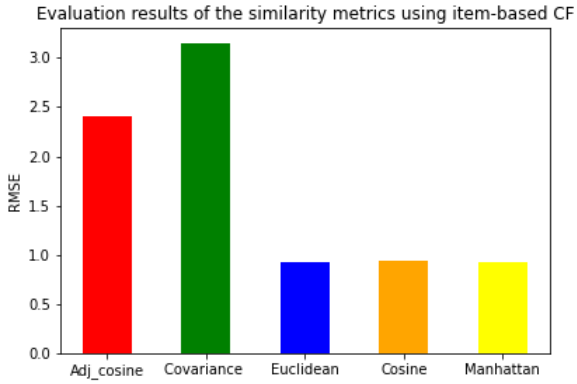

Figure 5.21: RMSE Evaluation results of the similarity measures using Item-based

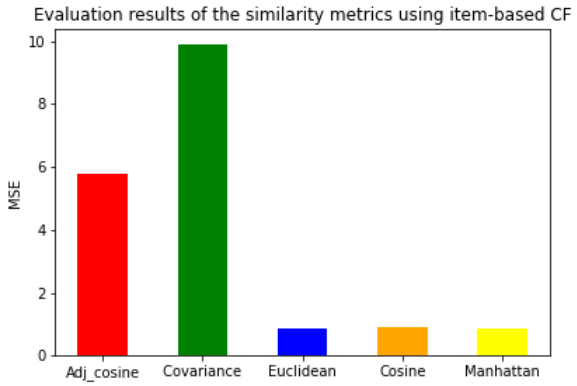

Figure 5.22: MSE Evaluation results of the similarity measures using Item-based

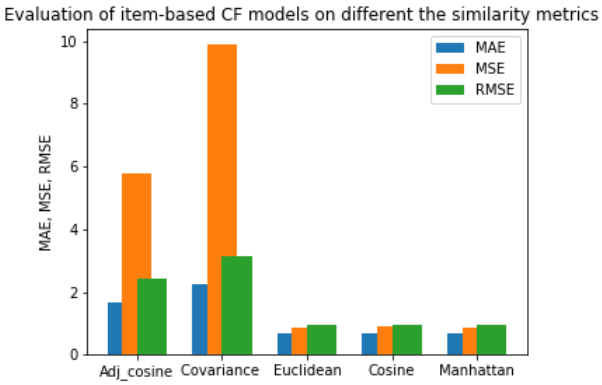

Figure 5.23: Performance Comparison Between The Evaluation results of the similarity measures using item-based

We applied Item-based CF, which were evaluated using RMSE ,MSE and MAE. In this section, we present the obtained findings. We perform item-based Neighborhood. The experimental study was carried out using different similarity measures Cosine , Adjusted Cosine ,Manhattan Distance , Euclidean Distance and the Covariance Distance , and After applying the item-based CF models, we obtained the results shown

in 5.5.4 In the table,and the graphs we present the values reached by each similarity measures in terms of RMSE, MSE, MAE

As shown in Table 5.5.4, Euclidean distance similarity provides the best result for the 3 evaluation metrics RMSE, MSE, MAE while the lower those evaluation metrics values are the better our models accuracy.For instance, Euclidean distance gets the best values: 0.676635(MAE), 0.87103(MSE), 0.93329(RMSE)

The main conclusion that can be deduced from this part of the experimental study that Euclidean distance measure outscores the other measures if it is applied in the Items-based approach ,on YASSIR EXPRESS dataset , while we may obtain different results by using another dataset .

Chapter 6

Appendix

Bibliography

[1] User modeling via stereotypes. *Cognitive Science*, 328–353.

[2] Approximation theory and methods M. J. D. (Michael James David) Powell, 1981.

[3] Automatic text processing : the transformation, analysis, and retrieval of information by computer Salton, Gerard ,1989 .

[4] Principles of forecasting: a handbook for researchers and practitioners JS Armstrong , 2001

[5] Recommender systems : the textbook Charu C. Aggarwal.

[6] Practical recommender systems Kim Falk.

[7] Hands-On Recommendation Systems with Python : Start Building Powerful and Personalized, Recommendation Engines with Python. Rounak. Banik 2018.

[8] Recommender Systems Handbook 3rd ed. 2022 Edition: Ricci, Francesco,Rokach, Lior, Shapira.

[9] Building Recommender Systems with Machine Learning and AI: Help people discover new products and content with deep learning, neural networks, and machine learning recommendations. by Frank Kane .

[10] Recommender Systems: An Introduction 1st Edition by Dietmar Jannach.

[11] Statistical Methods for Recommender Systems 1st Edition by Deepak K. Agarwal.

[12] Matrix and Tensor Factorization Techniques for Recommender Systems (SpringerBriefs in Computer Science) 1st ed. 2016 Edition .by Panagiotis Symeonidis, Andreas Zioupos.

[13] Recommender Systems: Algorithms and Applications 1st Edition, Kindle Edition by P. Pavan Kumar (Editor), S. Vairachilai (Editor), Sirisha Potluri (Editor), Sachi Nandan Mohanty (Editor)

[14] Number of netflix paying streaming subscribers worldwide from 3rd quarter 2011 to 1st quarter 2020.https://www.statista.com/statistics/250934/ quarterly-number-of-netflix-streaming-subscribers worldwide. Last accessed: 2020-12-09

[15] Burke, R. Hybrid recommender systems: Survey and experiments. User Model. User-Adapt. Interact. 2002, 12, 331–370.

[16] "Item-based collaborative filtering recommendation algorithms" , Published in the Proceedings of the 10th International Conference on World Wide Web, WWW 2001,Sarwar Badrul,Karypis George,Karypis George,Karypis George.

[17] MEASURING AND TESTING DEPENDENCE BY CORRELATION OF DISTANCESPDF Gábor J Székely, Maria L Rizzo et al.35, 6, 2007

[18] Dueck,J., Edelmann, D., Gneiting, T.& Richards, D.The affinely invariant distance correlation. Bernoulli,official journal of the Bernoulli Society for Mathematical Statistics and Probability 2014-11-04, 2305–2330.

[19] Probabilistic Matrix Factorization ,Ruslan Salakhutdinov, Andriy Mnih.

[20] Sage: Recommender engine as a cloud serviceSage: Recommender engine as a cloud ,Ronen, Royi Koenigstein, Noam Ziklik, Elad Sitruk, Mikael Yaari, Ronen Haiby-Weiss, Neta

[21] Eigentaste: A Constant Time Collaborative Filtering Algorithm Goldberg Ken ,Roeder Theresa ,Gupta Dhruv,Perkins Chris