University of Piraeus

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGIES
DEPARTMENT OF DIGITAL SYSTEMS

PRE-GRADUATE THESIS

A Comparative Study On Recommender System Approaches

Spyridon Mastrodimitris Gounaropoulos

Overseeing Professor:
Maria Halkidi, Associate Professor

PIRAEUS

JULY 2021

PRE-GRADUATE THESIS

A Comparative Study On Recommender System Approaches

Spyridon Mastrodimitris Gounaropoulos
R.N.: e14109

ABSTRACT

The subject of this paper is the presentation of Recommender Systems. The first chapters of the analysis focus on its significant diversification of application and its tremendous impact on modern online commerce and over-the-top media services. Many pages are spent on the theory behind the code, the parts of a recommendation system, and the metrics created to calculate its effectiveness. In the last part of the theoretical presentation, the sector's main challenges are presented alongside possible solutions.

The second segment focuses on the presentation of the code written for this project in the python language. For a deeper understanding and study of the topic, the MOVILENS 100K dataset was selected, 12 different recommendation algorithms were written using 2 of the most common approaches. The first group follows different uses of the K-nearest neighbor while the second two different applications of the Matrix Factorization approach. Also, four different similarity measures were used for each of the first groups with surprising results. Following the execution, metrics were extracted to evaluate their performance using the metrics analyzed in the first segment. By comparing these metrics, we try to prove which combination is the most suitable for the task.

Through this analysis, this paper does not prove that the best method for this experimentation offers the best result in every instance. This thesis aims to experiment with and observe many different recommender systems inside a specific subsector. The results can provide particular knowledge in this confinement but an improved understanding in general.

SUBJECT AREA: Information Filtering System, Recommender System, Movie Recommender System

KEYWORDS: Recommender-system, KNN, Matrix-Factorization, Python, Surprise, Cosine, MSD, Pearson, Pearson-Baseline, RMSE, MAE, Precision, Recall

ΠΕΡΙΛΗΨΗ

Η εργασία που θα παρουσιαστεί στις παρακάτω σελίδες περιέχει μελέτη των συστημάτων προτάσεων, τους τομείς χρήσης τους και στην μεγάλη σημασία που αυτά έχουν στο σύγχρονο εμπόριο και στο τομέα των υπηρεσιών περιεχομένου. Ασφαλώς υπάρχει δίνεται μία αναλυτική βιβλιογραφική ματιά στην θεωρία πίσω από την εφαρμογή, σε όλα τα ξεχωριστά στοιχεία από τα οποία ένα σύστημα προτάσεων αποτελείτε, τους τρόπους και τις μεθόδους που αξιολογούν την αποτελεσματικότητά τους. Επιπλέον δίνεται σύντομη αναφορά στα κύρια εμπόδια που εμφανίζονται στην αποδοτική και ευρεία χρήση τους.

Έχοντας τα προαναφερθέντα στοιχεία η εργασία συνεχίζει με την πρακτική τους εφαρμογή. Στα πλαίσια της εργασία γραμμένοι σε προγραμματιστική γλώσσα Python και με χρήση του dataset MOVILENS 100K αναπτύχθηκαν ποικιλία αλγορίθμων. Οι δύο κύριες κατηγορίες ήταν αλγόριθμοι κ-πλησιέστερων γειτόνων (kNN) και παραγοντοποίησης πίνακα (Matrix Factorization), ακόμα στους αλγορίθμους κ-γειτόνων δοκιμάστηκαν και 4 διαφορετικές μέθοδοι υπολογισμού ομοιότητας με πολύ ενδιαφέροντα αποτελέσματα. Στη συνέχεια εκτελείται εξαγωγή μετρήσεων γύρο από την ποιότητα των διαφορετικών μεθόδων. Τα αποτελέσματα των διαφορετικών αλγορίθμων αξιολογούνται βάση των προαναφερθέντων στοιχείων αξιολόγησης και στην συνέχεια γίνεται σύγκριση με σκοπό την διάκρισή του βέλτιστου συνδυασμού.

Παρά την ανάλυση αυτή η εργασία δεν σκοπεύει στην καταξίωση ενός από τους τρόπους ως του πανάκιου φαρμάκου στα συστήματα προτάσεων. Το φάσμα του πεδίου ανάλυσης είναι αρκετά περιορισμένο έτσι συγκεκριμένα συμπεράσματα αφορούν μόνο το στενό πλαίσιο αυτού του τομέα, όμως μέσα από αυτή την εξερεύνηση η γενική μας κατανόηση για όλες της δυνατές εφαρμογές τέτοιων αλγορίθμων έχει ωφεληθεί.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Συστήματα φιλτραρίσματος πληροφοριών, Συστήματα προτάσεων, Σύστημα προτάσεων σε περιβάλλον πλατφόρμας προβολής περιεχομένου

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Recommender-system, KNN, Matrix-Factorization, Python, Surprise, Cosine, MSD, Pearson, Pearson-Baseline, RMSE, MAE, Precision, Recall

This work is dedicated to my beloved parents, for without them, I could not have found myself in the position to create it.

Thank you

Acknowledgments

I want to thank the supervising professor, associate professor Maria Halkidi for her unending patience, constant feedback, suggestions, and overall cooperation during the writing of this paper.

I am also thankful to the online python community, as I lacked experience with the language when I decided to begin this journey. As such many times, I found myself at a dead end. With a lot of trial and error, the ease of access to simple information, and other times invaluable solutions offered by the community, I was able to move forward.

I would like to acknowledge my sincerest respect and appreciation for the professor and head of the department of digital systems, Ilias Maglogianis. For it was he who sparked my interest in the field of data analysis and pattern recognition. From then on, my academic and professional goals took a new direction, and I know it is for the better.

Contents

Introduction

Recommendation systems are a specific sector of information systems used in a great variety of subjects and industries. The necessity of analyzing massive datasets and having the ability to offer valuable conclusions is a primary concern as the amount of raw data continues to increase. I have chosen to study specifically the sector of movie recommendation.

In the previous years, Netflix has dominated the market by offering movies and series as a service at an affordable price, supplied in the comfort of one's home. Although the 'monopoly' of Netflix in the market was almost absolute, more and more competitors have made attempts to claim their place in the sun and a part in a 60 billion dollar market. Another catalyst was the global pandemic of Covid-19, as more and more people were forced to remain inside in order to reduce the spread rate of the disease and cinemas were closed in some places for more than a year, colossi of the entertainment industry turned their gaze on streaming services to reduce their deficits.

Platforms as Netflix, HBO, Amazon Prime, Disney+, and more are in a constant conflict to claim more subscribers through a wider variety of products, ambitious projects, marketing campaigns, and better services. Having to predict, choose and recommend to millions of people tens of thousands of different movies and series is a task that even the more experienced platforms still have not perfected even after years of optimization. As an information science student, I wanted to learn more and understand what practices are considered effective and even which algorithms can offer better results.

In the following pages, a complete and analytical presentation describes how the project was constructed and the code was run, the programming language and the IDE, and the libraries used. Also, the theory behind the recommendation algorithms and their differences. At last, the calculations were made to gather the metrics used for the final comparison, the comparison process, and the conclusions of this project.

Chapter 1: Recommender System Theory

The first chapter focuses on the presentations of the RS—its philosophy, the first implementations, and the many different approaches that have been proposed. Furthermore, a big part is dedicated to the different techniques used for calculating the similarity between the users and, finally, the sector's main challenges from its inception up to today.

1.1 Overview

Since the advent of the information revolution, the tremendous growth rate of data has increased the need for an application that navigates these oceans of data into seas of information and reveals pools of knowledge. One of the consequences was the given rise to information filtering systems.

An information filtering system aims to remove redundant or unwanted information from an information stream using (semi)automated or computerized methods before presenting those to the user. The IFSs main goal is the management of the information overload and increases the semantic signal-to-noise ratio. A subcategory of the broader field of information filtering systems is the recommender or recommendation system. Sometimes referred to as a recommendation engine or recommendation platform, its role is to predict the "preference" or "rating" of a user to an item. [1][2]

Recommender systems were first mentioned in 1990 in a technical report written by Jussi Karlsen titled "digital bookshelf" at Columbia University. The first time they were implemented at scale and worked through in technical information was from 1994 onwards by Jussi Karlgren. Then at SICS, and research groups led by Pattie Maes at MIT, Will Hill at Bellcore, and Paul Resnick.[3][4][5][6][7][8][9]

The variety of the fields a recommendation system can be implemented can be quite extensive. The most common examples are playlist generator of music and video streaming platforms, product recommender for online stores, content recommendation on social media sites even open web content recommendation. Other more obscure uses for such systems can vary from searching for the right restaurant to online dating to the recommendation of experts in a specific field, to collaborators to financial services. The input information can be acquired explicitly (typically by collecting users' ratings) or implicitly (typically by monitoring users' behavior. RS may use demographic features of users: age, nationality, gender. Also, other data resources can be social information: followers followed, twits, and posts commonly found and used in a 2.0 Web philosophy. Another growing trend is moving towards the use of information from the Internet of things (I.oT), for example, GPS locations, RFID, real-time personal health signals. RS tries to balance factors like accuracy, novelty, dispersity, and stability in the recommendations. In a way, a CF does not differ from the way

people make decisions. Using knowledge and experiences of friends and acquaintances to make decisions. [10][11][12][13][14][15][16]

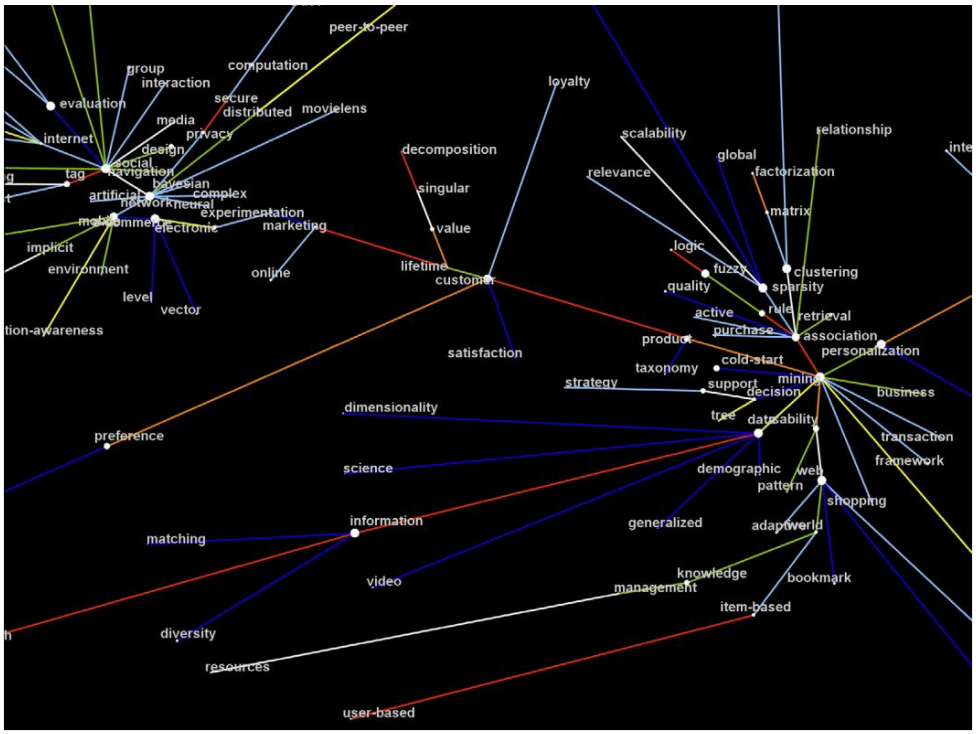


Figure 1: Most common terms of RC systems field. Based on the RS survey of 2012 by J. Bobadilla, F. Ortega, A. Hernando, A. Gutiérrez. The short edges between words-vertices represent the highest similarities. The warm colors indicate the reliability of the relationships. The size of the vertices suggests the importance of the words.

Figure 1 presents the essential terms involving an RS. The method of knowledge extraction was the following. "Common words were overlooked, like articles, prepositions and general-use words from the remaining pool, 300 terms were selected represented in the RS field. From a matrix of articles x words, wherein the importance of each word from each article was stored, a tree of relationships between the words was generated.

The size of the vertices indicates the importance of the words as a function of the parameters N^k, N^k, N^a (number of significative words in the keywords, title, and abstract) and N^k_w, N^t_w, N^a_w (number of times that the word w appears in the keywords, title and abstract). The equation used to determine the importance of each word w is as follows:

$$f_w = \frac{1}{3}\left(\frac{N_w^k}{N^k} + \frac{N_w^t}{N^t \log\frac{N^a}{N^t}} + \frac{N_w^a}{N^a \frac{N^a}{N^t}}\right) \quad (1)$$

The primary considerations that need to be taken before the designing and generation of a recommendation system are the following:

1. The type of data available in its database be ratings, user registration information, any content or features for items that can be ranked, social relationships between users, and location-aware information.

2. The filtering algorithm used (e.g., demographic, content-based, collaborative, social-based, context-aware, and hybrid). The model is chosen (e.g., based on direct use of data: "memory-based," or a model generated using such data: "model-based").

3. The employed techniques are also considered: probabilistic approaches, Bayesian networks, nearest neighbors algorithm, bio-inspired algorithms such as neural networks and genetic algorithms, fuzzy models, singular value decomposition (SVD) techniques to reduce the level of sparsity.

4. The databases' level of sparsity and the desired scalability. Performance of the system (time and memory-consuming). The objective sought is the predictions and the top recommendations as well as the desired quality of those results, e.g., novelty, coverage, and precision

Modern theoretical studies have proposed different possible approaches for the creation of a recommender system. Most importantly, the use of one course does not exclude another in tandem with the first. That fact offers an extensive group of different combinations, marking the field for its ability to adapt to specific problems providing custom solutions. The main approaches are the following: collaborative filtering, content-based filtering, knowledge-based systems, mobile recommendation systems, session-based recommendation systems, reinforcement learning for recommender systems, risk-aware recommendation systems, multi-criteria recommender systems, and Hybrid models. [17]

Collaborative filtering systems approaches aspire to build a model from a user's past behavior, items previously purchased, items selected, and the numerical evaluation given to those items and similar decisions made by other users. The next step for the model is to predict items or the ratings of items that the user or group of users may be interested in. In contrast, Content-based filtering approaches utilize item characteristics in order to recommend additional items with similar properties. Although each method has its benefits, most modern systems are a hybrid form, using at least the two main methods. [18][19]

Examples of the differences between the two can be extracted from the earlier implementations of those systems, specifically the differences between Last.Fm and Pandora Radio.

On the one hand, Last.Fm creates a station or catalog of recommended songs by observing what bands and individual tracks the user listens regularly and comparing those against the listening habits of other users. Last.Fm will play tracks that are often played by other users with similar interests but do not appear in the user's station. As this approach leverages the behavior of other users, it is an example of a collaborative filtering technique.

Pandora, on the other hand, uses the properties of either a song or an artist (using a subset of 400 attributes provided by the Music Genome Project) to seed a "station". The station plays music with similar attributes. To refine the results, user feedback is being used, deemphasizing specific attributes when a user "dislikes" a particular song and emphasizing other characteristics when a user "likes" a music piece. By focusing on the item group, it makes an example of a content-based approach.

No system is devoid of flaws. Last.Fm requires a large set of information concerning a user to make accurate recommendations. An excellent example of the cold start problem that is common in collaborative filtering. Whereas Pandora requires limited information to start, it is far more restricted in scope, meaning the recommendation have minimal variety and are often close to the original set. In the following pages, the main approaches will be further analyzed and compared to each other. [20][21][22][23][24]

1.2 A Deeper Review on the Different Approaches

Collaborative Filtering Recommendation systems (CF) in various applications have tried to provide users with an accurate recommendation to meet their needs and bring higher benefits to the platforms and providers of this service. Collaborative filtering is an effective and well-known technology in recommendation systems. Many websites, particularly Ecommerce sites, have used collaborative filtering technology in their recommendation systems to personalize the browsing experience for each user. As examples of successful implementations of collaborative filtering, Amazon has increased its sales by 29%, Netflix increased movie rentals by 60%, and Google news increased click-through rates by 30.9%.[25][26][27]

Collaboration Filtering is one of the techniques used in recommender systems, although by using the term CF, there are two different interpretations. In the newer and narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about a user's interest by collecting preferences or taste information from others (collaborating). In a more general sense, CF is the process of filtering information or patterns using techniques involving collaboration among multiple users, viewpoints and data sources. Applications of collaborative filtering typically involve enormous data sets. [28]

In both interpretations, the underlying assumption is that if person A has the same opinion as person B on an issue, A is possible to have B's opinion on a different matter than that of a randomly chosen person. The base for collaborative filtering comes from the hypothesis that people often get the best recommendations from someone with similar tastes. CF encompasses techniques for matching people with similar interests and making recommendations on this basis.

Easy categorization of Collaborative filtering (CF) algorithms is by diving them into item-based (IBCF) and user-based (UBCF). The first category focuses on collecting, analyzing, and creting conclusions around the items and then creating item clusters that are usually selected as a group. The second creates a profile of preferences for the user and then makes predictions for the next item the user would be interested in using.

1.2.1 Item-Based Collaborative Filtering (IBCF.)

IBCF allows users to give ratings about a set of elements (e.g., videos, songs, films in a CF-based website) in a way that when enough information is stored inside the system that recommendations can be made to each user based on the information provided by other users we consider them to have the most in common. CF is an interesting open research field. As noted earlier, user ratings can also be implicitly acquired (e.g., number of times a song is heard, information consulted, and access to a resource). [30][31][32][33][34][35][36][37]

Generally, input data in a recommendation system based on the CF technology consists of the user, the item, and the user opinions on observed items as a matrix m × n. Where the m symbolizes the total number of users, and n represents the total number of items. $R_{m,n}$ is the score of item In rated by the user U_m.

The CF approach uses statistical techniques to analyze the degree of similarity between users and form a set of similar-minded users called neighbors. Similarity measures are a metric of relevance between the two users seen as independent vectors. In UBCF systems, the computation of the inter vector similarity allows the creation of neighborhoods for the target user. [38][39][40]

The item-based collaborative filtering approach predicts items by inquiring into similarities between the items and other items that are already associated with the user. For example, if item A and item C are very similar. If a user approves Item A, IBCF can recommend item C. IBCF requires a set of items that the target has already rated to calculate the similarities between items and a target item. Then generates predictions in terms of the target item by combining the target user's previous preferences based on these item similarities. In IBCF, data can be collected in two ways. One is that the user explicitly gives a rating score to an item within a certain numerical scale. The alternative is that it implicitly analyzes user's purchase records or click-through rates. [29][40]

In contrast, the IBCF does not form neighborhoods after the similarity is calculated. The reason being IBCF already begins computing the similarity between co-rated items only as of the value of two vectors. This is similar to the formation process of neighbors in UBCF. Because the similarity measure plays a significant role in improving accuracy in prediction algorithms, it can effectively balance the significance of the ratings. [40][38][29]

Alternatively, item-based collaborative filtering proceeds in an item-centric manner: Creating an item-item matrix analyzing the relationships between pairs of items and inferring the target user's tastes by examining the matrix and matching that user's data. As an example, the Slope One item-based collaborative filtering family.

1.2.2 User-Based Collaborative Filtering (UBCF.)

The typical workflow of a collaborative filtering system:

- A user expresses preferences by rating items (e.g., movies, series, or books) of the system. These ratings can be considered as an approximate representation of the user's interest in the corresponding domain.

- The system matches the user's ratings against other users' and locates the people with the most similar tastes.

- The system recommends items that similar users have rated highly but have not yet been rated by this user (the absence of rating is often considered the unfamiliarity of an item).

- A fundamental problem of collaborative filtering is how to combine and evaluate the preferences of user neighbors. Often, users can rate the recommended items immediately, resulting in the system gaining an increasingly accurate representation of user preferences over time.

Collaborative filtering systems have many forms. Though it is possible for most systems to be reduced to two main steps: Search for users who share the rating patterns with the target user. Use the ratings acquired from those like-minded users found in step 1 to calculate a prediction for the target user. A specific application of this philosophy is the user-based Nearest Neighbor algorithm.

The user-based collaborative filtering approach recommends items to the target user that were already items of interest for other users who share similarities to the target user. As an example, user 1 and user 2 have very similar preferences. If user 1 likes Item X, UBCF can recommend it to user 2. UBCF requires the explicit rating scores of items rated by users to calculate similarities between users and uses k-Nearest Neighbor algorithms to find the nearest neighbors based on user similarities. Then, it generates predictions in terms of items by combining the neighbor user's rating scores based on similarity-weighted averaging. [41][29][40]

The most widely used algorithm for CF is the K Nearest Neighbors (kNN). In the user-to-user version, kNN executes the following three tasks to generate recommendations for an active user:

1. Locate k users neighbors with similar preferences to the active user

2. Implement an aggregation approach by using ratings given from the k neighbors for the items not yet rated by the target user user

3. Extract the predictions from step 2, then select the top N recommendations. Hybrid-filtering methods are common to use a combination of CF with demographic filtering or CF with content-based filtering to exploit each of these techniques' merits.

[30][42][37][43][44][45][46][47]

Memory-based methods can be defined as techniques that act only on the matrix of user ratings for items and use any rating generated before the referral process (i.e., its results are always updated). Memory-based methods usually use similarity metrics to obtain the distance between those two or two items based on their ratios. [30][48][49]

The memory-based approach uses users rating data to calculate the similarity between users or items. Typical examples of this approach are neighborhood-based collaborative filtering, item-based, and user-based top-N recommendations. As an example, in user-based approaches, the value of rating user u is being given to the item i, and it is calculated as an aggregation of some similar users' rating of the item:

$$r_{u,i} = \text{aggr}_{u' \in U} r_{u',i} \quad (2)$$

Where U denotes the group of top N users most similar to user u who rated item i, some examples of the aggregation function include:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i} \quad (3)$$

$$r_{u,i} = k \sum_{u' \in U} \text{simil}(u, u') r_{u',i} \quad (4)$$

Where k is a factor used for normalization and defined as:

$$k = 1 / \sum_{u' \in U} |\text{simil}(u, u')| \quad (5)$$

And

$$r_{u,i} = \bar{r_u} + k \sum_{u' \in U} \text{simil}(u, u')(r_{u',i} - \bar{r_{u'}}) \quad (6)$$

Where r_u is the average rating of user u for all the items rated by u.

The neighborhood-based algorithm evaluates the similarity between two users or items and offers a prediction for the user by taking the ratings' weighted average. The computation of the similarity between items or users is an integral part of this approach, and multiple measures, such as the Pearson correlation and the vector cosine-based similarity, are used for this.

The Pearson correlation similarity of two users x, y is defined as:

$$\text{simil}(x,y) = \frac{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})(r_{y,i} - \bar{r_y})}{\sqrt{\sum_{i \in I_{xy}} (r_{x,i} - \bar{r_x})^2} \sqrt{\sum_{i \in I_{xy}} (r_{y,i} - \bar{r_y})^2}} \quad (7)$$

where I_{xy} is the group of items rated by both user x and user y.

The cosine-based approach details the cosine-similarity between two users x and y as:

$$\text{simil}(x,y) = \cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}|| \times ||\vec{y}||} = \frac{\sum_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum_{i \in I_x} r_{x,i}^2} \sqrt{\sum_{i \in I_y} r_{y,i}^2}} \quad (8)$$

The user-based top-N recommendation algorithm uses the similarity-based vector model to identify the k most similar users to the target user. After the k neighborhood is found, their corresponding user-item matrices are aggregated to determine the set of items to be recommended. A popular method to find similar users is Locality-sensitive hashing, which implements the nearest neighbor mechanism in linear time.

The advantages of this approach include the explainability of the results, which is an essential part of any recommendation system, easy creation and use, easy facilitation of new data, content independence of the items being recommended, and good scaling with co-rated items.

There are though several disadvantages to this approach. Its performance decreases when the sparsity of the data increases, which frequently occurs with web-related items. That fact hinders the scalability of this approach and creates problems with big datasets. Even though it can handle new users quite efficiently because it relies on a data structure, adding new items becomes more and more complicated since that representation usually depends on a specific vector space. Adding new items requires the inclusion of the new item and the re-insertion of all the elements in the structure.

CB filtering makes recommendations based on user choices made in the past (e.g., in a web-based e-commerce RS, if the user purchased some fiction films in the past, the RS would probably recommend a new fiction film that he has not yet purchased on this website). Furthermore, another way of making suggestions is finding similarities between objects or products. That can be achieved by analyzing them like text, images, and sound. Similar objects to those already used, chosen, or preferred by the user are the next to be recommended by the system. [50][51][52]

Demographic filtering is based on the principle that individuals with certain common personal or physical attributes (sex, age, country) might also have common preferences. [53][34][44]

We use RS information to create a model that generates the recommendations. Herein, we consider a method model-based if new information from any user outdates the model. Among the most widely utilized models, we have Bayesian classifiers, neural networks, fuzzy systems, genetic algorithms, latent features, and matrix factorization.[30][34][54][55][56][57]

Certain studies have used dimensionality reduction techniques to reduce the problems from high levels of sparsity in RS databases. The reduction methods are based on Matrix Factorization (MF). MF algorithms operate by using the user-item interaction matrix to construct two lower dimensionality rectangular matrices. This group of methods became widely known during the Netflix prize challenge due to its effectiveness, analyzed by Simon Funk in his 2006 blog post, sharing his findings with the research community. We can improve the prediction results by assigning different regularization weights to the latent factors based on items' popularity and users' activeness. The idea behind matrix factorization is to reduce the number of dimensions necessary to represent users and items and rather use a lower-dimensional latent space. Since 2006 and the initial work by Funk, a multitude of matrix factorization approaches has been proposed for recommender systems. MF algorithms include Funk MF, SVD++, Asymmetric SVD, Group-Specific SVD, Hybrid, and Deep Learning. [58] [59][60][61][20]

Matrix factorization is especially adequate for processing large RS databases and providing scalable approaches. The model-based technique Latent Semantic Index (LSI) and the reduction method Singular Value Decomposition (SVD) are typically combined. SVD methods provide good prediction results but are computationally excessively costly, and they can only be deployed in static off-line settings where the known preference information does not change with time.[62][45][63][64]

We can tune the expressive power of the method by altering the number of latent factors. It has been observed that Matrix factorization with one latent factor is equivalent to a popular or a top popular recommender (e.g., recommends the items with the most interactions without personalization). Increasing the number of latent factors will vastly improve personalization, therefore the recommendation quality, until the number of factors becomes too high. At high latent factor numbers, the model starts to overfit, and the recommendation quality will decrease. A technique to avoid overfitting is to add regularization terms to the objective function. Funk MF was created as a rating prediction problem. Herein it uses explicit numerical ratings as user-item interactions. [65][24][66]

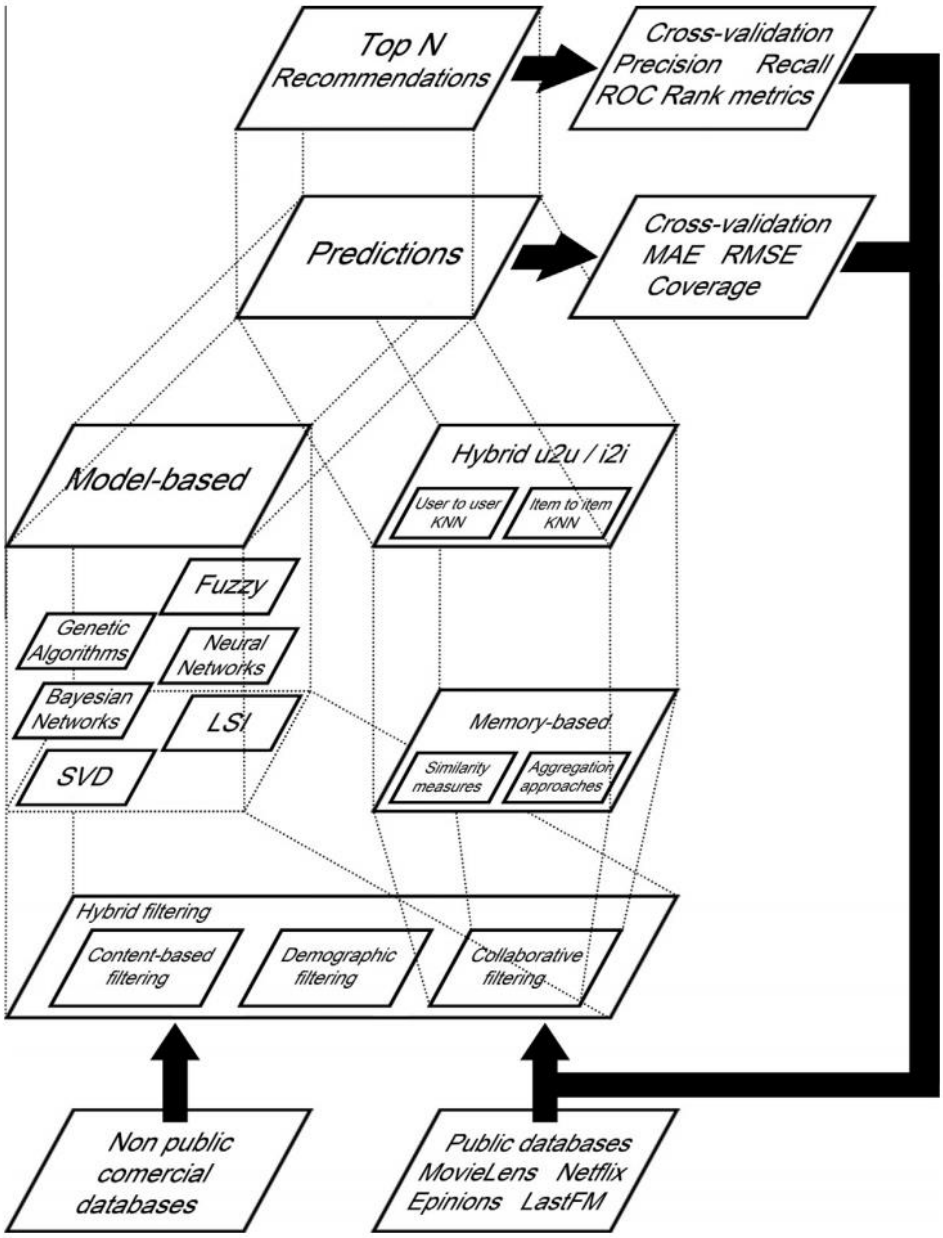


Figure 1.2.1: Traditional models of recommendations and their relationships.

RS can use clustering techniques to improve the prediction quality and reduce the cold-start problem when applied to hybrid filtering. It is typical to form clusters of items in hybrid RS. A different common approach uses clustering both for items and users (bi-clustering).

RS comprising social information has been clustered to improve the following areas: tagging, explicit social links, and explicit trust information. The graph in Figure 1.2.1 shows the most important traditional methods, techniques, and algorithms for the recommendation process and their relationships and groupings. As shown in Figure 1.2.1, we can use some of the traditional filtering methods (content-based, demographic, and collaborative) applied to databases. Model-based technologies (genetic algorithms, neural networks, etc.) make use of this kind of information. [69][70][71][72][73][74][75][76]

Typical memory-based approaches are item to item, user to user, and hybrids of the two previous. The primary purpose of both memory-based and model-based approaches is to get the most accurate predictions in users' tastes. The accuracy of these predictions may be evaluated through the classical information retrieval measures, like MAE, precision, and recall. Researchers make use of these measures to improve the RS methods and technologies.

Hybrid filtering is usually based on bioinspired or probabilistic methods. Those can be genetic algorithms, fuzzy genetic, neural networks, clustering, Bayesian networks, and latent features. A widely accepted taxonomy divides recommendation methods into memory-based and model-based method categories: The collaborative filtering approach encapsulates different calculation methods. [79][76][77][78][79][80][81][74]

Most recommender systems nowadays use a hybrid approach, using collaborative filtering, content-based filtering, and other approaches. Hybrid approaches can be implemented in multiple ways. Some are making content-based and collaborative-based predictions separately, combining them, adding content-based capabilities to a collaborative-based approach (and vice versa), or unifying the approaches into one model. Several studies empirically compared the hybrid's performance with the pure collaborative and content-based methods and demonstrated that the hybrid methods could provide more accurate recommendations than pure approaches. These methods can also overcome some common problems in recommender systems, such as cold start and the sparsity problem and the knowledge engineering bottleneck in knowledge-based approaches.[82][83]

An excellent example of the use of hybrid recommender systems is Netflix. The recommendations are made by comparing similar users' watching and searching habits (i.e., collaborative filtering) and offering movies that share characteristics with films that a user has rated highly (content-based filtering). [83]

Some hybridization techniques are:

- Weighted: Combining the value of different recommendation components numerically.

- Switching: Choosing from recommendation components and applying the selected one.

- Mixed: Recommendations from different and separate recommenders whose results are presented together as recommendations.

- Feature Combination: Features are derived from different knowledge sources and later combined and given to a recommendation algorithm.

- Feature Augmentation: Computing one feature or set of features, which is then part of the input to the following technique.

- Cascade: Recommenders are given different strict priorities, with the lower priority ones breaking ties in the scoring of the higher ones.

- Meta-level: As a first step, one recommendation technique is applied and produces some model, which is then the input used by the following technique.[86]

Session-Based Recommender Systems

These systems utilize the interactions of a user within a single session. Both Youtube and Amazon use Session-based recommender systems. These are particularly useful when the activity (e.g., past clicks, purchases, products rated) of a user is either not available or not relevant in the current session. Most session-based RS relies on the sequence of recent interactions within a session without requiring any additional details such as the history or demographics of the user. Techniques for session-based recommendations are primarily based on generative sequential models such as Recurrent Neural Networks, Transformers, and other deep learning-based approaches. [85][86][87][88][89][90][30]

Multi-criteria recommender systems

MCRS can be defined as systems of recommendation that incorporate preference information upon multiple criteria. Not by developing recommendation techniques based on a single criterion value, the overall preference of user u for the item i, but try to predict a rating for unexplored items of u by exploiting preference information on various criteria that affect this overall preference value. Several researchers approach MCRS as a multi-criteria decision-making problem and apply MCDM methods and techniques to implement MCRS systems.[91][93]

Risk-aware recommender systems

Most existing approaches focus on recommending relevant content to users by using contextual information yet do not take into consideration the risk of upsetting the user with unwanted notifications. It is crucial to acknowledge the risk of disturbing the user by pushing recommendations in certain circumstances, for example, during an important professional meeting, early in the morning, or late at night. As a result, the performance of the RS depends in part on the degree to which it has incorporated the risk into the process of recommendations. One solution is to manage this issue is DRARS, a system that models the context-aware recommendation as a bandit problem. This system blends a content-based technique and a contextual bandit algorithm.[93]

Mobile recommender systems

Mobile recommender systems make use of internet-accessing smartphones to offer personalized, context-sensitive recommendations. This is a most challenging area of research as mobile data is more complex than data that recommender systems often have to tackle. It is heterogeneous, noisy, requires temporal and spatial auto-correlation, and has validation and generality problems.[94]

The factors that could affect the mobile recommender systems and the accuracy of prediction results are the context, the recommendation method, and privacy. Also, mobile recommender systems suffer from a transplantation problem, and recommendations may not apply in all regions (for example, it might be problematic to recommend a recipe in an area where all of the ingredients may not be available).[95]

An example of a mobile recommender system is the approaches chosen by Lyft and Uber to set driving routes for taxi drivers in a city. This system uses the GPS data of the routes that taxi drivers take while working, including location (latitude and longitude), time stamps, even operational status. Those data are used to provide recommendations, a list of pickup points along a route to optimize occupancy times and profits.[94]

Reinforcement learning for recommender systems

The recommendation problem can is in a way, a particular instance of a reinforcement learning problem. The user is the environment inside which the recommender system acts to receive a reward, for example, a click or engagement by the user. An aspect of reinforcement learning that is of particular use in the recommender system field. The reason behind this fact is that policies or models can be learned by providing rewards to the recommendation agent. In contrast with traditional learning techniques, where relying on supervised learning approaches are less flexible. Reinforcement learning recommendation techniques grand the potential to train models that can be optimized directly on engagement and user interest metrics.[96]

1.3 Similarity Measures

For the kNN algorithms to locate the neighbors with the same characteristics and construct a cluster of same preference users, it needs to calculate their similarity. The metrics used for that approximation of similarity are the similarity measures. Most of the time, analyzing the users or items as vertices and calculating the difference of their angle. Many different ideas have been proposed; in the following pages, the most common are presented.

Cosine vector similarity is one of the famous metrics in statistics. It notionally considers only the angle of the two vectors without the magnitude, making it an instrumental measurement with data missing preference information as long as it can count the number of times that term appears in the data. In (2), the cosine vector similarity looks into the angle between the two vectors (the target Item i and the other Item j) of ratings in n-dimensional item space. $R_{k,j}$ is the rating of the target Item i by User k. $R_{k,j}$ is the rating of the other Item j by user k. n is the number of all rating users gave to Item i and Item j. [97]

$$sim(i,j) = \cos(\vec{i},\vec{j}) = \frac{\vec{i} \times \vec{j}}{\|\vec{i}\|^2 \times \|\vec{j}\|^2} = \frac{\sum_{k=1}^{n} R_{k,i} R_{k,j}}{\sqrt{\sum_{k=1}^{n} {R_{k,i}}^2 \sum_{k=1}^{n} {R_{k,j}}^2}} \quad (9)$$

When the angle between two vectors is close to the 0 degrees mark (they are in the same direction), the Cosine similarity value, $sim(i,j)$, is 1, meaning almost identical. When the angle between two vectors closes near 90 degrees, $sim(i,j)$ is 0, meaning completely irrelevant. Furthermore, when the angle between two vectors is near 180 degrees (they are in the opposite direction), $sim(i,j)$ is -1, meaning very dissimilar. In the case of information retrieval using CF, $sim(i,j)$ ranges from 0 to 1. This is because the angle between two-term frequency vectors cannot be greater than 90 degrees.[74]

Pearson correlation coefficient is one of the often-used methods in CF to measure how greater a number in one series is relative to a corresponding number. As the following formula shows, it is used to measure the linear correlation between two vectors (Item i and Item j)

$$sim(i,j) = \frac{\langle R_{k,i} - A_i,\ R_{k,j} - A_j \rangle}{\|R_{k,i} - A_i\| \|R_{k,j} - A_j\|} = \frac{\sum_{k=1}^{n}(R_{k,i} - A_i)\,(R_{k,j} - A_j)}{\sqrt{\sum_{k=1}^{n}(R_{k,i} - A_i)^2 \times \sum_{k=1}^{n}(R_{k,j} - A_j)^2}} \quad (10)$$

It calculates the tendency of two series of numbers, paired up one-to-one, to move together. When the two vectors have a high tendency, the correlation, $sim(i,j)$, is close to 1. When the two vectors have a low tendency, $sim(i,j)$ is close to 0. When two vectors have an opposite tendency, $sim(i,j)$ is close to -1. The item-based similarity is computed with the curated items where users rated both (Item i and Item j). $R_{k,j}$ is the rating of the target Item i given by User k. $R_{k,j}$ is the rating of the other Item j given by User i. A_i is the average rating of the target Item i for all the co-rated users, and A_j is the mean rating of the other Item j for all the co-rated users. n is the number of ratings the users gave to Item i and Item j. [97]

The **Euclidean distance** method uses the distance between items. With that forms coordinates to put preference values between items and measures the Euclidean distance between each point. When the distance value between two points, $sim(i,j)$, is considerable, it means the two points are not similar. When $sim(i,j)$ is negligible, it means two points are identical. The Euclidean distance formula is given below:

$$sim(i,j) = \sqrt{\sum_{k=1}^{n}(R_{k,i} - R_{k,i})^2} \quad (11)$$

$R_{i,j}$ is the ratings of the target Item i given by User k. n is the ratings of the other Item j given by User k. n is the number of rating users to Item i and Item j.

The **Tanimoto coefficient** is known as the Jaccard similarity coefficient. It does not take into account the preference values of an item rated by a user. It only takes into consideration if users express a preference. The Tanimoto coefficient is the ratio of the size of the intersection, or overlay, in two users' preferred items to produce the union of users' preferred items. When two items are entirely overlapped, Tanimoto coefficient, $sim(i,j)$, is 1. When two items are not entirely overlapped, $sim(i,j)$ is zero.

As shown in (5), the degree of overlap between two sets is examined to compare the similarity and diversity of two sets. f_i It is a set of Item i for which users express their preference. f_i It is a set of Item j for which users express their preference. $f_i \cap f_j$ It is an intersection of Item i and Item j for items where preference is expressed by users.

$$sim(i,j) = \frac{|f_i \cap f_j|}{|f_i| + |f_j| - |f_i \cap f_j|} \quad (12)$$

Plenty of users tend not to rate items, or the RS might not have enough information for a user. Tanimoto would be helpful to compute similarity but only if preference information as Boolean type is available. Once CF calculates the similarity between users (in UBCF) or items (in IBCF) and then finds the set of most similar users or similar items, it generates a prediction of the target user's interest as the most critical step in CF.

Since UBCF gets the user's neighborhood, UBCF can calculate the predictive rating for the target User u on the target Item i. Then Scaled by the weighted average of the neighbors' ratings on the target Item i as following: [38][34]

$$P_{u,i} = A_u + \frac{\sum_{w=1}^{n} (R_{w,i} - A_w) \times sim(u,w)}{\sum_{w=1}^{n} sim(u,w)} \quad (13)$$

A_u Is the average ratings of the User u to all other rated items, and A_w is the average ratings of the neighbor User w to all other rated items. $R_{w,i}$ Is the rating of the neighbor User w to the target item i. $sim(i,j)$ is the similarity of the target User u and the neighbor User w, and n is the total number of neighbors.

Foremost the IBCF has got the neighborhood of items, then IBCF strives to make sure the target user does rates similar items. To make sure that the prediction is in the predefined range, the predictive rating for the target User u on the target Item i is scaled by the weighted average of all neighbor items' ratings given by the target User u according to the following formula.

$$P_{u,i} = \frac{\sum_{j=1}^{n} R_{u,j} \times sim(i,j)}{\sum_{j=1}^{n} sim(i,j)} \quad (15)$$

$R_{u,i}$ Is the rating of the target User u to the target Item i. sim(i,j) is the weighted similarity of the target Item i and the neighbor Item j, n is the total number of neighbor items. [98][29] [99]

1.4 Main Challenges of Recommendation Systems

1.4.1 Scalability

The amount of data utilized as input to an RS is snowballing as more users and items are added. For example, for a popular website, the size of stored user behavior data can easily reach TBs per day. Despite the immense amount of data, most RS aspire to respond interactively in less than a second to keep the users engaged. A significant challenge here is to design efficient learning algorithms that can handle such large-scale datasets.

CF algorithms are unable to function on users independently. The algorithms learn jointly from the behavior of all users. As the user numbers continue to grow, the time complexity keeps scaling. For example, $k - NN$ method (Resnick, 1994) first finds a neighborhood for each user by thresholding similarity scores or finding k most similar users. It then generates predictions by calculating the weighted average of the ratings of the neighboring users. Since the similarities among all pairs of users are computed, the running time of the method is quadratic in the number of users.[8]

Multiple approaches have been suggested to tackle the issue of scalability. Das (2007) uses an online learning algorithm that processes each user and updates parameters sequentially. The online learning algorithm is generally speaking more efficient than the batch method. The main reason is its incorporation of updates immediately rather than waiting to process the accumulative cluster of data from all users at once. Furthermore, Gemulla (2011) proposes a distributed algorithm in which most computations could be conducted on multiple machines in parallel. Most RS can be modeled as a problem of matrix completion, with rows being users, columns being items, and entries being ratings.[116][59][117]

The objective is to complete the matrix given partially observed entries. The method for matrix completion that utilizes trace norm regularization (Jaggi and Sulovsk in 2010, Xin and Jaakkola in 2012, Yang and Yuan in 2013, Ji and Ye in 2009) can offer solutions. The convexity of the problem guarantees that any local minimum is a global minimum, meaning the final result does not rely on the initialization, and its convergence property can be theoretically analyzed. Although, the associated convexity creates optimization difficulties. Many proposed algorithms, unfortunately, only work for small data sets. [118][119][120][121]

1.4.2 Privacy

Understanding the value of the users' data, the majority of websites collect as much user data as possible. This fact raises privacy concerns because the data may contain sensitive information that the users wish to keep private, for example, the addresses physical and digital and, obviously, payment history. Even though users are occasionally presented with privacy policies concerning data usage, they usually have no direct control over the data.

Most research on privacy protection focuses on publishing data from a central database, such as medical records collected by hospitals. The privacy mechanisms can be separated into interactive and non-interactive. The non-interactive setting publishes a "sanitized" version of the data. This setting involves data permutation and removing identifier information such as names and addresses (Sweeney (2002), Machanavajjhala (2007)). Though, without specified utilities of the data, general sanitization methods may not achieve satisfying results (Chawla (2005)). Furthermore, the sanitized data could still be used to identify users when additional information is provided. Netflix released its user rating data for its recommendation competition by removing all identifiers. Barbaro (2006) shows that the Netflix users can still be identified by referencing public IMDB user data because it is rare for two different users to share a remarkably similar taste in movies. [123][124][125][126]

In contrast, interactive settings allow users to pose queries about the data and receive (possibly noisy) data. A commonly used concept is Differential Privacy (DP). That metric guarantees that in the query response, it is difficult to determine if a user is in the database or not. Unlike the anonymity approach, even if an adversarial user has additional background information, privacy is guaranteed. The most favored method of implementing DP is by adding noise to the querys' response. The degree of noise depends on the sensitivity of the query function to a single data record.

Despite the adequate theoretical results, setting up a secure central database for RS that holds all user data is complex. To be more specific, it requires users to release their data to a trusted system that operates in a restricted manner. Considering the potential business interests involved and the complexity of restrictions applied to the system, the setting can be unrealistic. If each user protects his data on a personal computer and only share limited information with central servers, privacy can be preserved.

Matrix completion accuracy depends directly on the amount of information that each user is willing to share with the system (Alvim, 2012). It might be possible for some cases to avoid this statistical trade-off by building Peer-to-Peer networks with homomorphic encryption, which is computationally challenging. [127]

The statistical trade-off between accuracy and privacy depends on the notion of privacy we adopt. A commonly utilized privacy concept is Differential Privacy (DP) (Dwork, 2008), first introduced to protect information leaked from database queries. [128]

Users can agree to a trusted party to hold and aggregate their data and perform computations. Privacy guarantees are then demanded of any results published beyond the trusted party and the users. In a set of recommendations, differential privacy can be achieved through obfuscation (adding noise) without a significant loss of accuracy (McSherry and Mironov, 2009). In contrast to McSherry and Mironov (2009), we view the system as an untrusted entity and assume that users wish to guard their data. We part from differential privacy and separate the computations that can be done locally by individual users but the computations that the system must perform. [128]

For example, only the item features have to be solved by the system in terms of low-rank matrices. The corresponding users' features can be acquired locally by the users and then used for the rankings.

From this perspective, we divide the users into two groups, the set of users who openly share their preferences and the more extensive set of private users who require explicit privacy guarantees. It happens that theoretically and possible to demonstrate empirically that a moderate number of public users is sufficient for an accurate estimation of item features. The remaining private users can make use of these item features without any release of information. Furthermore, we propose a new second-order privacy concept that uses limited (second-order) information from private users as well and illustrates how recommendations can be further improved while still maintaining marginal deniability of private information.

Firstly, we provided explicit guarantees for estimating item features in matrix completion problems. Secondly, if shared with new users, the resulting estimates can be used to predict their ratings depending on the level of overlap between their private ratings and the relevant item subspace. The empirical results demonstrate that even a small number of public users with a large number of ratings suffices for an excellent performance.

Third, introducing a new privacy mechanism for releasing second-order information needed for estimating item features while maintaining first-order deniability. Experiments show that this mechanism indeed can perform adequately in comparison to other mechanisms. It is believed that allowing different levels of privacy is an exciting research topic.

1.4.3 Cold start

The cold-start problem occurs when it is impossible to make reliable recommendations because of the initial lack of ratings. We can separate the kinds of cold-start problems into new community, new item, and new user. The last is the most important in RS that are already in operation. The new community problem refers to the challenge of starting up an RS in obtaining a sufficient amount of data (ratings) to make reliable recommendations. [129][30][23][132]

Two common approaches utilized for tackling the new item problem are encouraging the users to make ratings through different means and taking CF-based recommendations when there are enough users and ratings. This problem arises because the new items that enter an RS usually lack initial ratings, which is unlikely to be recommended. This results, the item going unnoticed by a large part of the community of users, and as they are unaware of it, they do not rate it. This way, we can enter a vicious circle in which a set of RS items are left out of the ratings/recommendations process. [132][133]

The new item problem has less of an impact on RS in which the items can be discovered via other means (e.g., movies) than in RS, where this is not the case (i.e., e-commerce, blogs, photos, videos, and more). A standard solution to this problem is to have a group of motivated users responsible for rating each new item in the system.

The new user problem is perhaps one of the great difficulties faced by the RS in operation. Since new users in the RS have not yet provided any rating in the RS, they cannot receive any personalized recommendations based on memory-based CF. When the users enter their firsts ratings, they expect the RS to offer them personalized recommendations. However, the number of ratings introduced in the RS is usually not yet sufficient to make reliable CF-based recommendations, and, therefore, new users may feel that the RS does not offer the service they expected and may stop using it. [134][135]

The common strategy to tackle the new user problem consists of turning to additional information to the group of ratings in order to be able to make recommendations based on the data available for each user. The cold-start problem is often faced using hybrid approaches (usually CF-content based RS, CF-demographic based RS, CF-social based RS). Leung proposes a novel content-based hybrid approach that uses cross-level association rules to integrate content information about domain items. Kim uses collaborative tagging employed as an approach in order to grasp and filter users' preferences for items. They explore the advantages of collaborative tagging for data sparseness and cold-start users (they collected the dataset by crawling the collaborative tagging delicious site). [136][137] [138]

Weng combines the implicit relations between users' item preferences and the additional taxonomic preferences to have higher quality recommendations and alleviate the cold-start problem. [139]

Loh represents user's profiles with information extracted from their scientific publications. Martinez presents a hybrid RS that combines a CF algorithm with a knowledge-based one. Chen and He propose a number of common terms/ term frequency (NCT/TF) CF algorithm based on demographic vector. Saranya and Atsuhiro propose a hybrid RS that utilizes latent features extracted from items represented by a multi-attributed record using a probabilistic model. Park proposes a new approach: they use filterbots and surrogate users rate items based only on user or item attributes. [140][141][142][143]

Chapter 2: Evaluation Measures

The second chapter focuses on the difficulties of calculation the efficacy of RS. Moreover, it analytically presents the generally acceptable metrics proposed for the execution of this task. They are divided into multiple categories based on the RSs' effect upon a particular aspect of its function.

2.1 The Challenging Goal of Universality

Having a wide range of problems and solutions in the RS field, evaluating the results of each method became paramount. Research in the recommender system field requires both quality measures and evaluation metrics to prove the quality of the algorithms, techniques, and methods for predictions and recommendations. Evaluation metrics and evaluation frameworks facilitate comparisons between solutions for the same problem and selection from different promising lines of research that generate superior results. [31][100][101][102][32][37]

The extensive use of these evaluation measures has allowed the rapid advance and optimization of the RS as a field. That set of existing representative evaluation measures has standard formulations and groups of open RS public databases that have facilitated quality comparisons for newly proposed recommendation methods and previously published methods. Thus, RS methods and algorithms research has continued to progress. The most commonly used quality measures are the following: (1) prediction evaluations, (2) evaluations for recommendation as sets, and (3) evaluations for recommendations as ranked lists. [64][101][102]

Evaluation metrics can be classified in more accurately specified groups: (a) prediction metrics such as the accuracy metrics (Mean Absolute Error (MAE), Root of Mean Square Error (RMSE), Normalized Mean Average Error (NMAE)) and the coverage, (b) set recommendation metrics (Precision, Recall and Receiver Operating Characteristic (ROC)), (c) rank recommendation metrics (half-life and the discounted cumulative gain) and (d) diversity metrics (diversity and the novelty) of the recommended items. The validation process is performed by employing the most common cross-validation techniques (random sub-sampling and k-fold cross-validation). For cold-start situations, due to the limited number of users (or items) votes involved, the usual method chosen to carry out the experiments is leave-one-out cross-validation. [103][104][31][102][23]

General publications and reviews also exist, including the most commonly accepted evaluation measures: mean absolute error, coverage, precision, recall, and derivatives of these: RMSE, NMAE, ROC, and fallout. Goldberg focuses on the aspects not related to the evaluation. Breese compares the predictive accuracy of various methods in a set of representative problem domains.[109][106]

The majority of articles discuss attempted improvements to the accuracy of RS results (RMSE, MAE, among others). It is also common to attempt to improve the recommendations (precision, recall, ROC, and other metrics). However, additional objectives should be considered for generating greater user satisfaction, such as topic diversification and coverage serendipity.[107]

Currently, the field has a growing interest in generating algorithms with diverse and innovative recommendations, even at the expense of accuracy and precision. To evaluate these aspects, various metrics have been proposed to measure recommendation novelty and diversity. [108][109]

The frameworks aid in defining and standardizing the methods and algorithms employed by RS, as well as the mechanisms to evaluate the quality of the results. Among the most influential papers that propose CF frameworks are Herlocker which evaluates the following: similarity weight, significance weighting, variance weighting, selecting neighborhood, and rating normalization.[32]

Hernández and Gaudioso propose a framework in which any RS is formed by two different subsystems, one to guide the user and the other to provide useful/interesting items. Koutrika is a framework that introduces levels of abstraction in the CF process, making the modifications in the RS more flexible. Antunes presents an evaluation framework assuming that evaluation is an evolving process during the system lifecycle.[102][110]

The majority of RS evaluation frameworks proposed until now suffer from two primary deficiencies the lack of formalization and the absence of standardization. Although the evaluation metrics are well defined, there are various details in the implementation of the methods that, in the event they are not specified, can lead to different results in similar experiments. The second deficiency is the absence of standardization of the evaluation measures in aspects such as novelty and trust of the recommendations.

Bobadilla provides a complete series of mathematical formalizations based on sets theory. Authors provide a set of evaluation measures, including the quality analysis of the following aspects: predictions, recommendations, novelty, and trust. Presented next is a representative selection of the RS evaluation quality measures most often used in the bibliography.[37]

2.2 Quality of Prediction

MAE, RMSE, and Coverage

In order to measure the accuracy of an RS through its results, it is usual to use the computation of some of the most common prediction error metrics, amongst which the Mean Absolute Error (MAE) and its related metrics: mean squared error, root mean squared error, and normalized mean absolute error stand out.

We define U as the set of the RS users, i as the set of the RS items, $r_{u,i}$ the rating of user u on item i, ● the lack of rating ($r_{u,i}$ = ● means user u has not rated item i), $p_{u,i}$ the prediction of item i on user u.

Let, $O_u = \{i \in I | p_{u,i} \neq \bullet \wedge r_{u,i} \neq \bullet\}$ is rated by users u having prediction values. We define the MAE and RMSE of the system as the average of the user's MAE. We remark that the absolute difference between prediction and real value, $|p_{u,i} - r_{u,i}|$ informs about the error in the prediction.

$$\mathrm{MAE} = \frac{1}{\#U} \sum_{u \in U} \left(\frac{1}{\#O_u} \sum_{i \in O_u} |p_{u,i} - r_{u,i}| \right) \tag{16}$$

$$\mathrm{RMSE} = \frac{1}{\#U} \sum_{u \in U} \sqrt{\frac{1}{\#O_u} \sum_{i \in O_u} (p_{u,i} - r_{u,i})^2} \tag{17}$$

The coverage could be described as the capacity of predicting from a metric applied to a specific RS. In short, it calculates the percentage of situations in which at least one k-neighbor of each target user can rate an item that has not been rated yet by that target user. We defined $K_{u,i}$ as the set of neighbors of u which have rated the item i. We define the coverage of the system as the average of the user's coverage:

$$C_u = \{i \in I | r_{u,i} = \bullet \wedge K_{u,i} \neq \varnothing\}, \quad D_u = \{i \in I | r_{u,i} = \bullet\} \tag{18}$$

$$coverage = \frac{1}{\#U}\sum_{u\in U}\left(100 \times \frac{\#C_u}{\#D_u}\right) \quad (19)$$

2.3 Quality of Recommendation

Precision, Recall, and Prediction

The users' confidence in a particular RS does not depend directly on the accuracy of the set of possible predictions. A user gains confidence in the RS when this user accepts the reduced set of recommendations made by the RS as agreeable to his preferences. In this section, the following three most widely used recommendation quality measures are defined as precision, which suggests the proportion of relevant recommended items from the total number of recommended items, recall, which suggests the proportion of relevant recommended items from the number of relevant items, and F1, which is a synthesis of precision and recall.

Let X_u as the set of recommendations to user u, and Zu as the set of n recommendations to user u, we will represent the evaluation precision, recall, and F1 measures for recommendations obtained by making n test recommendations to the user u, taking a θ relevancy threshold. As long as that all users accept n test recommendations:

$$precision = \frac{1}{\#U}\sum_{u\in U}\frac{\#\{i \in Z_u | r_{u,i} \geqslant \theta\}}{n} \quad (20)$$

$$recall = \frac{1}{\#U}\sum_{u\in U}\frac{\#\{i \in Z_u | r_{u,i} \geqslant \theta\}}{\#\{i \in Z_u | r_{u,i} \geqslant \theta\} + \#\{i \in Z_u^c | r_{u,i} \geqslant \theta\}} \quad (21)$$

$$F1 = \frac{2 \times precision \times recall}{precision + recall} \quad (22)$$

Rank measures

The importance of the first items recommended to the user is elevated further as the number n recommended items increases. The mistakes incurred in these items are more serious errors than those in the last items on the list. The measures used to evaluate these situations are called rank measures.

Among the ranking measures most often used are the following standard information retrieval measures: (a) half-life (7), which assumes an exponential decrease in the interest of users as they move away from the recommendations at the top and (b) discounted cumulative gain (8) wherein decay is logarithmic. [106][111]

$$\mathrm{HL} = \frac{1}{\#U} \sum_{u \in U} \sum_{i=1}^{N} \frac{\max(r_{u,p_i} - d, 0)}{2^{(i-1)/(\alpha-1)}} \tag{23}$$

$$\mathrm{DCG}^k = \frac{1}{\#U} \sum_{u \in U} \left(r_{u,p_1} + \sum_{i=2}^{k} \frac{r_{u,p_i}}{\log_2(i)} \right) \tag{24}$$

$p_{1,\ldots}, p_n$represents the recommendation list, r_{u,p_i} represents the true rating of the user u for the item p_i , k is the rank of the evaluated item, d is the default rating, α is the number of the item on the list that has a 50% chance that the user will review that item.

Novelty and diversity

The novelty evaluation measure indicates the scope of difference between the items recommended to the user and those already know. The diversity quality measure indicates the degree of differentiation among the recommended items. Currently, novelty and diversity measures do not have a standard; therefore, different authors propose different metrics. Certain authors have used the following: [112][109][113]

$$diversity_{Z_u} = \frac{1}{\#Z_u(\#Z_u - 1)} \sum_{i \in Z_u} \sum_{j \in Z_u, j \neq i} [1 - sim(i,j)] \tag{25}$$

$$novelty_i = \frac{1}{\#Z_u - 1} \sum_{j \in Z_u} [1 - sim(i,j)], \quad i \in Z_u \tag{26}$$

Here, $sim(i,j)$ indicates item to item memory-based CF similarity measures. Z_u indicates the set of n recommendations to user u.

Stability

Stability in the prediction and recommendation influences the trust the user has towards the RS. An RS is stable if its predictions can adjust slowly over time and not change dramatically in small intervals. Adomavicius and Zhang propose a quality measure of stability, MAS (Mean Absolute Shift). This measure is defined through a set of available rating R1 and a set of predictions of all unknown ratings, P1. For an interval of time, users of the RS will have rated a subset S of these unknown ratings, and the RS can now make new predictions, P2. MAS is defined as follows: [114]

$$stability = \text{MAS} = \frac{1}{|P_2|} \sum_{(u,i) \in P_2} |P_2(u,i) - P_1(u,i)| \tag{27}$$

Reliability

The reliability of a prediction or a recommendation informs about how seriously we may consider this prediction. When RS recommends an item to a user with a prediction of 4.5 on a scale $\{1, \dots, 5\}$, this user expects to be satisfied by this item. However, this value of prediction (4.5 over 5) does not reflect which degree the RS has concluded that the user will approve of the item that has been recommended (with a value of 4.5 over 5). Indeed, this prediction of 4.5 is much more reliable if it has been obtained utilizing multiple tens or hundreds of users.

In Hernando, a reliability measure is proposed according to the usual notion that when a prediction becomes more reliable, the less liable to be wrong. Although this reliability measure is not a quality measure used for comparing different techniques of RS through cross-validation, this can be regarded as a quality measure associated with a prediction and a recommendation. Hence, the RS provides a pair of values (prediction value, reliability value), through which users may balance their preference: for example, users would probably prefer the option (4, 0.9) to the option (4.5,0.1). Consequently, the reliability measure proposed in Hernando provides a new understandable factor, which users may consider for making decisions. Nevertheless, this reliability measure is just constrained to those RS based on the kNN algorithm. [115]

The definition of reliability on the prediction, $p_{u,i}$ is based on two numeric factors: $s_{u,i}$ and $v_{u,i}$., $s_{u,i}$measures the similarity of the neighbors used for making the prediction $p_{u,i}$, $v_{u,i}$ measures the degree of disagreement between these neighbors rating the item i. Finally, the reliability measure is defined as follows:

$$s_{u,i} = \sum_{v \in K_{u,i}} sim(u, v) \tag{28}$$

$$f_S(s_{u,i}) = 1 - \frac{\bar{s}}{\bar{s} + s_{u,i}} \tag{29}$$

$$f_v(v_{u,i}) = \left(\frac{\max - \min - v_{u,i}}{\max - \min}\right)^{\frac{\ln 0.5}{\ln \frac{\max - \min - \bar{v}}{\max - \min}}}, \quad v_{u,i}$$
$$= \frac{\sum_{v \in K_{u,i}} sim(u, v)(r_{v,i} - \bar{r}_v - p_{u,i} + \bar{r}_u)^2}{\sum_{v \in K_{u,i}} sim(u, v)} \tag{30}$$

here $\bar{s}$ and $\bar{v}$ are respectively the median of the values of $s_{u,i}$ and $v_{u,i}$ in the specific RS. $K_{u,i}$ is the set of neighbors of u which have rated the item i. {min,..., max} is the discrete range of rating values.

Chapter 3: Recommendation System Testing

This chapter focuses on the presentation of the algorithms selected to be tested and compared as well as the chosen similarity measures to be combined with. After the analytical presentation of the theory comes the technical specification, the environment, and the tools used for the testing. This chapter aims to offer a better understanding of this work's scope and depth as it uses a system suboptimal to its purpose.

3.1 Examined Algorithms

For the evaluation of different recommended systems, four algorithms were selected. Two of them belonged to the nearest neighbor approach and two to the matrix factorization approach. The four algorithms were kNN with means, kNN with Z-score normalization, non-negative Matrix Factorization (NMF), and the singular value decomposition algorithm (SVD).

K Nearest Neighbor

The training examples are vectors inside a multidimensional feature space, each with a class label. The algorithm's training phase consists exclusively of storing the training samples' feature vectors and class labels. In the phase of classification, k is a user-defined constant, and an unlabeled vector (a query or test point) is classified by assigning the most frequent label among the k training samples nearest to that query point. A commonly used distance metric for continuous variables is Euclidean distance. For discrete variables, such as for text classification, another metric can be used, such as the overlap metric (or Hamming distance). For example, in the context of gene expression microarray data, kNN has been employed with correlation coefficients, such as Pearson and Spearman, as a metric. Often, the classification accuracy of kNN can be improved significantly if the distance metric is learned with specialized algorithms such as Large Margin Nearest Neighbor or Neighborhood components analysis.

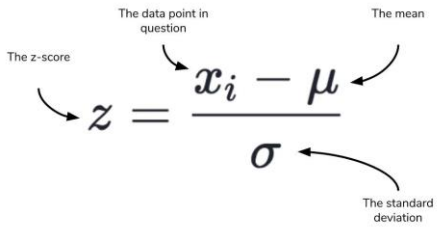
The specific kNN algorithm provided by surprise takes into account the mean rating of each user. The prediction follows the function:

$$\hat{r}_{ui} = \mu_i + \frac{\sum\limits_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - \mu_j)}{\sum\limits_{j \in N_u^k(i)} \text{sim}(i,j)} \tag{31}$$

K Nearest Neighbor with Z-score Normalization

Z-score is a numerical measurement that illustrates a value's relationship to the mean of a group of values. The Z-score is calculated in terms of standard deviations from the mean. When Z-score is 0, it indicates that the data points' score is identical to the mean score. A Z-score of 1.0 would hint at a value that is one standard deviation from the mean. Z-scores may be either positive or negative, with a positive value suggesting the score is above the mean and a negative score indicating it is below the mean. There are two usual approaches to bringing different features onto the same scale: normalization and standardization. Normalization usually refers to the rescaling of the features to a range of [0, 1], which is a particular case of min-max scaling. Using standardization, we center the feature columns at

mean 0 with a standard deviation of 1 so that the feature columns take the form of a normal distribution, making it easier to learn the weights. Standardization maintains useful information about outliers and makes the algorithm less sensitive to them in contrast to min-max scaling.



(32)

Thus, the function that produces the prediction differs from the kNN-means and is:

$$\hat{r}_{ui} = \mu_i + \frac{\sum_{j \in N_u^k(i)} \text{sim}(i,j) \cdot (r_{uj} - \mu_j)}{\sum_{j \in N_u^k(i)} \text{sim}(i,j)} \quad (33)$$

[147]

Non-Negative Matrix Factorization (NMF)

Matrix factorization is a class of collaborative filtering algorithms used in RS. Matrix factorization algorithms work by decomposing the user-item interaction matrix into the product of two lower dimensionality rectangular matrices. This family of methods became known during the Netflix prize challenge due to its effectiveness. By assigning different regularization weights to the latent factors based on items' popularity and users' activeness, the prediction results can be improved. More specifically, the surprise toolkit algorithm is based on Non-negative Matrix Factorization. The algorithm is very similar to SVD. The prediction $\hat{r}_{u_i}$ is set as:

$$\hat{r}_{ui} = q_i^T p_u \quad (34)$$

Where user and item factors are kept positive, this implementation follows that suggested in [NMF:2014]—a direct application of NMF for dense matrices [NMF_algo]. The optimization procedure is a regularized stochastic gradient descent with a specific choice of step size that ensures the non-negativity of factors, as long that their initial values are also positive. During the different steps of the SGD procedure, the factors for user u and item i are updated as follows:

$$p_{uf} \leftarrow p_{uf} \cdot \frac{\sum_{i \in I_u} q_{if} \cdot r_{ui}}{\sum_{i \in I_u} q_{if} \cdot \hat{r_{ui}} + \lambda_u |I_u| p_{uf}}$$
$$q_{if} \leftarrow q_{if} \cdot \frac{\sum_{u \in U_i} p_{uf} \cdot r_{ui}}{\sum_{u \in U_i} p_{uf} \cdot \hat{r_{ui}} + \lambda_i |U_i| q_{if}} \quad (35)$$

This algorithm has a high dependency on initial values. Baselines are optimized in like manner as in the SVD algorithm. While resulting better accuracy, the biased version seems highly prone to overfitting. A possible solution can be the reduction of the number of factors (or an increased regularization). For those reasons, the final tests were made with the unbiased version.

Singular Value Decomposition Algorithm (SVD)

The famous SVD algorithm, as popularized by Simon Funk during the Netflix Prize. When baselines are not used, this is equivalent to Probabilistic Matrix Factorization. The prediction is calculated from the following function:

$$\hat{r}_{ui} = \mu + b_u + b_i + q_i^T p_u \quad (36)$$

If user u is unknown, then the bias b_u and the factors p_u are assumed to be zero. The same applies to item i with b_i and q_i. To estimate all the unknown, the regularized squared error is minimized:

$$\sum_{r_{ui} \in R_{train}} (r_{ui} - \hat{r}_{ui})^2 + \lambda \left(b_i^2 + b_u^2 + ||q_i||^2 + ||p_u||^2 \right) \quad (37)$$

The minimization is performed by a simple, straightforward stochastic gradient descent:

$$\begin{aligned} b_u &\leftarrow b_u + \gamma(e_{ui} - \lambda b_u) \\ b_i &\leftarrow b_i + \gamma(e_{ui} - \lambda b_i) \\ p_u &\leftarrow p_u + \gamma(e_{ui} \cdot q_i - \lambda p_u) \\ q_i &\leftarrow q_i + \gamma(e_{ui} \cdot p_u - \lambda q_i) \end{aligned} \quad (38)$$

Where $e_{ui} = r_{ui} - \hat{r}_{ui}$These steps are performed over all the ratings of the trainset and repeated **n_epochs** times. Baselines are initialized to 0. [148]

3.2 Evaluated Similarity Measures

The calculation of the similarity between users and elements can be less than straightforward. For the solution of this optimization problem, a plethora of methods has been proposed. For the purposes of this analysis, four methods have been tested

Cosine Similarity

Cosine similarity is a way of measuring the similarity between two non-zero vectors of an inner product space. It is detailed to equal the cosine of the angle between them, which is the same as it is the inner product of the same vectors normalized to both have length 1. The cosine of 0 degrees is 1, and it is less than 1 for any angle in the interval between 0 and π radians. Thus, it is a judgment of orientation and not magnitude. The two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at 90 degrees relative to each other have a similarity of 0, and two vectors opposed have a similarity of -1 and independent of their magnitude.

The cosine similarity is mainly used in positive space, where the outcome is bounded between 0 and1. The name derives from the term direction cosine: in this case, unit vectors are maximally "similar" if they are parallel and entirely "dissimilar" if they are orthogonal (perpendicular). Akin to the cosine, which is unity (maximum value) when the segments subtend a 0 angle and 0 (uncorrelated) when the segments are perpendicular. These constraints apply to any number of dimensions.

The cosine similarity is commonly used in high-dimensional positive spaces. In information retrieval and text mining, for example, each term is notionally assigned to a different dimension, and a vector characterizes a document by where the value in each dimension corresponds to the number of times the term appears in the document. Again, only ordinary users are considered. The cosine similarity is defined as:

$$\text{cosine\_sim}(i,j) = \frac{\sum\limits_{u \in U_{ij}} r_{ui} \cdot r_{uj}}{\sqrt{\sum\limits_{u \in U_{ij}} r_{ui}^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} r_{uj}^2}} \tag{39}$$

Root Mean Square Deviation (MSD)

Root-mean-square deviation (RMSD) or root-mean-square error (RMSE) is a metric for calculating the differences between the values (sample or population values) predicted by a model or an estimator and the observable values. The RMSE is the square root of the second sample moment of the differences between predicted values and values observed or the quadratic mean of these differences. When the calculations are performed over the data sample used for estimation, these deviations are called residuals. When the calculations are computed, out-of-sample are called errors (or prediction errors). RMSE serves to aggregate the magnitudes of the errors in predictions for various data points into a single measure of predictive power. The RMSE measures accuracy to compare forecasting errors of different models for a particular dataset, not between datasets, as it is scale-dependent. RMSE is always non-negative. The MSD between the most common users is calculated, and the similarity function is:

$$\text{msd\_sim}(i,j) = \frac{1}{\text{msd}(i,j) + 1} \tag{40}$$

Pearson Coefficient

The Pearson correlation coefficient is a measure of linear correlation between two sets of data. It is the covariance of two variables, divided by the product of their standard deviations. Thus, it is a normalized measurement of the covariance, resulting in a value between −1 and 1. Though, the measure can only reflect a linear correlation of variables and ignores many other types of relationship or correlation. In this example, only ordinary users are considered. The Pearson correlation coefficient can be examined as mean-centered cosine similarity and is defined as:

$$\text{pearson\_sim}(i,j) = \frac{\sum\limits_{u \in U_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum\limits_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} (r_{uj} - \mu_j)^2}} \tag{41}$$

Pearson Baseline Coefficient

Compute the (shrunk) Pearson correlation coefficient between all pairs of users (or items), using baselines for centering instead of means. The shrinkage parameter assists to avoid overfitting when only a few ratings are available. The Pearson-baseline correlation coefficient is defined as:

$$\text{pearson\_baseline\_sim}(i,j) = \hat{\rho}_{ij} = \frac{\sum\limits_{u \in U_{ij}} (r_{ui} - b_{ui}) \cdot (r_{uj} - b_{uj})}{\sqrt{\sum\limits_{u \in U_{ij}} (r_{ui} - b_{ui})^2} \cdot \sqrt{\sum\limits_{u \in U_{ij}} (r_{uj} - b_{uj})^2}} \quad (42)$$

The shrunk Pearson-baseline correlation coefficient is then described as:

$$\text{pearson\_baseline\_shrunk\_sim}(i,j) = \frac{|U_{ij}| - 1}{|U_{ij}| - 1 + \text{shrinkage}} \cdot \hat{\rho}_{ij} \quad (43)$$

[149]

3.3 Hardware Resources

The system used was running Windows 10 Pro, the processor was an AMD Ryzen 9 3900X counting 12 cores running at 3.8 GHz, the system had 32GB of DDR4 memory at a frequency of 1600MHz.

3.4 Software Resources

Programming Language

The language I chose was Python3. Python was something I did not possess in my programming repertoire and, at the same time, a skill I wanted to cultivate. The Python language offered many advantages. Some are its code readability, a significant advantage as I was organizing the biggest programming project of my academic path, its language constructs, and its object-oriented approach, its features of dynamic typing and garbage-collection. Two more and perhaps the most valued were its comprehensive libraries and its numerous and highly active community, a helpful hand when everything goes as planned, and a lifesaver when everything is falling apart.

Integrated Development Environment (IDE)

The platform below the IDE was Anaconda, a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, and more), aiming to simplify package management and deployment. For the development of the project, I chose the SPYDER IDE. Spyder is an open-source scientific environment written in Python and designed for scientists, engineers, and data analysts. It encompasses a unique blend of the advanced editing, analysis, debugging, and profiling functionality of a comprehensive development tool and also the data exploration, interactive execution, deep inspection, and visualization capabilities of a scientific package. More specifically, the programs were run in the 5.0.0 patch.

3.5 Python Modules, Libraries, and External Packages

NumPy: a library for adding support for large, multi-dimensional arrays and matrices, along with an extensive collection of high-level mathematical functions to run on these arrays. NumPy targets the CPython reference implementation of Python, which is a non-optimizing bytecode interpreter. Mathematical algorithms are written for this version of Python often run more sluggish than compiled equivalents. NumPy addresses the slowness problem partly by offering multidimensional arrays and functions and operators that operate efficiently on arrays, requiring rewriting some code, primarily inner loops, using NumPy. Many modern large-scale scientific computing applications have requirements that exceed the potential of the NumPy arrays. For example, NumPy arrays are often loaded into a computer's memory, which might have insufficient capacity to analyze vast datasets. Moreover, NumPy operations are executed on a single CPU.[144]

Matplotlib: a plotting library for Python and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK. A procedural "pylab" interface is also based on a state machine (like OpenGL), designed to resemble MATLAB, though its use is discouraged closely.[145]

Pandas: a software library written for Python, used for data manipulation and analysis. It provides data structures and operations for manipulating numerical tables and time series. It was released under the three-clause BSD license. Pandas are mainly used for data analysis. Pandas allow importing data from various file formats such as comma-separated-values, JSON, SQL, and Microsoft Excel. Pandas do allow various data manipulation operations such as selecting, merging, reshaping, data cleaning, and data wrangling features.[146]

Surprise: a python scikit for building and analyzing recommender systems that deal with explicit rating data. Surprise provides various ready-to-use prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (SVD, PMF, SVD++, NMF), and many others. Also, various similarity measures (cosine, MSD, Pearson…).[147][148][149]

CSV module: The CSV module implements classes to be able to read and write tabular data in CSV format. It allows programmers to write data in a format preferred by Excel or read data from a file generated by Excel without knowing the precise details of the CSV format used by Excel. Programmers can also describe the CSV formats understood by other applications or define their special-purpose CSV formats.

Collections module: This module implements specialized container datatypes providing alternatives to Python's general-purpose built-in containers, dict, list, set, and tuple.

Statistics module: This module offers functions for calculating mathematical statistics of numeric (Real-valued) data. It was not intended to be a competitor to third-party libraries such as SciPy, NumPy, or proprietary full-featured statistics packages aimed for professional statistical calculations such as Minitab, SAS, and Matlab. Instead, it is aimed at the level of graphing and scientific calculators.

OS module: This module provides a portable way of using operating system-dependent functionality. Helpful if the goals are reading or writing a file, manipulating paths, or reading lines in all the files on the command.

Time module: This module provides various time-related functions. They are mainly used to calculate the runtime, which is highly important, especially on long runtime scripts or multiple consecutive runs.

3.5 The 100K MOVIELENS dataset

The GroupLens Research Project collected the MovieLens datasets. The GroupLens Research Project is a research group based in the Department of Computer Science and Engineering at the University of Minnesota. Members of the GroupLens Research Project are involved in many research projects related to the fields of information filtering,

This dataset consists of:

- 100,000 ratings from one to five from 943 users on 1682 movies
- Every one of the 943 users has rated at least 20 movies
- Simple demographic info for the users such as age, gender, occupation, zip

The data was collected through the MovieLens website (movielens.umn.edu) during the seven months from September 19th, 1997, through April 22nd, 1998. This data has been cleaned up – users who had less than 20 ratings or did not have complete demographic information were removed from this data set. Detailed descriptions of the data file can be found at the end of this file. [150][151]

Chapter 4: Recommendation System Test Results

This chapter focuses on presenting the results the 12 algorithms gave and not evaluating or seeking to decipher the meaning behind the number as this is the task of Chapters 5 and 6. Although through this chapter, it is becoming known the structure, operation of the algorithms, and data collection sequence. Also, the movie dataset 100K is briefly analyzed in the first pages, and more information about its size and characteristics is shown.

4.1 Dataset Statistical Analysis

All the algorithms start the same way. The first step is to insert the data from the CSV file to a pandas dataframe called rankings. Then, the following steps are the interpretation of the statistical characteristics of the dataframe.

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931
..	...	...	...	...
95	1	1445	3.0	964984112
96	1	1473	4.0	964980875
97	1	1500	4.0	964980985
98	1	1517	5.0	964981107
99	1	1552	4.0	964982620

[100 rows x 4 columns]

Figure 2.1: The structure of the rankings.csv contains the columns of user and movie ids, movie rating, and timestamp.

As seen in Figure 2.1, the dataframe contains the user and movie id, the rating the user has provided for that movie, and timestamps. The following procedures work upon this single dataframe. The first step is the calculation of the number of users, movies, and ratings. Also, from this dataframe is calculated the sparsity of the rating matrix. The rating matrix has the number of users and the number of movies for its axis, and inside the matrixs' cells are saved the movie ratings. Thus, each user who has not rated a movie leaves an empty space.

```
Total number of Users: 610
Total number of Movies: 9724
Total number of ratings provided:  100836
Total number ratings provided & not specified: 5931640
Rating Matrix is 98.3% sparse
Sparsity:  0.9830003169443864
```

Figure 2.2: The algorithm calculates the number of different users, movies, how many ratings have been submitted, how many empty rating spaces exist, and the percentage of them over the max possible ratings.

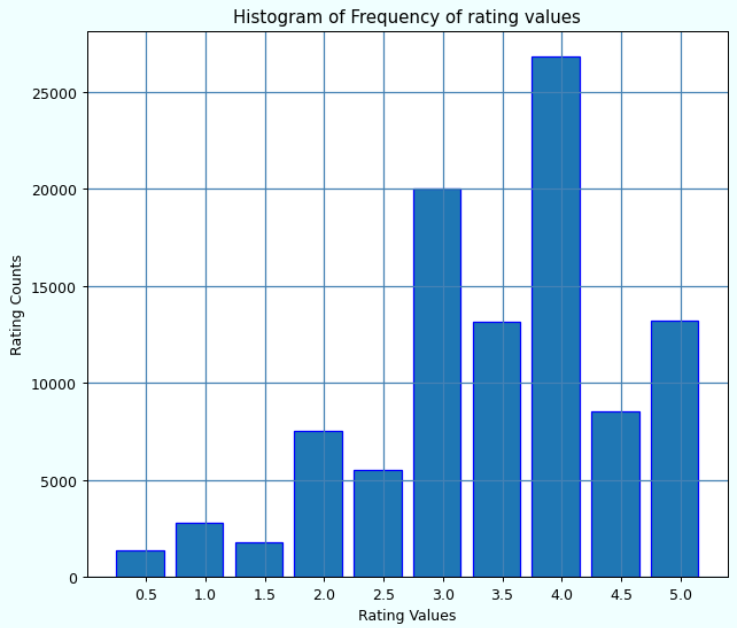


Figure 2.3: Visualization of every movie group based on rating on an increment of 0.5.

Figure 2.3 shows that most movies have a user rating between 3 and 4 while very few have a rating smaller than 2. Also, from Figure 2.4, one can say it results from the exceedingly high sparsity of the dataframe as the vast majority of movies lack user reviews. Only a small percentage has more than 50, meaning 1 out of 12 users has rated them. As a result, a problem is the credibility of a mean rating for a movie as the outlier review will have a more substantial influence.

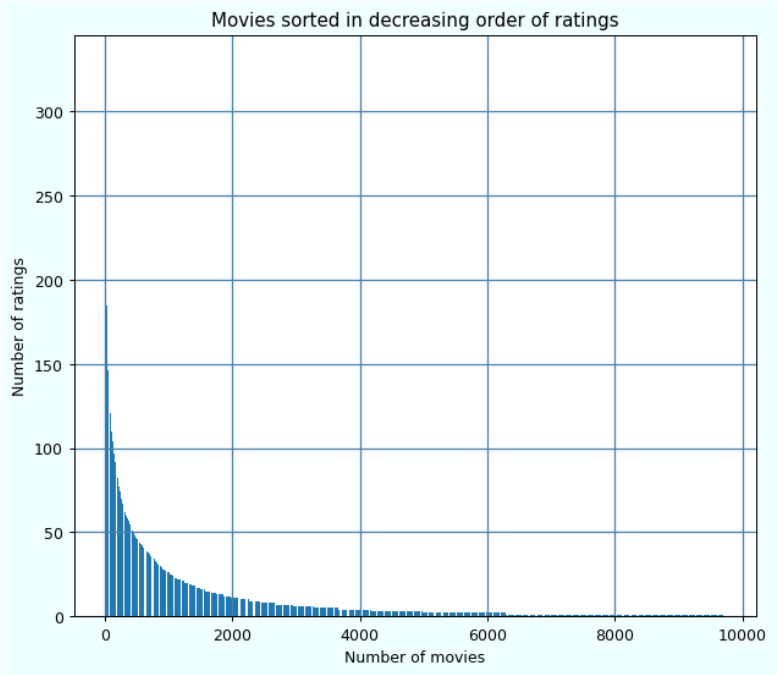


Figure 2.4: Every movie shown in decreasing number of ratings.

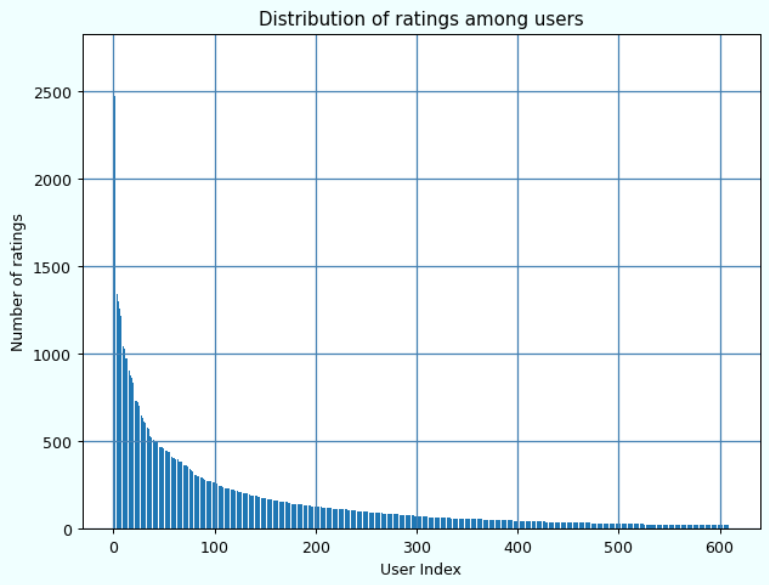


Figure 2.5: Every user shown in decreasing number of ratings submitted.

Some interesting conclusions from Figure 2.5 are that only a few users have a habit of rating every movie they see. However, a significant percentage leaves reviews on a few of the movies they have seen, one can hypothesize on those that have left a great or a dreadful impression. Also, more than half of the users have less than 100 reviews. Furthermore, one great advantage from this dataset is that every user has rated some movies making it easier for the algorithm to give suggestions.

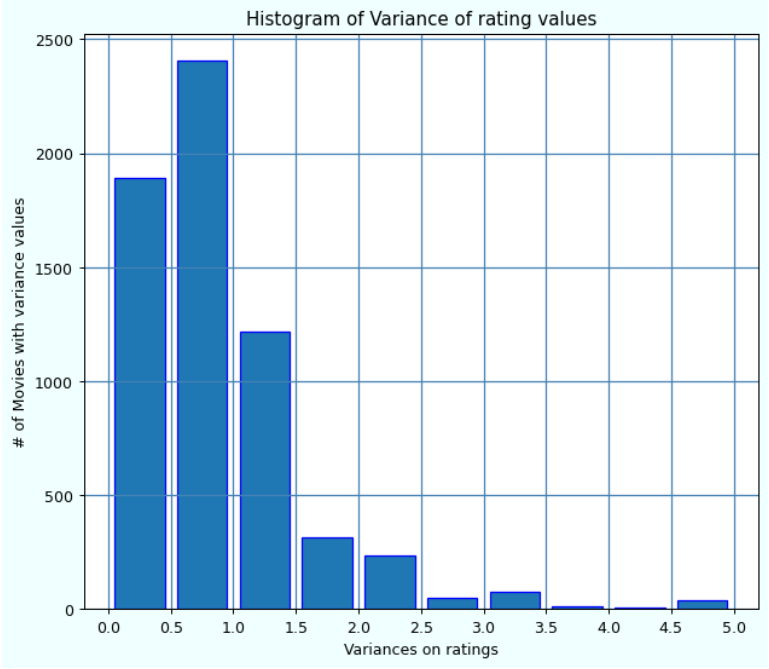


Figure 2.6: For each movie, it is shown the difference between its best and worst rating.

Another helpful element is the generally very slight variance between different user ratings of the same movie, which means that different users more or less have the same impression of the same movie. Thus, in theory, the suggestions will have a greater recall, as similar users preferring, for example, the same kind of movie and the suggested group was valued by the user cluster will probably be valued the same amount by the lone user.

If a method is better than another, the comparison must be made between the same constants, the same dataset, and the same number of neighbors. As the best number of neighbors for each method in this dataset is unknown, and overfitting may falsify the final results, the algorithms will run 74 times. Each for the number of neighbors from 2 to 75. The Root Mean Square Error (RMSE) and Mean Absolute Error (MAE) is calculated for every fold.

4.2.1 kNN With Means (Cosine)

```
2  neigbours

Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Computing the cosine similarity matrix...
Done computing similarity matrix.
Evaluating RMSE, MAE of algorithm KNNWithMeans on 10 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Fold 6  Fold 7  Fold 8  Fold 9  Fold 10 Mean    Std
RMSE (testset)    1.0220  1.0189  1.0234  1.0021  1.0267  1.0153  1.0122  1.0269  1.0002  1.0032  1.0151  0.0097
MAE (testset)     0.7845  0.7876  0.7906  0.7732  0.7956  0.7839  0.7802  0.7901  0.7749  0.7729  0.7833  0.0075
Fit time          0.44    0.46    0.45    0.46    0.48    0.46    0.47    0.48    0.47    0.47    0.46    0.01
Test time         0.31    0.35    0.33    0.40    0.34    0.33    0.32    0.32    0.33    0.33    0.34    0.02
```

Figure 3.1.1: Printed the number of neighbors imported for the specific execution, the RMSE, MAE, and other elements for each fold. Those are also averaged at the end.

At the end of the tenth fold, a mean of the folds for the RMSE and MAE is calculated. Those will be the RMSE and MAE of the algorithm for that number of neighbors. For example, in Figure 3.1.1, the kNN algorithm with the cosine similarity for two neighbors has an RMSE equal to 1.01151.

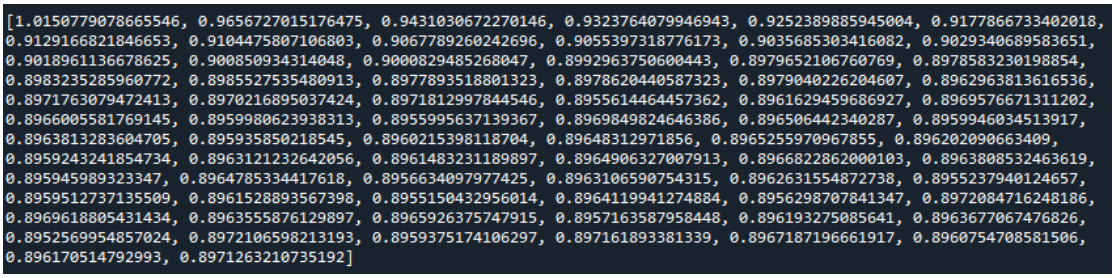


Figure 3.1.2: kNN Cosine RMSE averages for 74 executions. From 2 to 75 neighbors.

At the end of all 74 executions and the calculation of the RMSE, MAE means for the different number of neighbors the algorithm prints the results as seen in Figures 3.1.2 and 3.1.3.

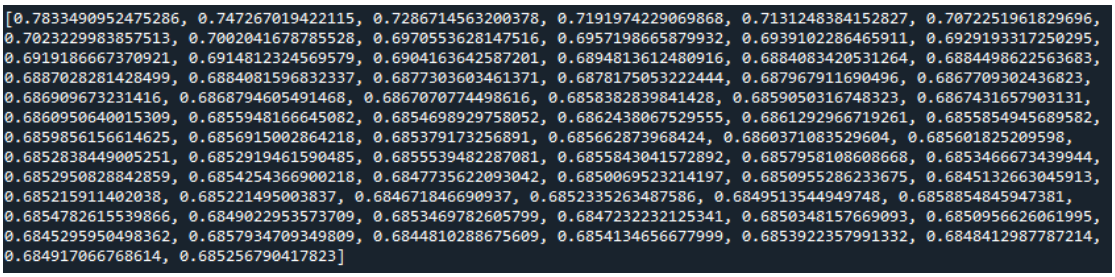


Figure 3.1.3: kNN Cosine MAE averages for 74 executions. From 2 to 75 neighbors.

Then the algorithm tests the results, calculating the mean of means, meaning the mean of the method overall, and locating the best results. For example, Figure 3.1.4. shows that the mean of the results for the RMSE in Figure 3.1.2 is 0.91922 and that the minor error has been located for 68 neighbors, and it was 0.89525.

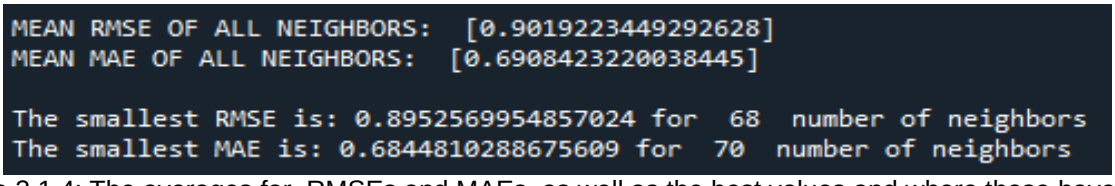


Figure 3.1.4: The averages for RMSEs and MAEs, as well as the best values and where those have been observed.

The results from Figures 3.1.2 and 3.1.3 are then visualized in the following graph, showing the gradual improvement of the method as the number of neighbors changes.

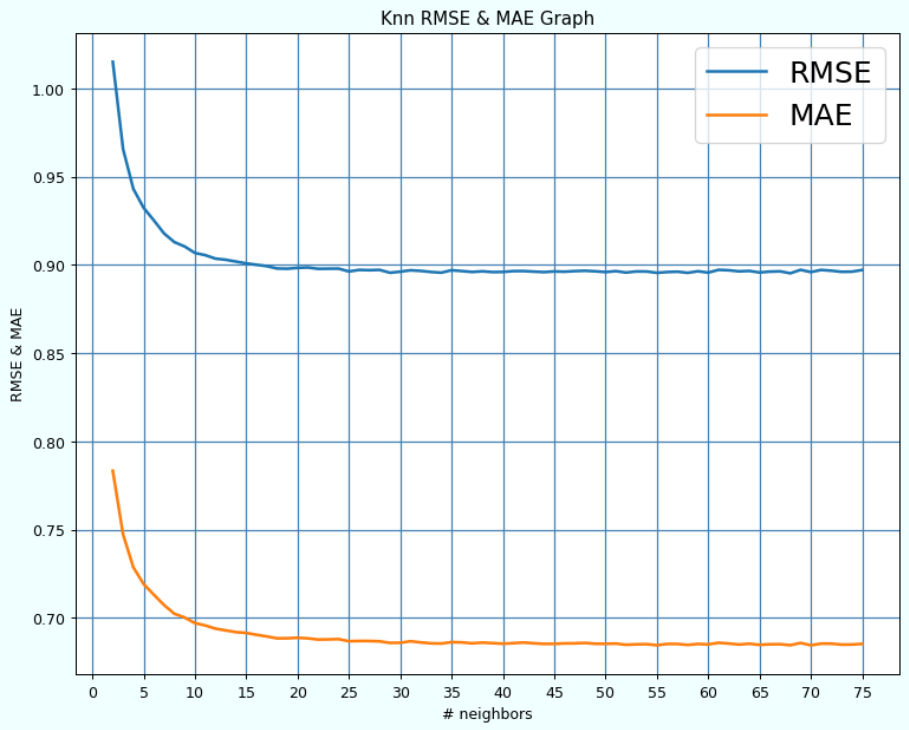


Figure 3.1.5: kNN Cosine RMSE, MAE graph for the different number of neighbors.

Some other essential metrics are Precision and Recall. Those are also calculated at the end for each neighbor number. As before, the results are printed in groups. First, the Precision means, and then the Recall means.

```
Precision
[0.5294637258965789, 0.5075237859594321, 0.4963858397556485, 0.4845684298890873, 0.47662534474870705, 0.4734621303950468,
0.46178264414327136, 0.4684791415106525, 0.46342647268095954, 0.45836224664451464, 0.4597301164059818, 0.454047872500069,
0.45410223906669234, 0.45607351393800216, 0.4575533846969252, 0.45294370914193083, 0.4546473132160559, 0.4536992932594034,
0.4585846574235834, 0.4550439014685133, 0.4577579547213526, 0.45472949362668313, 0.4585756494428825, 0.4572234231683027,
0.4574924275268912, 0.4599755019213422, 0.4589868206679486, 0.458845718016932, 0.4587794966517026, 0.4591550177545042,
0.4590370627941923, 0.4567946009908773, 0.4630288510853472, 0.4578511892528587, 0.4592929374566947, 0.46263804152113036,
0.4607445230253811, 0.4611302274738728, 0.463573525235789, 0.4586368391007003, 0.4594738019163456, 0.46180412550884187,
0.4641992561232522, 0.46385141936591606, 0.46304373800049453, 0.4651676193960876, 0.4628253040110156, 0.4662862186103932,
0.46607582243071005, 0.46563092501726533, 0.463615254759788, 0.4722718966478098, 0.4683764356393313, 0.46746090789081435,
0.4657936832544963, 0.4669897124731719, 0.4662860385843387, 0.46465190638329223, 0.4675816060638184, 0.4710186426502837,
0.47025664057022576, 0.46484579224396205, 0.4748758045800958, 0.4656492704906473, 0.47214241245006355, 0.4705909705566338,
0.4691260987374394, 0.4675594294250008, 0.4724460602290278, 0.47176065974488673, 0.4696176287968066, 0.469153662726677,
0.4727479158505064, 0.4706035422199878]
```

Figure 3.1.6: kNN Cosine Precision averages for 74 executions. From 2 to 75 neighbors.

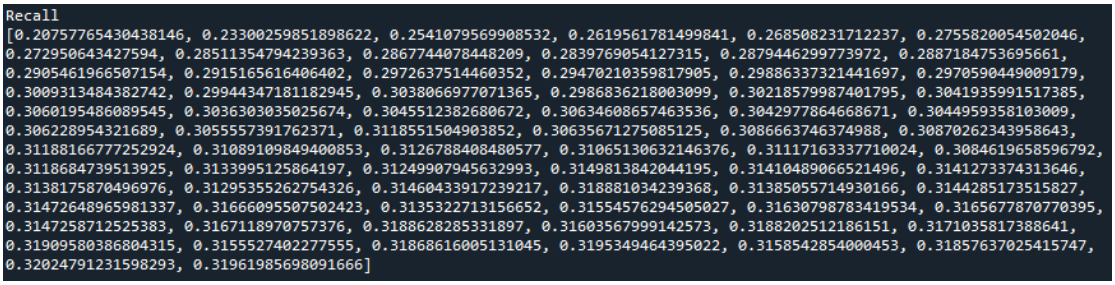


Figure 3.1.7: kNN Cosine Recall averages for 74 executions. From 2 to 75 neighbors.

Then the algorithm tests the results, calculating the mean of means, meaning the mean of the method overall, and locating the best results. For example, Figure 3.1.8. shows that the mean of the results for the Precision in Figure 3.1.6 is 0.4656 and that the highest Precision has been located for 2 neighbors, and it was 0.52946.

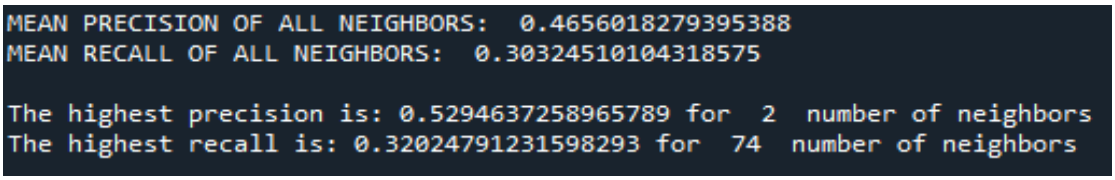


Figure 3.1.8: The averages for Precision and Recall, the best values, and where those have been observed.

The results from Figures 3.1.6 and 3.1.7 are then visualized in the following graph, showing the gradual improvement or deterioration of the method as the number of neighbors changes.

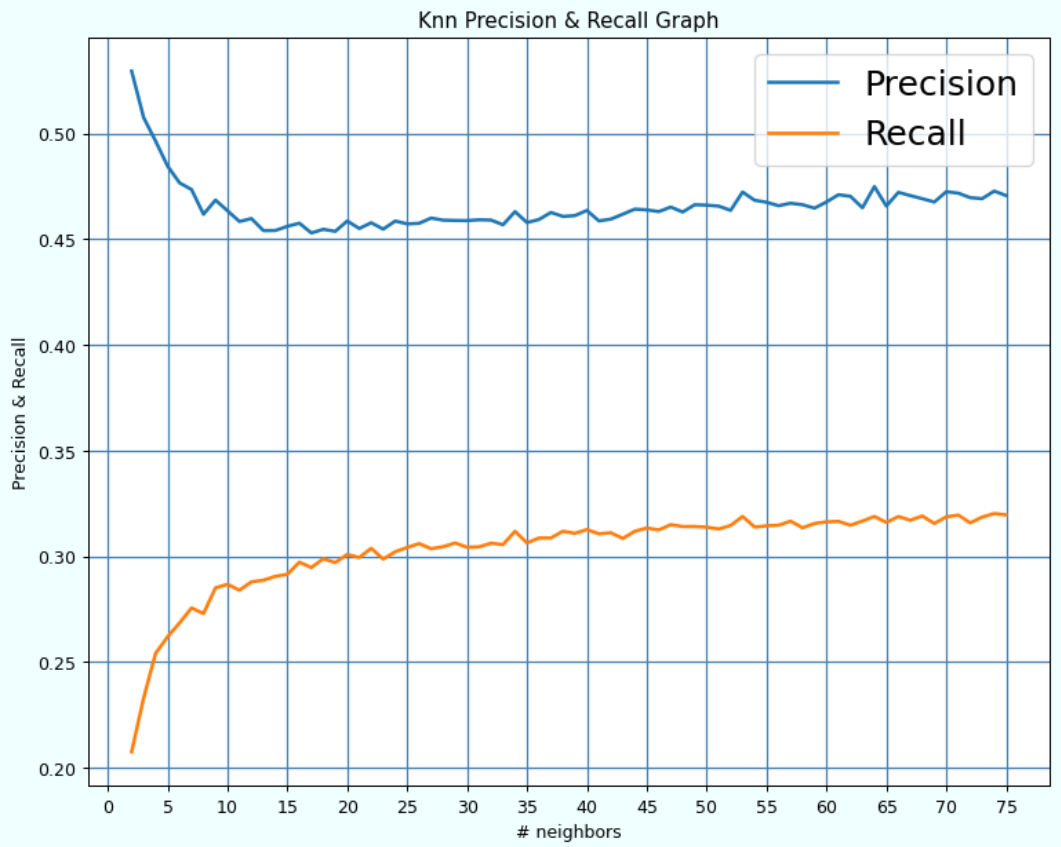


Figure 3.1.9: kNN Cosine Precision and Recall graph for the different number of neighbors.

All 12 algorithms follow the same structure as presented for the kNN Cosine Similarity. For that reason, the presentation will be confined to the results for the subsequent algorithms and not the method.

4.2.2 kNN (MSD)

As in the previous method, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The kNN algorithm is the same, with a single change in the similarity calculation method. In this part, the Mean Square Difference Similarity is examined. Each completion for a different number of neighbors produces a different mean. The RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 3.2.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8973217563393401]
MEAN MAE OF ALL NEIGHBORS:  [0.6862626250302634]

The smallest RMSE is: 0.890760328124021 for  32  number of neighbors
The smallest MAE is: 0.6805569349852826 for  65  number of neighbors
```

Figure 3.2.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 3.2.2, in order to visualize the changes based on the difference in the number of neighbors.

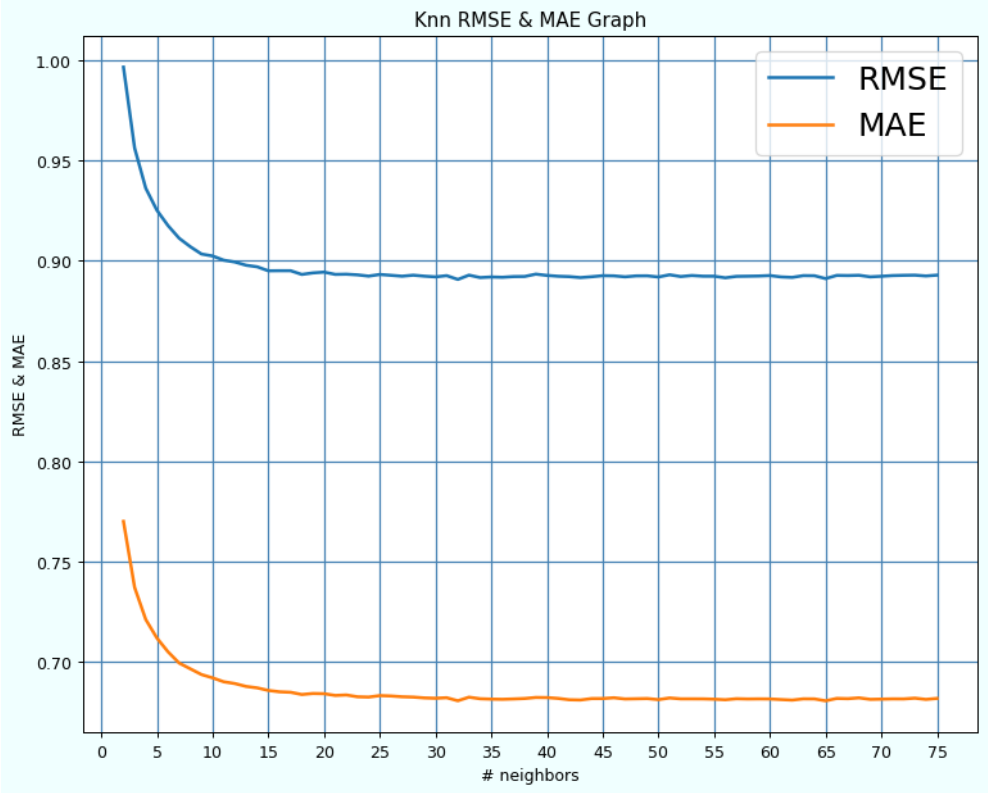


Figure 3.2.2: kNN MSD RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 3.2.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.4901972015465206
MEAN RECALL OF ALL NEIGHBORS:  0.3182259451693848

The highest precision is: 0.551296413242268 for  2  number of neighbors
The highest recall is: 0.3326411580747258 for  72  number of neighbors
```

Figure 3.2.3: The averages of Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 3.2.4, in order to visualize the changes based on the difference in the number of neighbors.

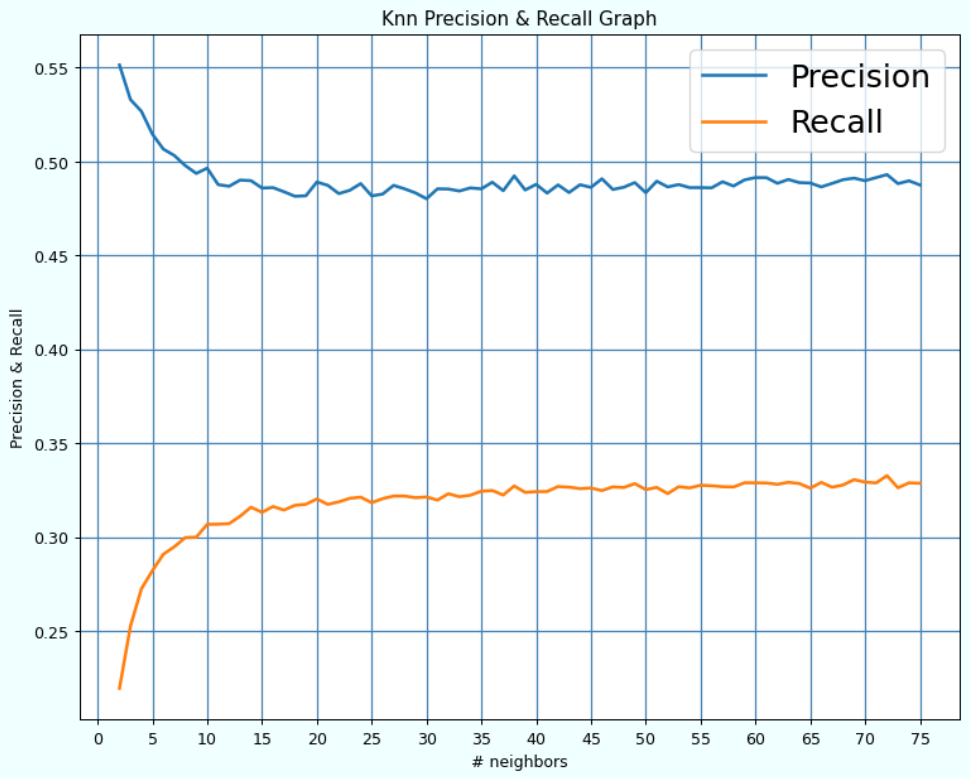


Figure 3.2.4: kNN MSD Precision and Recall graph for the different number of neighbors.

4.2.3 kNN (Pearson Coefficient)

As in the previous methods, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The kNN algorithm is the same, with a single change in the similarity calculation method. In this part, the Pearson Coefficient is examined. Each completion for a different number of neighbors produces a different mean. The RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 3.3.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8961451843822313]
MEAN MAE OF ALL NEIGHBORS:  [0.6831128304634516]

The smallest RMSE is: 0.8886433252427917 for  51  number of neighbors
The smallest MAE is: 0.6766907485435686 for  42  number of neighbors
```

Figure 3.3.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 3.3.2, in order to visualize the changes based on the difference in the number of neighbors.

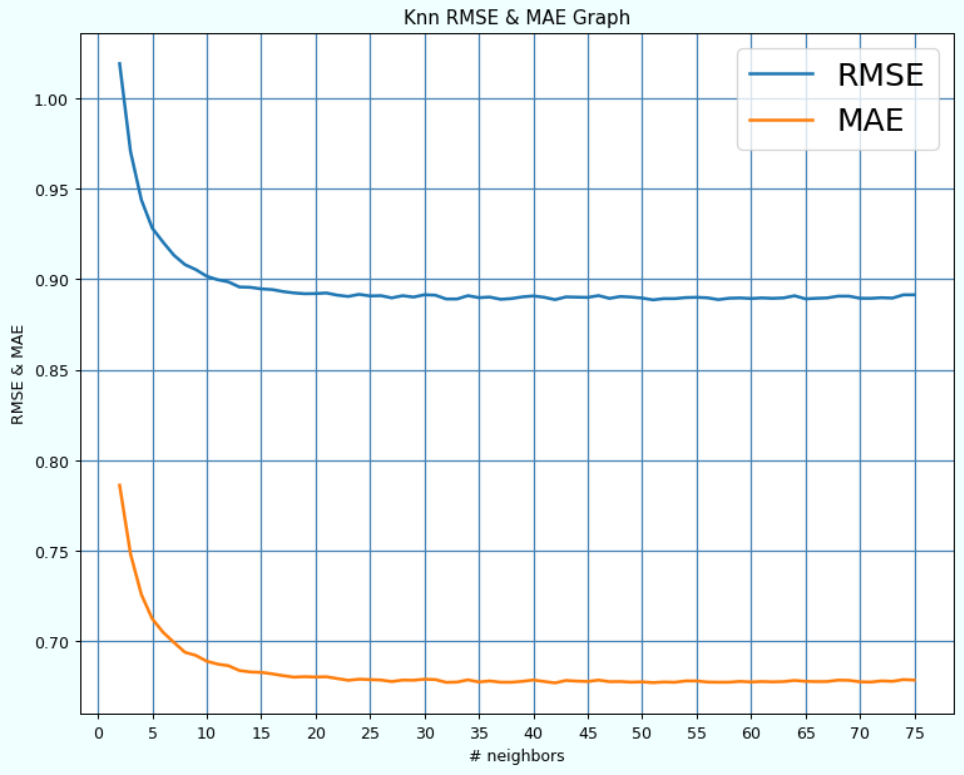


Figure 3.3.2: kNN Pearson RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 3.3.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5187555831003041
MEAN RECALL OF ALL NEIGHBORS:  0.3486455399147711

The highest precision is: 0.5518055290728936 for  2  number of neighbors
The highest recall is: 0.36086296152915526 for  57  number of neighbors
```

Figure 3.3.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 3.3.4, in order to visualize the changes based on the difference in the number of neighbors.

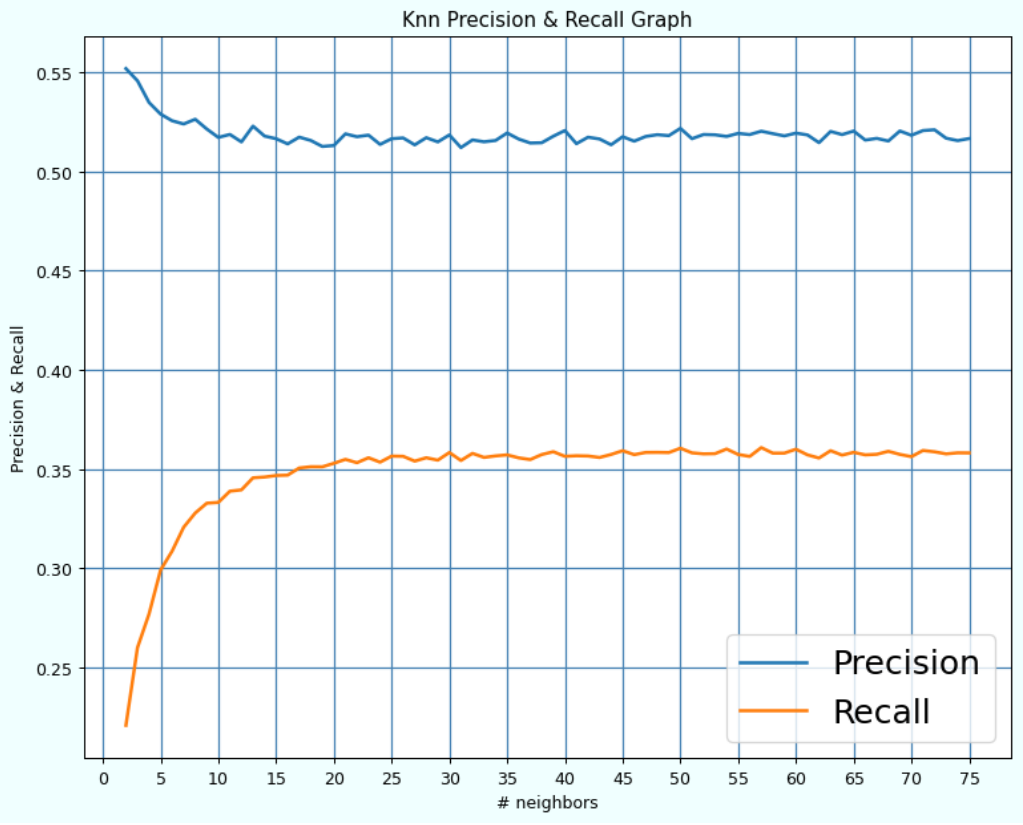


Figure 3.3.4: kNN Pearson Precision and Recall graph for the different number of neighbors.

4.2.4 kNN (Pearson Baseline Coefficient)

As in the previous methods, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The kNN algorithm is the same, with a single change in the similarity calculation method. In this part, the Pearson Baseline Coefficient is examined. Each completion for a different number of neighbors produces a different mean. The RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 3.4.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8911180237479102]
MEAN MAE OF ALL NEIGHBORS:  [0.6755232616725099]

The smallest RMSE is: 0.8860932810261504 for  25  number of neighbors
The smallest MAE is: 0.6712288653035409 for  25  number of neighbors
```

Figure 3.4.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 3.4.2, in order to visualize the changes based on the difference in the number of neighbors.

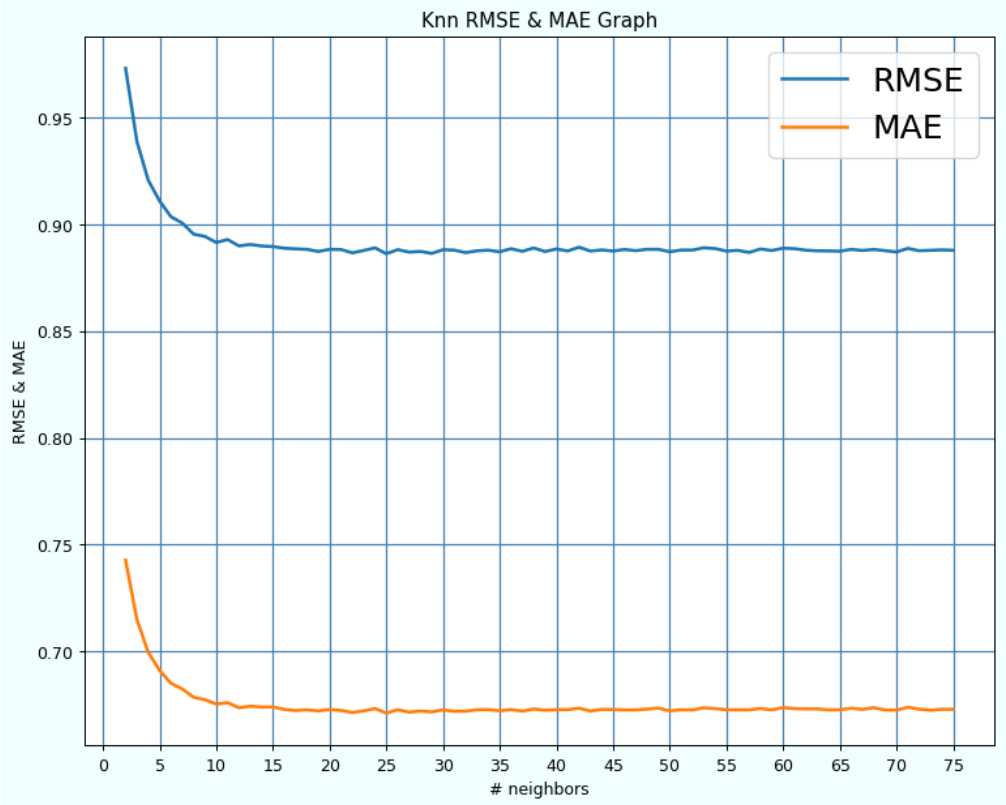


Figure 3.4.2: kNN Pearson Baseline RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 3.4.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5626638734655804
MEAN RECALL OF ALL NEIGHBORS:  0.38411471343152426

The highest precision is: 0.6244077845592485 for  2  number of neighbors
The highest recall is: 0.4032560314526135 for  15  number of neighbors
```

Figure 3.4.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 3.4.4, in order to visualize the changes based on the difference in the number of neighbors.

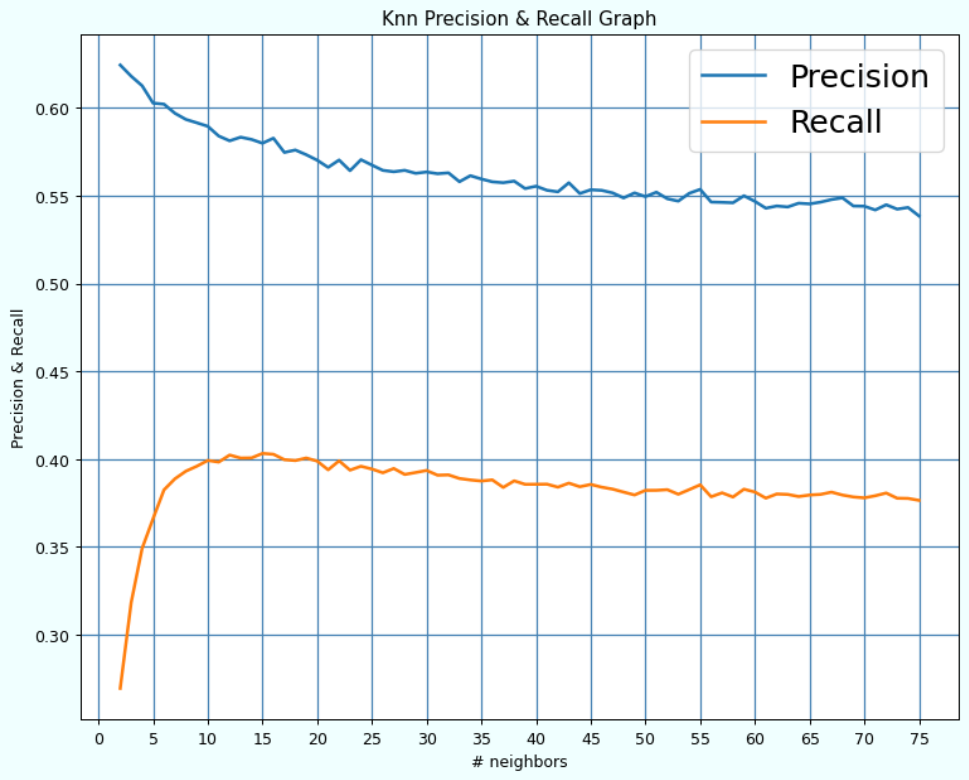


Figure 3.4.4: kNN Pearson Baseline Precision and Recall graph for the different number of neighbors.

4.3.1 kNN Z-Score Normalisation (Cosine Coefficient)

As in the previous methods, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The algorithm used is the kNN with Z-Score normalization, further analyzed in Chapter 3, combined with the Cosine Coefficient similarity calculation method. Each completion for a different number of neighbors produces a different mean. Finally, the RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 4.1.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.9018134826007166]
MEAN MAE OF ALL NEIGHBORS:  [0.6858520089195157]

The smallest RMSE is: 0.8930653421043122 for  51  number of neighbors
The smallest MAE is: 0.6779558117091132 for  61  number of neighbors
```

Figure 4.1.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 4.1.2, in order to visualize the changes based on the difference in the number of neighbors.

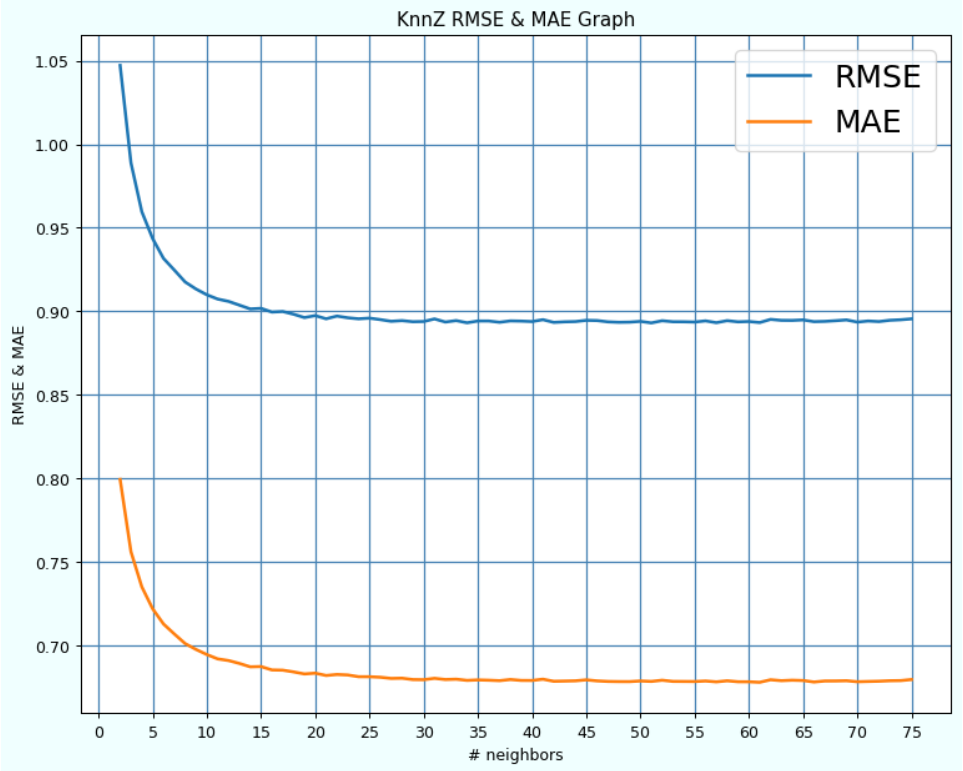


Figure 4.1.2: kNNZ Cosine RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 4.1.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.4854975990235108
MEAN RECALL OF ALL NEIGHBORS:  0.3117692098444642

The highest precision is: 0.5431514865751629 for  2  number of neighbors
The highest recall is: 0.3277966925723634 for  71  number of neighbors
```

Figure 4.1.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 4.1.4, in order to visualize the changes based on the difference in the number of neighbors.

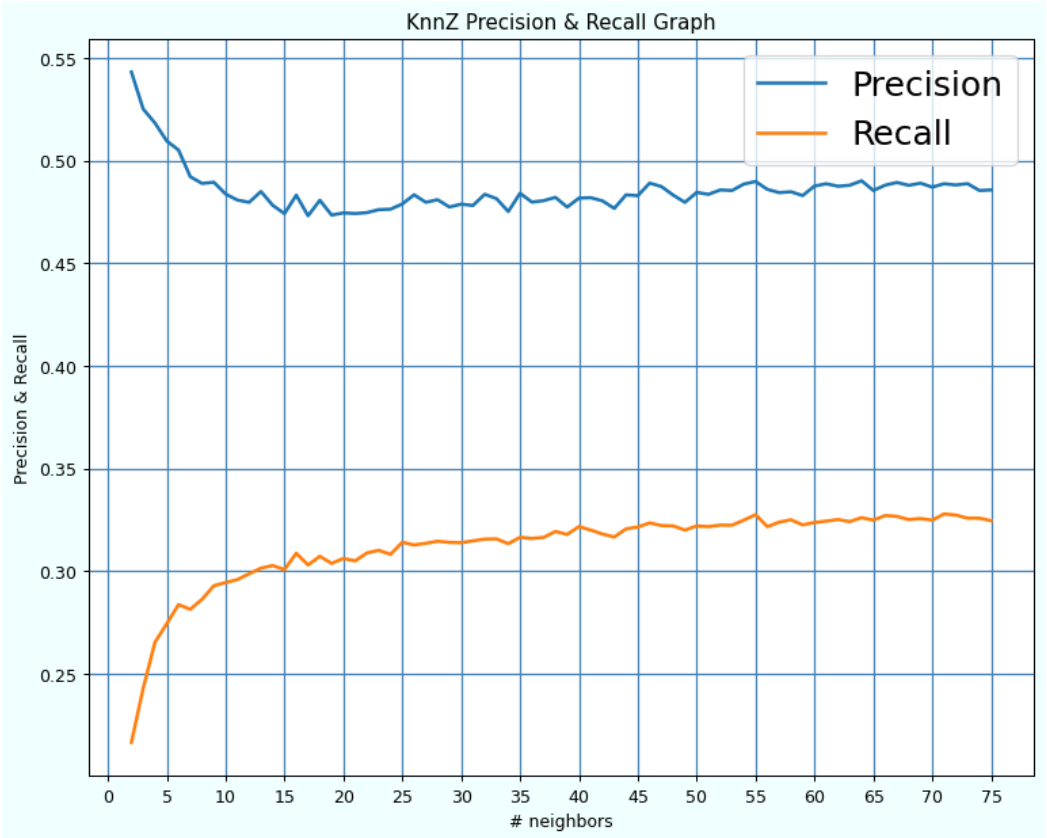


Figure 4.1.4: kNNZ Cosine Precision and Recall graph for the different number of neighbors.

4.3.2 kNN Z-Score Normalisation (MSD)

As in the previous method, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The kNNZ algorithm is the same, with a single change in the similarity calculation method. In this part, the Mean Square Difference Similarity is examined. Each completion for a different number of neighbors produces a different mean. The RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 4.2.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8987336010948734]
MEAN MAE OF ALL NEIGHBORS:  [0.6824722498467017]

The smallest RMSE is: 0.8909029983577772 for  34  number of neighbors
The smallest MAE is: 0.6755607879508192 for  41  number of neighbors
```

Figure 4.2.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 4.2.2, in order to visualize the changes based on the difference in the number of neighbors.

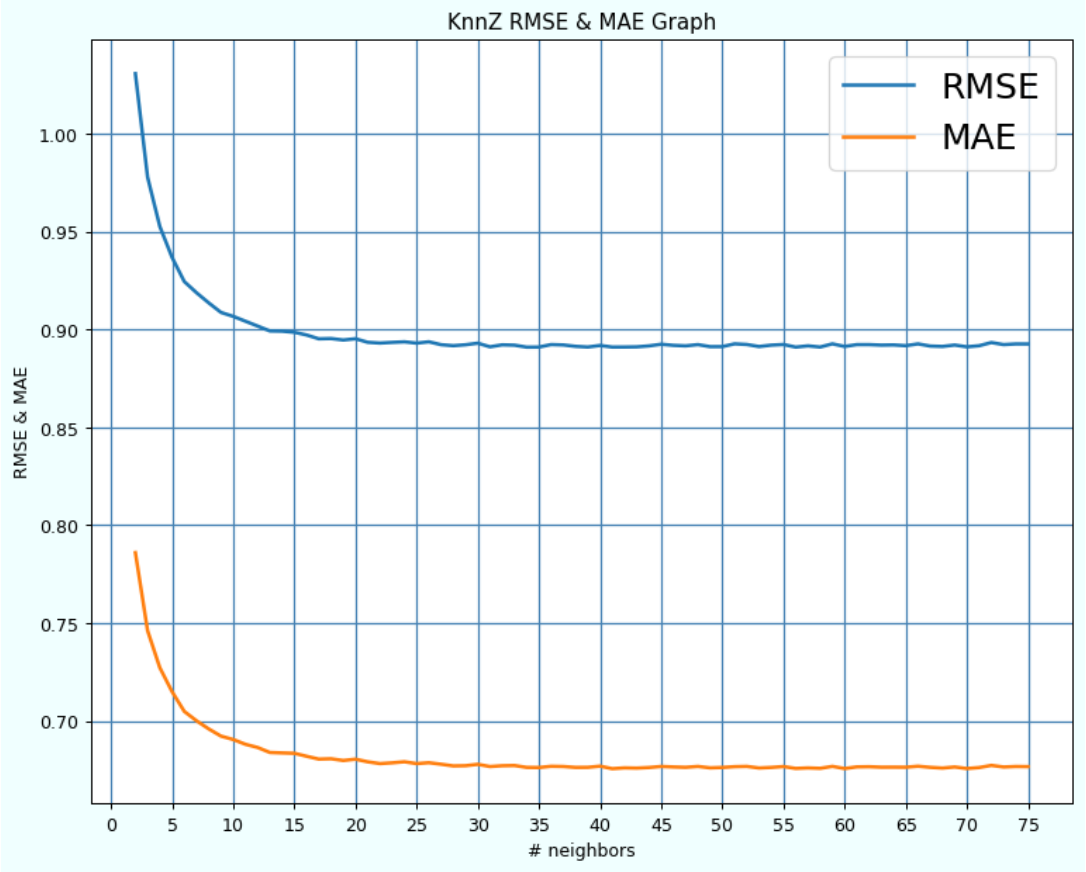


Figure 4.2.2: kNNZ MSD RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 4.2.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5118799185287803
MEAN RECALL OF ALL NEIGHBORS:  0.32854726633099735

The highest precision is: 0.5549588024078468 for  2  number of neighbors
The highest recall is: 0.34057350822004817 for  75  number of neighbors
```

Figure 4.2.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 4.2.4, in order to visualize the changes based on the difference in the number of neighbors.

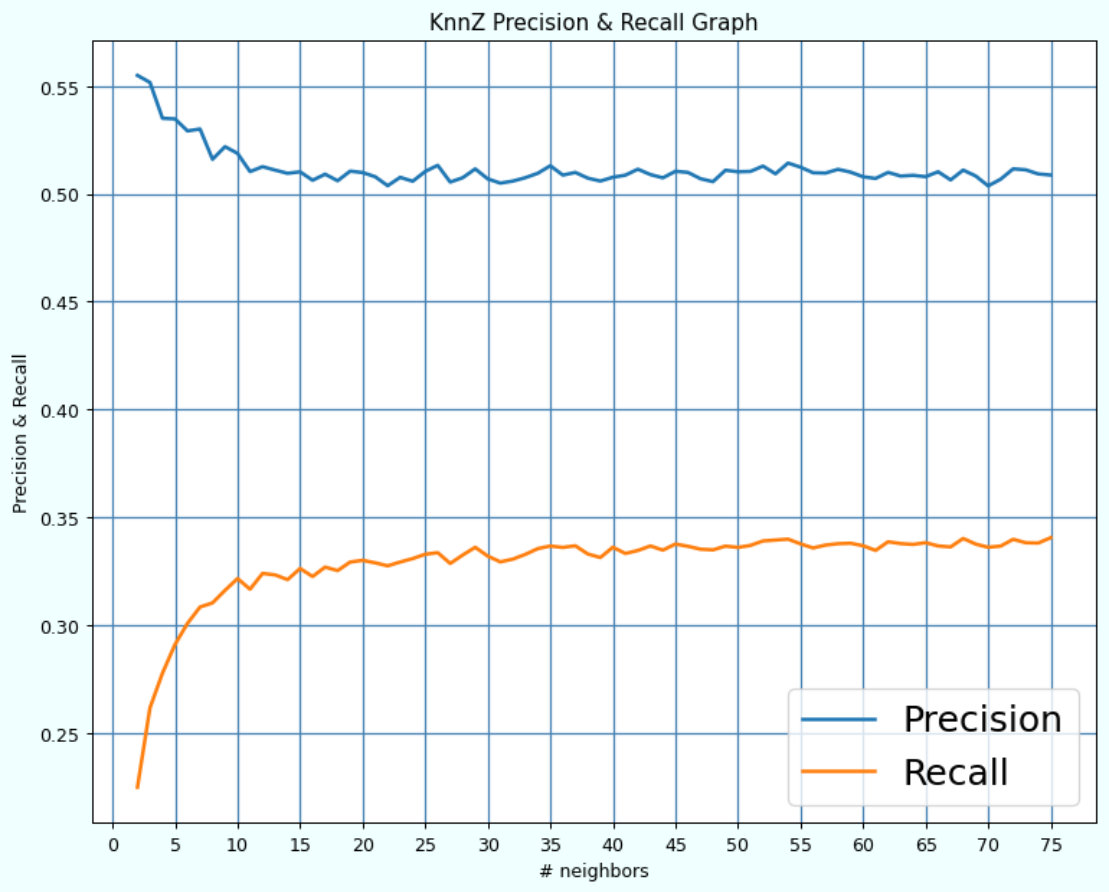


Figure 4.2.4: kNNZ MSD Precision and Recall graph for the different number of neighbors.

4.3.3 kNN Z-Score Normalization (Pearson Coefficient)

As in the previous method, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The kNNZ algorithm is the same, with a single change in the similarity calculation method. In this part, the Pearson Coefficient is examined. Each completion for a different number of neighbors produces a different mean. The RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 4.3.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8929861155460977]
MEAN MAE OF ALL NEIGHBORS:  [0.674965891977546]

The smallest RMSE is: 0.8859308142886759 for  31  number of neighbors
The smallest MAE is: 0.6691910432207575 for  46  number of neighbors
```

Figure 4.3.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 4.3.2, in order to visualize the changes based on the difference in the number of neighbors.

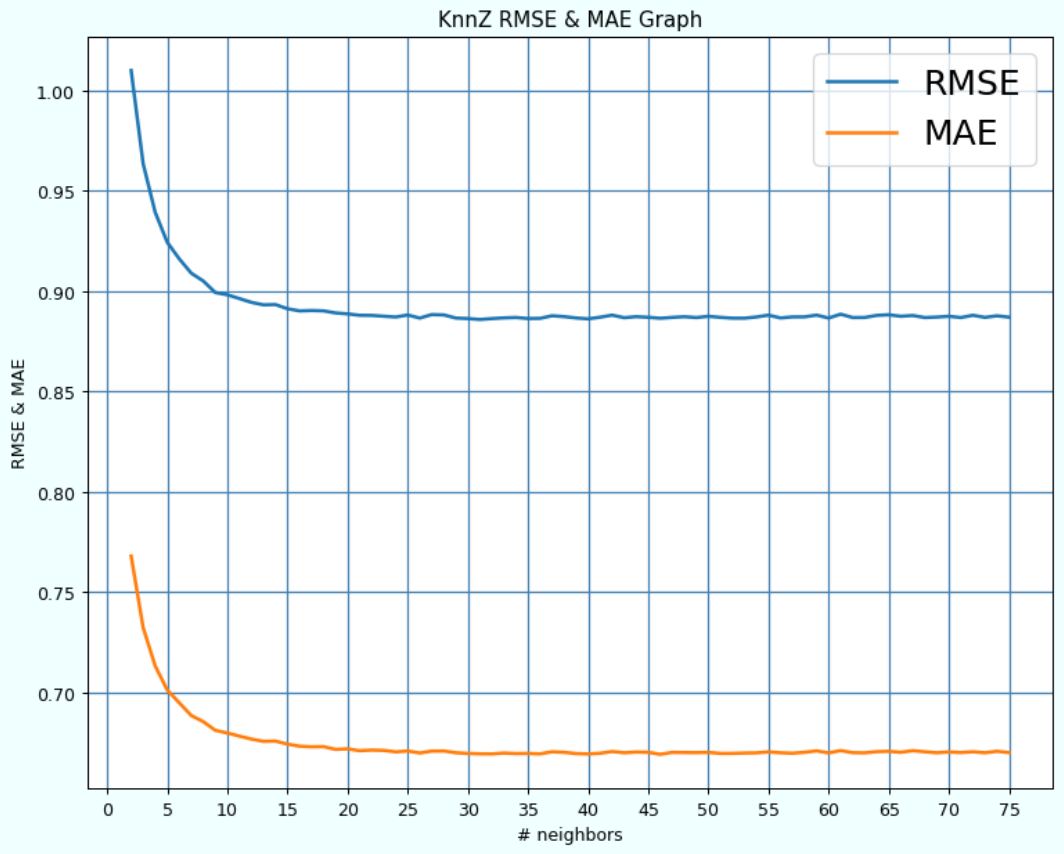


Figure 4.3.2: kNNZ Pearson RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 4.3.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5223480636836835
MEAN RECALL OF ALL NEIGHBORS:  0.34956301971632914

The highest precision is: 0.5419513083417127 for  2  number of neighbors
The highest recall is: 0.36046511098344197 for  43  number of neighbors
```

Figure 4.3.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 4.3.4, in order to visualize the changes based on the difference in the number of neighbors.

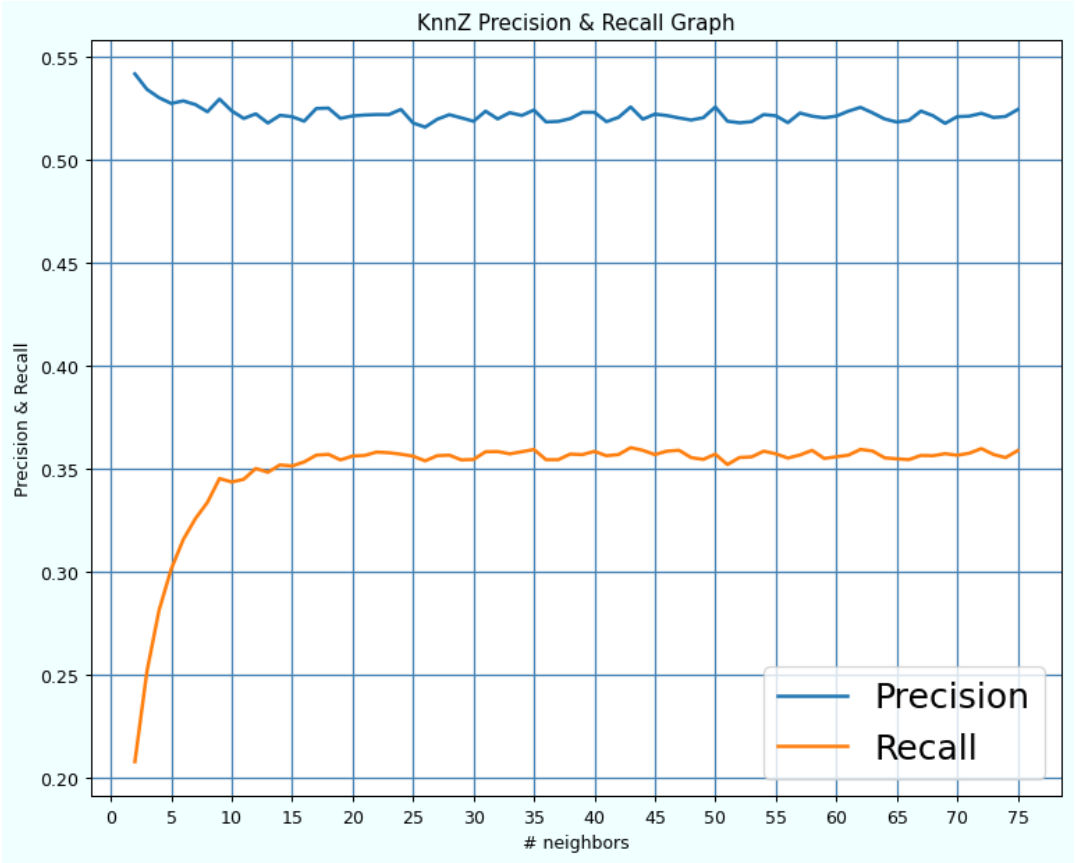


Figure 4.3.4: kNNZ Pearson Precision and Recall graph for the different number of neighbors.

4.3.4 kNN Z-Score Normalization (Pearson Baseline Coefficient)

As in the previous method, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The kNNZ algorithm is the same, with a single change in the similarity calculation method. In this part, the Pearson Baseline Coefficient is examined. Each completion for a different number of neighbors produces a different mean. The RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 4.4.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8896212804986628]
MEAN MAE OF ALL NEIGHBORS:  [0.6687882017017738]

The smallest RMSE is: 0.8846591868093764 for  22  number of neighbors
The smallest MAE is: 0.6648375576172852 for  15  number of neighbors
```

Figure 4.4.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 4.4.2, in order to visualize the changes based on the difference in the number of neighbors.

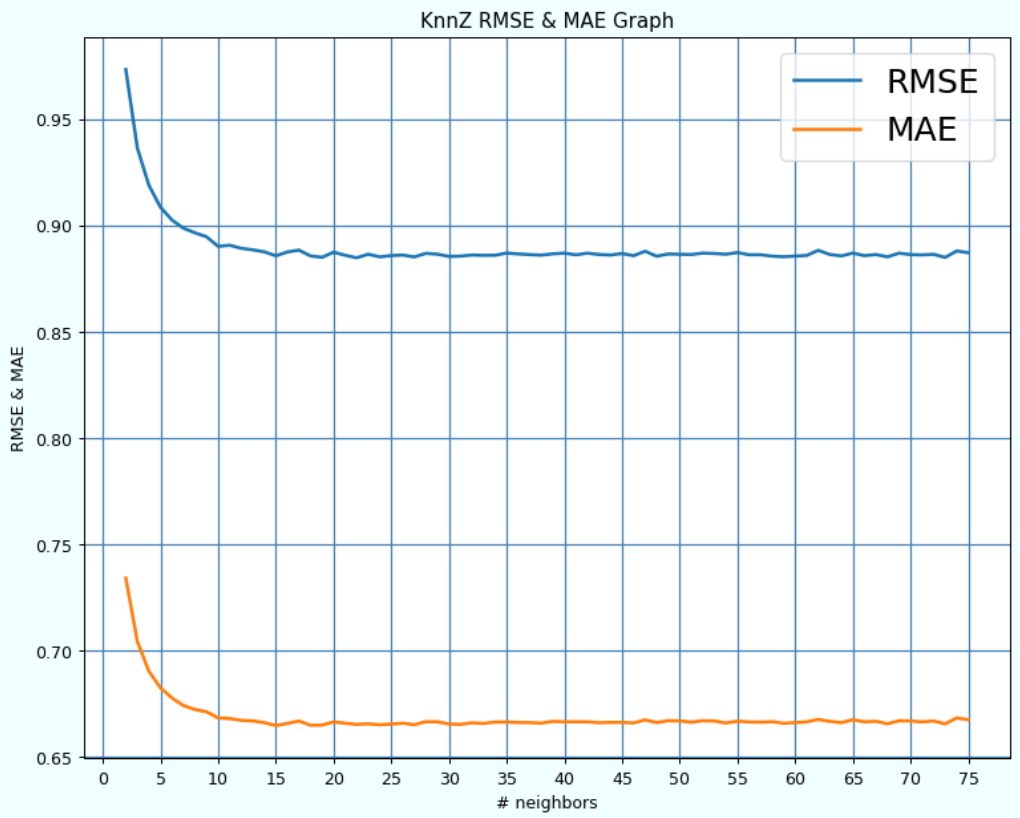


Figure 4.4.2: kNNZ Pearson Baseline RMSE and MAE graph for the different number of neighbors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated, as well as the best values observed and for which number of neighbors as seen in Figure 4.3.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5647553041292949
MEAN RECALL OF ALL NEIGHBORS:  0.38323079885606265

The highest precision is: 0.6236556620682704 for  2  number of neighbors
The highest recall is: 0.40505578730598313 for  13  number of neighbors
```

Figure 4.4.3: The averages for RMSE and MAE, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 4.3.4, in order to visualize the changes based on the difference in the number of neighbors.

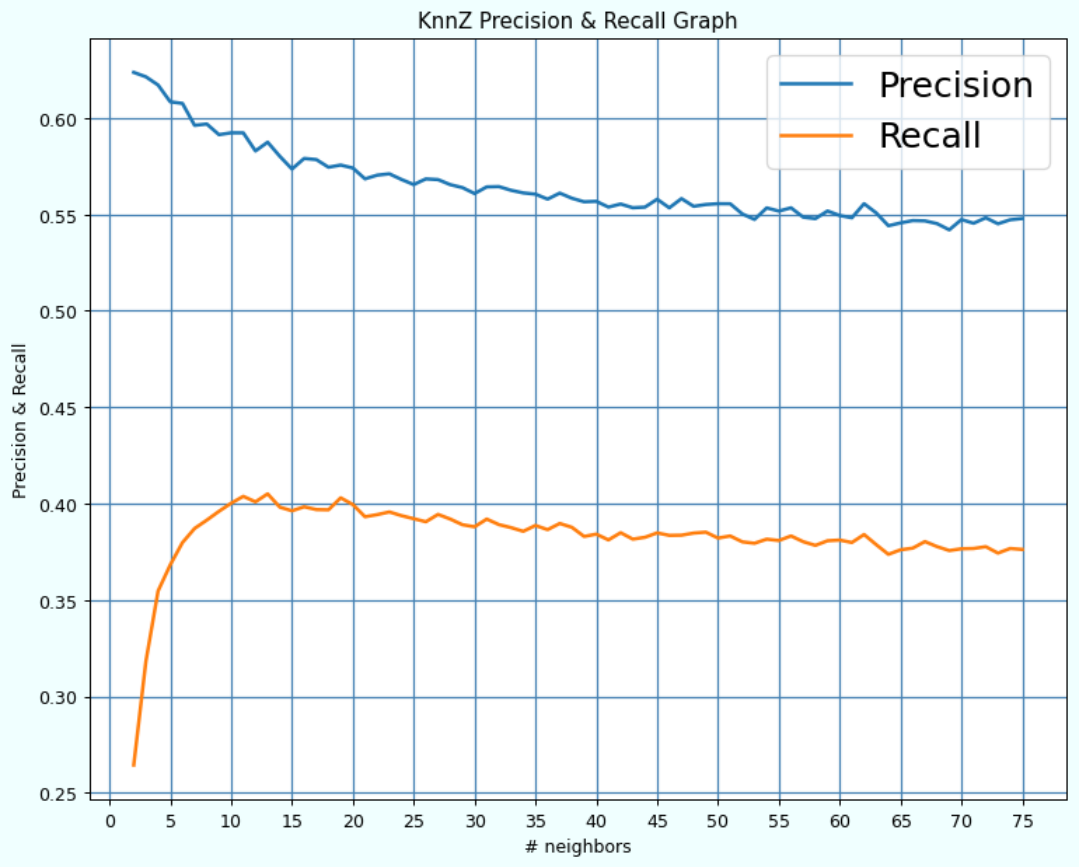


Figure 4.4.4: kNNZ Pearson Baseline Precision and Recall graph for the different number of neighbors.

4.4 Non-Negative Matrix Factorization (NMF)

As in all previous methods, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The algorithm used is the Non-Negative Matrix Factorization, further analyzed in Chapter 3. Each completion for a different number of latent factors produces a different mean. Finally, the RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 5.1.1.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.9669260304845875]
MEAN MAE OF ALL NEIGHBORS:  [0.7533966131221933]

The smallest RMSE is: 0.9442985800055084 for  37  number of neighbors
The smallest MAE is: 0.7260769032785845 for  47  number of neighbors
```

Figure 5.1.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 5.1.2, in order to visualize the changes based on the difference in the number of latent factors.

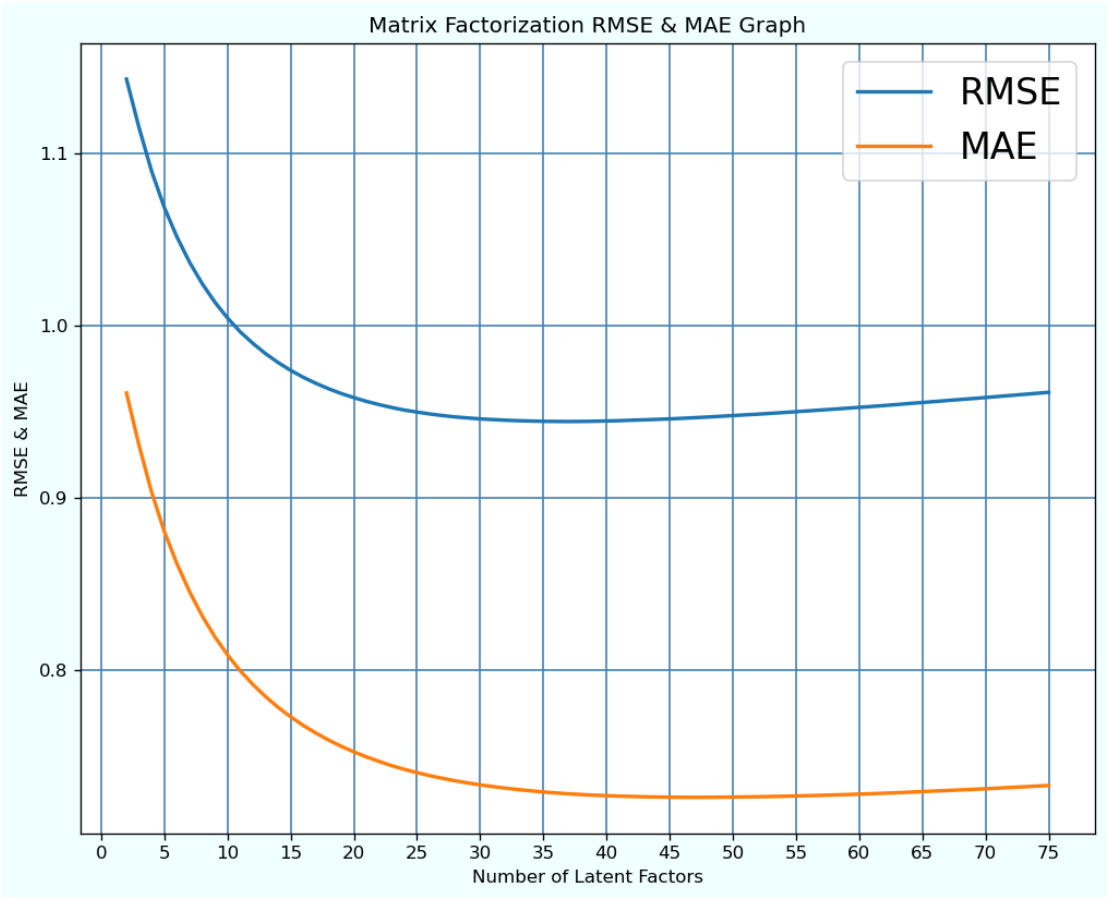


Figure 5.1.2: NMF RMSE and MAE graph for the different number of latent factors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated and the best values observed and for which number of latent factors as seen in Figure 5.1.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5533321765552833
MEAN RECALL OF ALL NEIGHBORS:  0.49408076126986844

The highest precision is: 0.614652923877092 for  67  number of neighbors
The highest recall is: 0.6643441834079423 for  75  number of neighbors
```

Figure 5.1.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 5.1.4, in order to visualize the changes based on the difference in the number of latent factors.

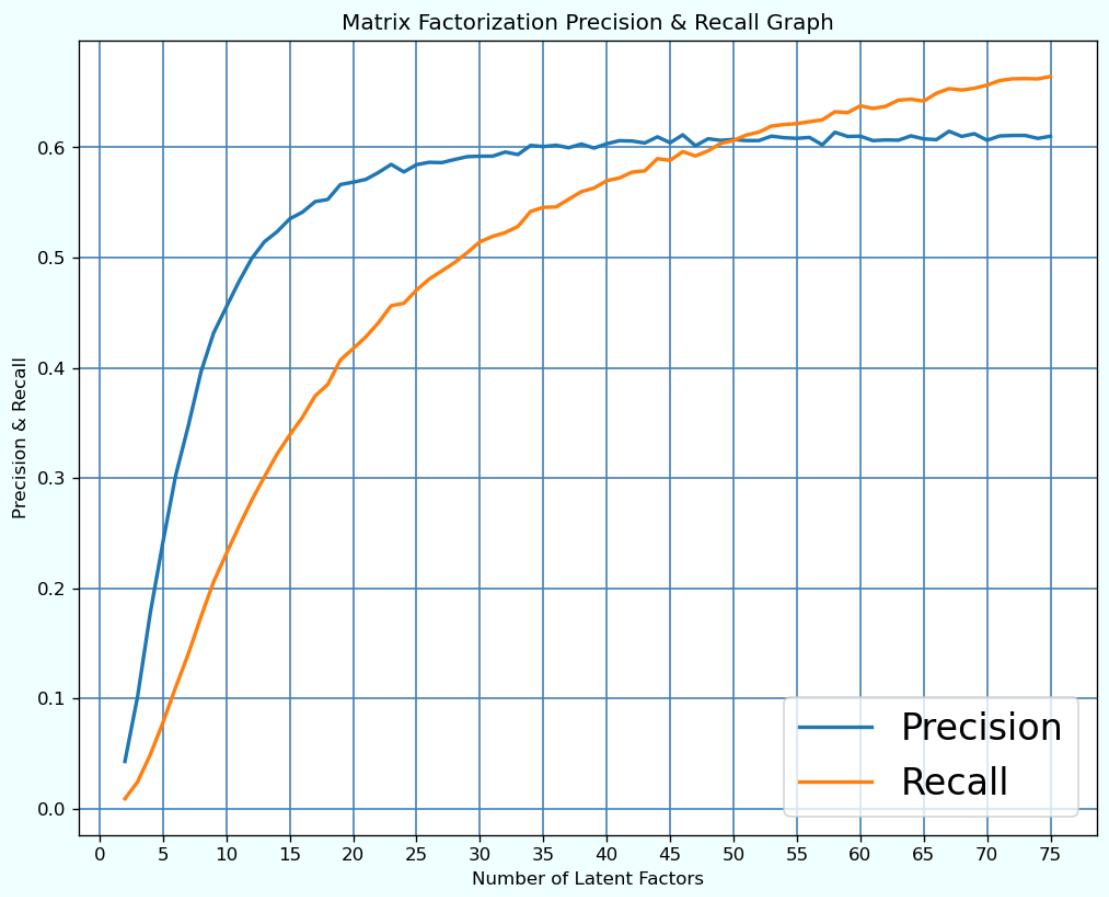


Figure 5.1.4: NMF Precision and Recall graph for the different number of latent factors.

4.5 Singular Value Decomposition Algorithm (SVD).

As in all previous methods, the algorithm begins with the insertion and evaluation of the dataset. Since the dataframe is the same, the same results and conclusions are produced. The algorithm used is the Singular Value Decomposition, further analyzed in Chapter 3. Each completion for a different number of latent factors produces a different mean. Finally, the RMSE and MAE means are saved and analyzed, producing the overall mean of the method, the best results, and where it has been observed, represented in Figure 6.1.1. Perhaps the strongest contestant, an algorithm with stable and low error values, as is proven on the following page.

```
MEAN RMSE OF ALL NEIGHBORS:  [0.8662798456653483]
MEAN MAE OF ALL NEIGHBORS:  [0.6653402388827655]

The smallest RMSE is: 0.8661269062837099 for  50  number of neighbors
The smallest MAE is: 0.665140796618049 for  49  number of neighbors
```

Figure 6.1.1: The averages for RMSE and MAE, the best values, and where those have been observed.

The RMSE, MAE means are plotted in the following graph, Figure 6.1.2, in order to visualize the changes based on the difference in the number of latent factors.

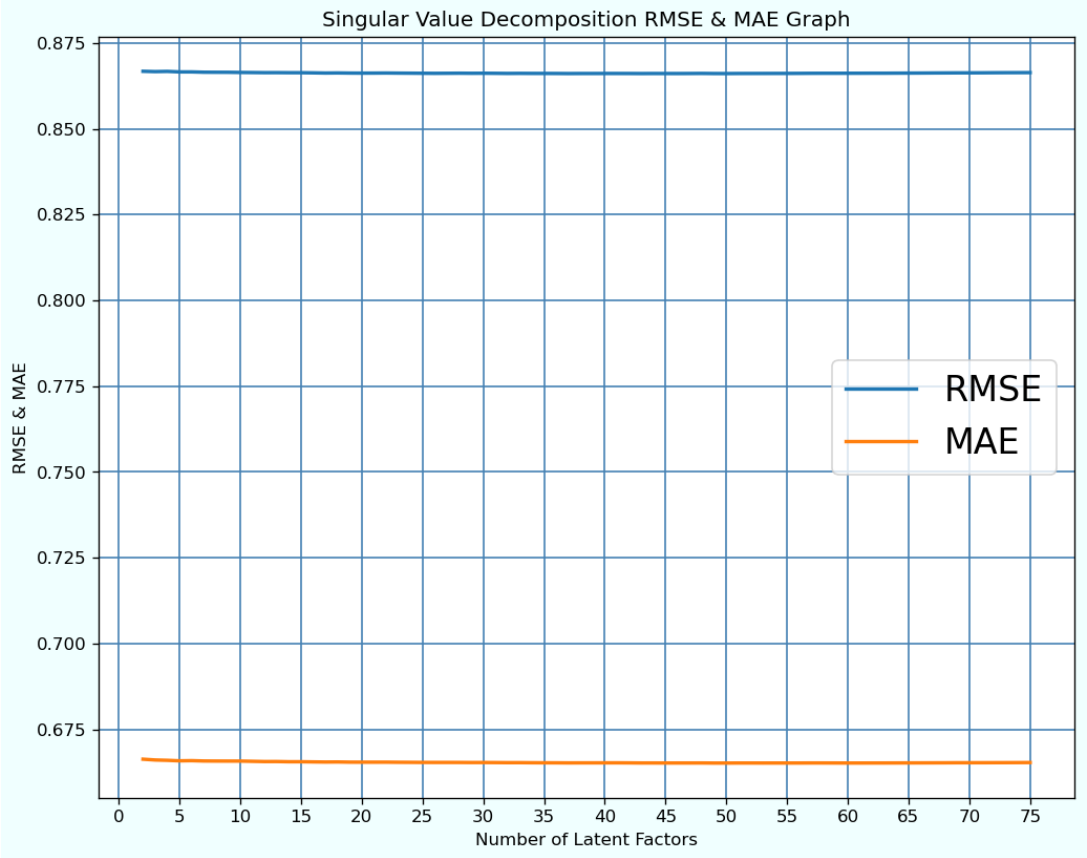


Figure 6.1.2: SVD RMSE and MAE graph for the different number of latent factors.

The next step is the collection of the means for Precision and Recall. From the group of means, the overall average is calculated and the best values observed and for which number of latent factors as seen in Figure 6.1.3.

```
MEAN PRECISION OF ALL NEIGHBORS:  0.5128338692448615
MEAN RECALL OF ALL NEIGHBORS:  0.32445459573533275

The highest precision is: 0.5251526457913094 for  62  number of neighbors
The highest recall is: 0.33725926344132484 for  51  number of neighbors
```

Figure 6.1.3: The averages for Precision and Recall, the best values, and where those have been observed.

The Precision and Recall means are then plotted in the following graph, Figure 6.1.4, in order to visualize the changes based on the difference in the number of latent factors.

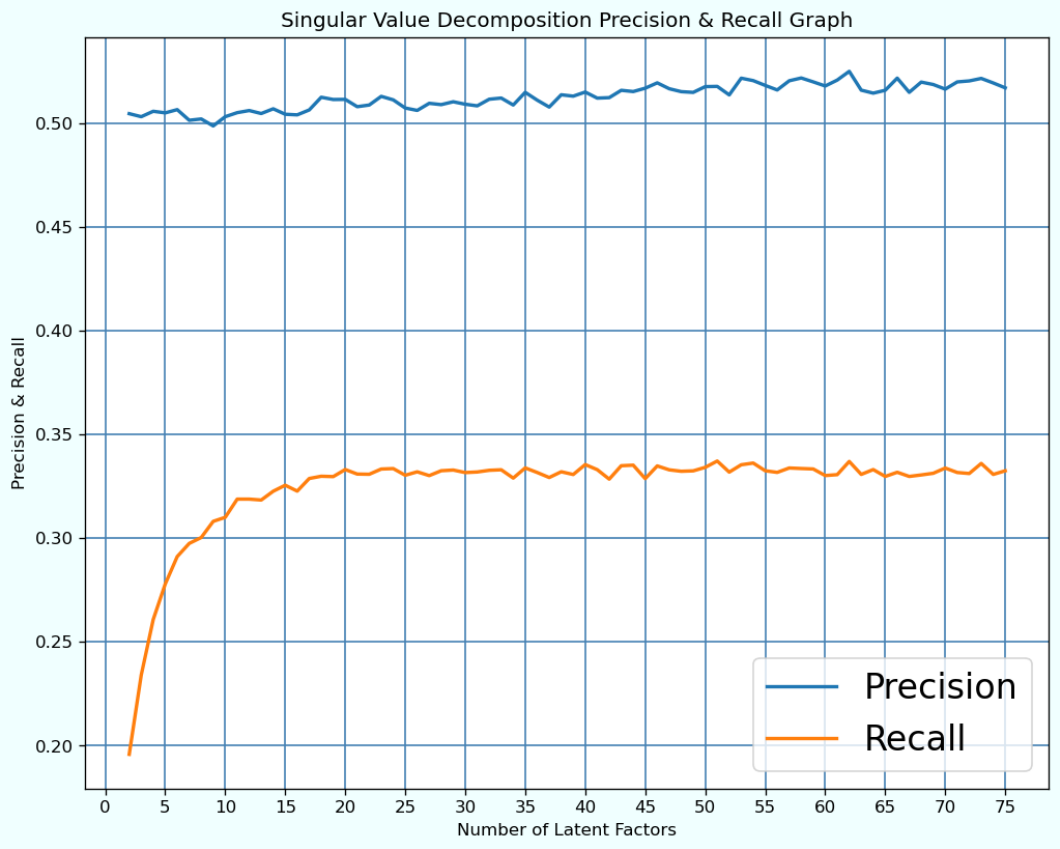


Figure 6.1.4: SVD Precision and Recall graph for the different number of latent factors.

Chapter 5: Recommendation Systems Test Results Comparison

In this chapter, the data acquired in the previous steps are compared to locate the best method if such a method exists. The first comparison is for the kNN algorithm, the second for the kNN with Z-score normalization, the third for the more efficient paradigms of the first two comparisons. The last and final comparison is between the best results of the nearest neighbor philosophy and matrix factorization (NMF, SVD).

5.1 Comparing the different similarity measures for the kNN algorithm

The values of all the means of the kNN have been saved in CSV files. At this point are separated by category(RMSE, MAE, Precision, Recall) and compared between them. Thus, the sequence of comparison is RMSE, MAE, Precision, Recall.

RMSE

The accuracy of prediction is a most crucial factor for an RS. One of the primary metrics for calculating quality efficiently between different algorithms is the RMSE. A metric that is further explained in Chapter 2. Here in Figure 7.1.1 are presented the values of the RMSE means for the kNN algorithm for the Cosine, MSD, P, and PB similarities for the different number of neighbors.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	1.015078	0.996673	0.972804	1.019342
1	3	0.965673	0.956310	0.938610	0.970691
2	4	0.943103	0.936260	0.920453	0.943929
3	5	0.932376	0.925267	0.910780	0.928302
4	6	0.925239	0.917592	0.903398	0.920490
..	...	...	...	...	...
69	71	0.897162	0.892678	0.888584	0.889483
70	72	0.896719	0.892817	0.887502	0.889811
71	73	0.896075	0.892891	0.887678	0.889609
72	74	0.896171	0.892490	0.887898	0.891410
73	75	0.897126	0.892905	0.887686	0.891439

Figure 7.1.1: The dataframe of RMSE values of the different similarity measures.

The values contained in this particular dataframe are then visualized in a graph, Figure 7.1.2, to assist in selecting the best similarity measuring method for the kNN algorithm.

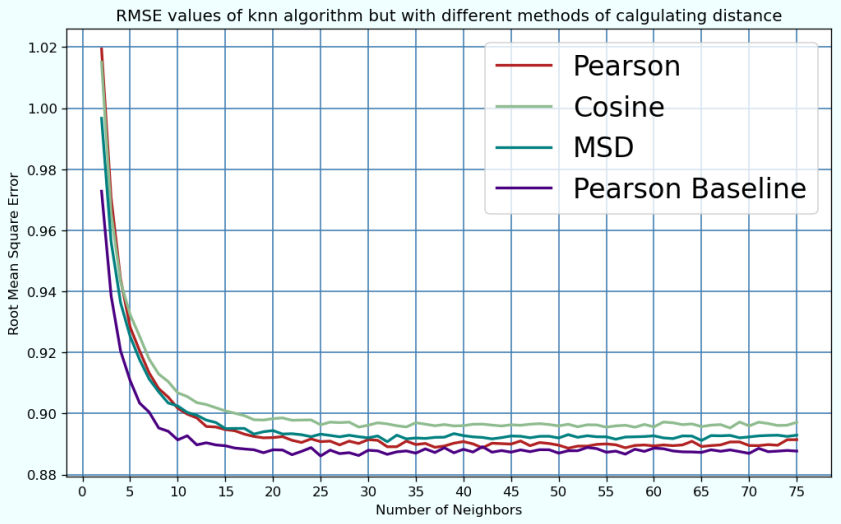


Figure 7.1.2: The graph of kNN RMSE for Cosine, MSD, Pearson, and Pearson Baseline methods.

From Figure 7.1.2 is becoming evident that the lowest mean error rate has been given from the method of Pearson Baseline (purple), as for almost every number of different neighbors, the error is lower than the rest.

MAE

Another important metric of prediction accuracy is the Mean Absolute Error Rate calculating quality efficiently between different algorithms. A metric that is further explained in Chapter 2. Here in Figure 7.1.3 are presented the values of the MAE means for the kNN algorithm for the Cosine, MSD, P, and PB similarities for the different number of neighbors.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	0.783349	0.770066	0.742765	0.786090
1	3	0.747267	0.737116	0.714976	0.748062
2	4	0.728671	0.721101	0.699669	0.725575
3	5	0.719197	0.711856	0.691260	0.712136
4	6	0.713125	0.704978	0.685218	0.704662
..	...	...	...	...	...
69	71	0.685413	0.681518	0.673966	0.677187
70	72	0.685392	0.681499	0.673078	0.677833
71	73	0.684841	0.681874	0.672596	0.677592
72	74	0.684917	0.681259	0.673019	0.678557
73	75	0.685257	0.681747	0.673016	0.678278

Figure 7.1.3: The dataframe of MAE values of the different similarity measures.

The values contained in this particular dataframe are then visualized in a graph, Figure 7.1.4, to assist in selecting the best similarity measuring method for the kNN algorithm.

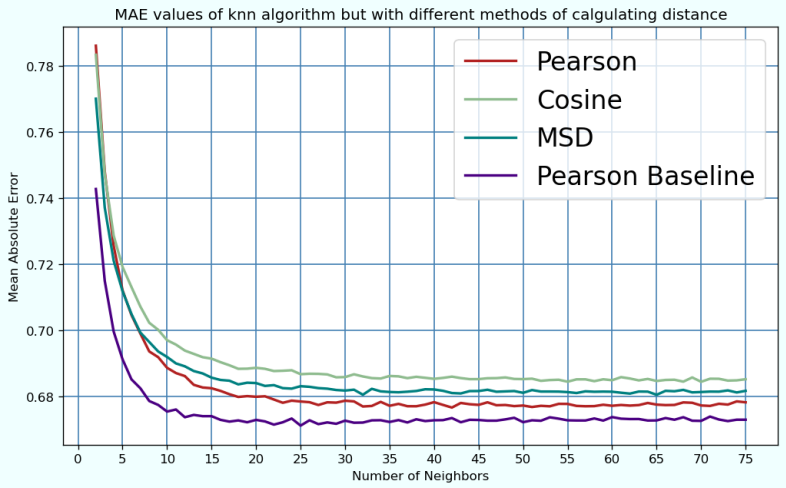


Figure 7.1.4: The graph for kNN MAE of Cosine, MSD, Pearson, and Pearson Baseline methods.

From Figure 7.1.4 is becoming clear that the difference for mean absolute error is even more pronounced. The Pearson similarity measure that was inferior but close to the Pearson Baseline for the RMSE is not on par for the MAE. Thus, having judged both prediction quality measures, the Pearson Baseline Similarity measure offers the lowest error rates for the kNN algorithm.

Precision

The users' confidence in an RS does not only depend on its ability to predict items the user approves but, between them, recommend the best. Therefore, the vital role of measuring the RSs' ability to do so is are the recommendation metrics. In this section, the means of Precision for the kNN algorithm are compared between them.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	0.529464	0.551296	0.624408	0.551806
1	3	0.507524	0.532927	0.618054	0.545661
2	4	0.496386	0.526643	0.612615	0.534727
3	5	0.484568	0.514521	0.602726	0.528874
4	6	0.476625	0.506530	0.602128	0.525509
..	...	...	...	...	...
69	71	0.471761	0.491356	0.541882	0.520581
70	72	0.469618	0.492954	0.544854	0.520980
71	73	0.469154	0.488123	0.542364	0.516732
72	74	0.472748	0.489659	0.543296	0.515517
73	75	0.470604	0.487441	0.538470	0.516556

Figure 7.2.1: The dataframe of Precision values of the different similarity measures.

The values contained in this particular dataframe are then visualized in a graph, Figure 7.2.2, to assist in selecting the best similarity measuring method for the kNN algorithm.

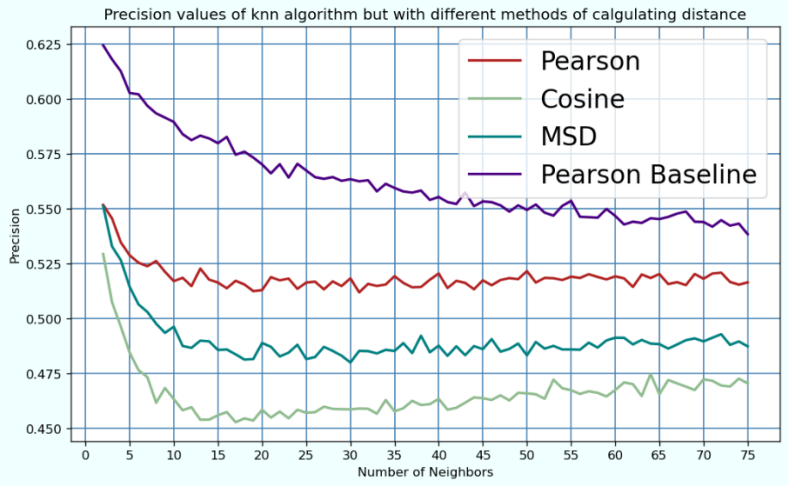


Figure 7.2.2: The graph for kNN Precision of Cosine, MSD, Pearson, and Pearson Baseline methods.

As seen in Figure 7.2.2., and moreover observed in the previous metrics, the Pearson Baseline method offers noticeable better results, higher precision than the rest especially compared with the Cosine and MSD methods.

Recall

The final metric is Recall, another quality of recommendation metric. The Recall means for the different number of neighbors are presented in Figure 7.2.3.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	0.207578	0.219378	0.269497	0.220917
1	3	0.233003	0.252469	0.318362	0.260090
2	4	0.254108	0.272385	0.349170	0.277034
3	5	0.261956	0.282026	0.366113	0.299173
4	6	0.268508	0.290790	0.382515	0.308724
..	...	...	...	...	...
69	71	0.319535	0.328776	0.379153	0.359411
70	72	0.315854	0.332641	0.380715	0.358734
71	73	0.318576	0.326179	0.377755	0.357700
72	74	0.320248	0.328796	0.377644	0.358219
73	75	0.319620	0.328600	0.376531	0.358170

Figure 7.2.3: The dataframe of Recall values of the different similarity measures.

The means from the dataframe presented in Figure 7.2.3 are then visualized to assist in selecting the best similarity measuring method for the kNN algorithm Figure 7.2.4.

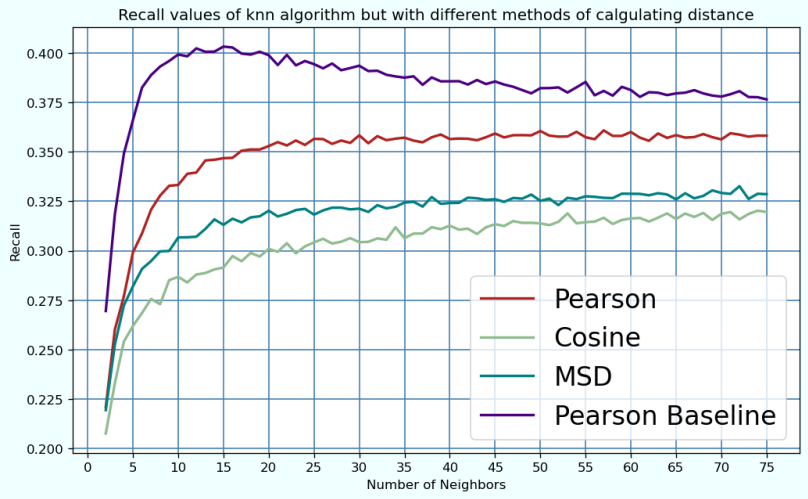


Figure 7.2.4: The graph for kNN Recall of Cosine, MSD, Pearson, and Pearson Baseline methods.

From Figures 7.1.2, 7.1.4, 7.2.2, 7.2.4, we can conclude that the Pearson Baseline and Pearson methods supply much better results on all accounts: RMSE, MAE, Precision, and Recall. More specifically, in this dataset, the best solution, based on prediction and recommendation measures for the kNN algorithm, is the Pearson Baseline similarity measure.

5.2 Comparing the different similarity measures for the kNNZ algorithm

The Pearson Baseline method has come ahead from the previous comparisons as the best for the simple kNN. Since both algorithms are similar, the hypothesis can be made that similar results will be found for the kNNZ.

RMSE

Here in Figure 8.1.1 are presented the values of the RMSE means for the kNNZ algorithm for the Cosine, MSD, P, and PB similarities for the different number of neighbors.

	Neighbors	Pearson_Baseline	Cosine	MSD	Pearson
0	2	0.973180	1.047219	1.030629	1.010072
1	3	0.936264	0.988854	0.977715	0.963562
2	4	0.918749	0.959641	0.952553	0.939370
3	5	0.908474	0.943544	0.936749	0.924447
4	6	0.902413	0.931679	0.924414	0.916172
..	...	...	...	...	...
69	71	0.886035	0.894203	0.891628	0.886902
70	72	0.886326	0.893864	0.893260	0.888036
71	73	0.884823	0.894640	0.892208	0.886987
72	74	0.887880	0.894949	0.892542	0.887765
73	75	0.887058	0.895486	0.892523	0.887091

Figure 8.1.1: The dataframe of RMSE values of the different similarity measures.

The means from the dataframe presented in Figure 8.1.2 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 8.1.2.

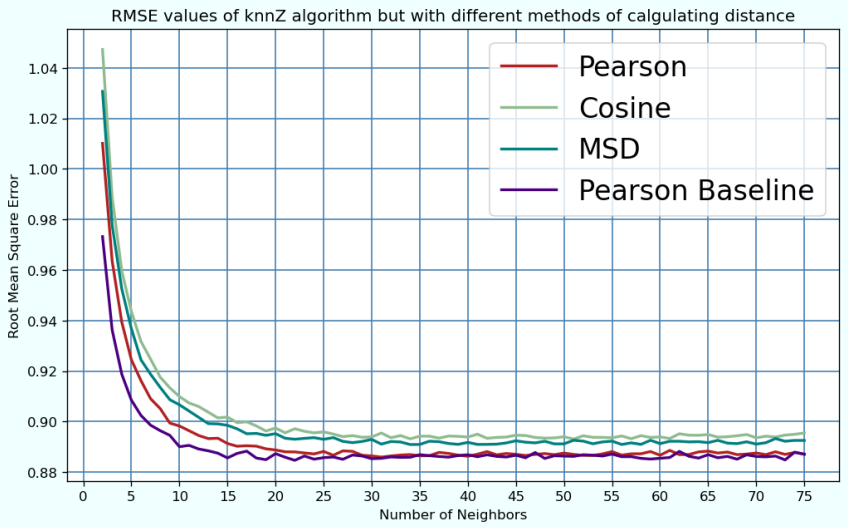


Figure 8.1.2: The graph of kNNZ RMSE for Cosine, MSD, Pearson, and Pearson Baseline methods.

We have seen, from previous observations, that for the simple kNN and the RMSE metrics, Pearson and Pearson Baseline were pretty close. In the kNNZ, the difference is even more obscure. Nevertheless, both show lower error rates than the Cosine and MSD.

MAE

Figure 8.1.3 presents the values of the MAE means for the kNNZ algorithm for the Cosine, MSD, P, and PB similarities for the different number of neighbors.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	0.799394	0.785943	0.734101	0.768041
1	3	0.756099	0.746047	0.704312	0.732338
2	4	0.735102	0.727144	0.690341	0.713260
3	5	0.721951	0.714920	0.682547	0.701293
4	6	0.712858	0.704819	0.677770	0.694903
..	...	...	...	...	...
69	71	0.678447	0.676129	0.666516	0.670084
70	72	0.678598	0.677242	0.666954	0.670462
71	73	0.678884	0.676418	0.665582	0.669979
72	74	0.678981	0.676679	0.668387	0.670724
73	75	0.679597	0.676612	0.667519	0.670034

Figure 8.1.3: The dataframe of MAE values of the different similarity measures.

The means from the dataframe presented in Figure 8.1.3 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 8.1.4.

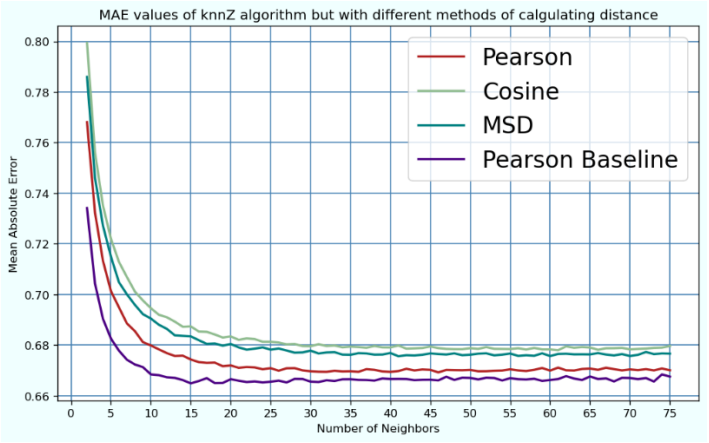


Figure 8.1.4: The graph of kNNZ MAE for Cosine, MSD, Pearson, and Pearson Baseline methods.

Once more, even without the theoretical background, Figure 8.1.4 attests to the connection between kNN and kNNZ. Both kNN and kNNZ show an increased differentiation in the MAE compared to the RMSE between Pearson and Pearson Baseline. As the best similarity measures were closer in the RMSE are more distant in the MAE. The Pearson Baseline again is superior by a small margin.

Precision

In this section, the means of Precision for the kNNZ algorithm are compared between them. Figure 8.2.1 shows the values collected from the previous algorithms.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	0.543151	0.554959	0.623656	0.541951
1	3	0.525113	0.551704	0.621355	0.534427
2	4	0.518306	0.535070	0.617098	0.530391
3	5	0.509542	0.534792	0.608377	0.527608
4	6	0.505132	0.529211	0.607573	0.528834
..	...	...	...	...	...
69	71	0.488701	0.506835	0.545411	0.521421
70	72	0.488101	0.511636	0.548239	0.522800
71	73	0.488758	0.511174	0.545130	0.520777
72	74	0.485408	0.509270	0.547202	0.521265
73	75	0.485689	0.508758	0.547855	0.524638

Figure 8.2.1: The dataframe of Precision values of the different similarity measures.

The means from the dataframe presented in Figure 8.2.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 8.2.2.

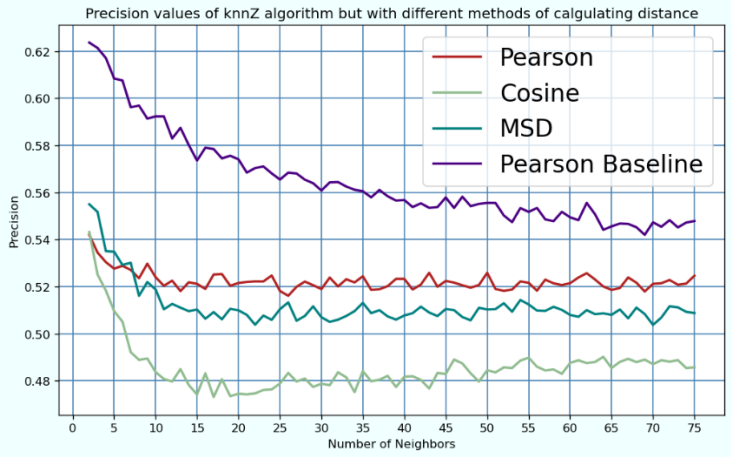


Figure 8.2.2: The graph of kNNZ Precision for Cosine, MSD, Pearson, and Pearson Baseline methods.

Pearson Baseline continues to excel compared with the rest of the similarity measure offering higher Precision at all numbers of neighbors and at an impressive margin.

Recall

The final metric is Recall, another quality of recommendation metric. The Recall means for the different number of neighbors are presented in Figure 7.2.3.

	Neighbors	Cosine	MSD	Pearson_Baseline	Pearson
0	2	0.216494	0.224880	0.264394	0.207933
1	3	0.242768	0.261687	0.318267	0.251880
2	4	0.265265	0.277781	0.354689	0.281526
3	5	0.274297	0.291046	0.368232	0.301441
4	6	0.283623	0.300864	0.379757	0.315803
..	...	...	...	...	...
69	71	0.327797	0.336690	0.376715	0.357741
70	72	0.327250	0.339832	0.377650	0.360008
71	73	0.325769	0.338242	0.374315	0.357137
72	74	0.325742	0.338097	0.376706	0.355617
73	75	0.324472	0.340574	0.376218	0.359024

Figure 8.2.3: The dataframe of Recall values of the different similarity measures.

The means from the dataframe presented in Figure 8.2.3 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 8.2.4.

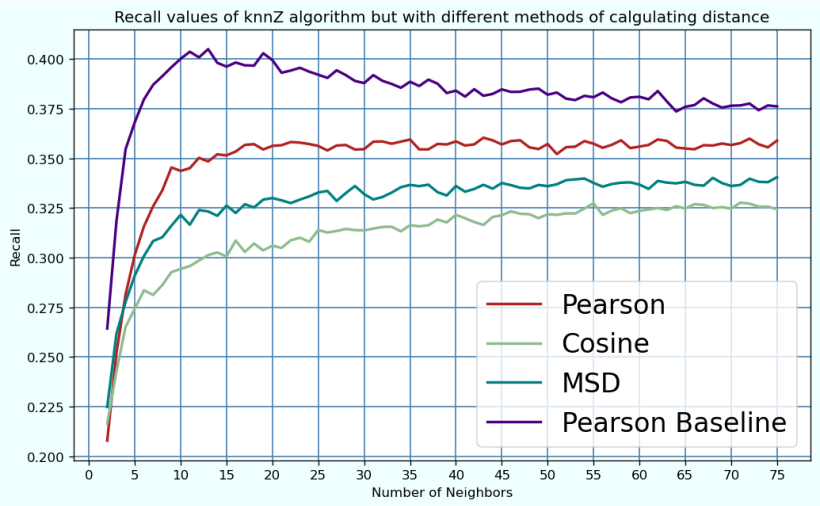


Figure 8.2.4: The graph of kNNZ Recall for Cosine, MSD, Pearson, and Pearson Baseline methods.

Figure 8.2.4 shows that the Pearson Baseline similarity measure offers the highest Recall. Thus, it is evident that the PB is the best method for the kNNZ algorithm by having lower errors and higher Precision and recall. Although due to its proximity for the prediction measures, the Pearson similarity will be included for the next step of comparisons.

5.3 Comparing the best similarity measures for the Knn and KnnZ algorithms

As already stated, the combinations that have been selected for a deeper analysis are the kNN Pearson Baseline, the kNNZ Pearson, and the kNNZ Pearson Baseline. A secondary objective is to prove which of the two algorithms, kNN and kNNZ, is better.

RMSE

Here in Figure 8.1.1 are presented the values of the RMSE means for the kNN PB, the kNNZ P, and kNNZ PB algorithm for the different number of neighbors.

	Neighbors	Pearson_Baseline	Pearson_BaselineZ	Pearson
0	2	0.972804	0.973180	1.010072
1	3	0.938610	0.936264	0.963562
2	4	0.920453	0.918749	0.939370
3	5	0.910780	0.908474	0.924447
4	6	0.903398	0.902413	0.916172
..	...	...	...	...
69	71	0.888584	0.886035	0.886902
70	72	0.887502	0.886326	0.888036
71	73	0.887678	0.884823	0.886987
72	74	0.887898	0.887880	0.887765
73	75	0.887686	0.887058	0.887091

Figure 9.1.1: The dataframe of RMSE values of the different similarity measures for the KNN Pearson Baseline, kNNZ Pearson, and KNN Pearson Baseline.

The means from the dataframe presented in Figure 9.1.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 9.1.2.

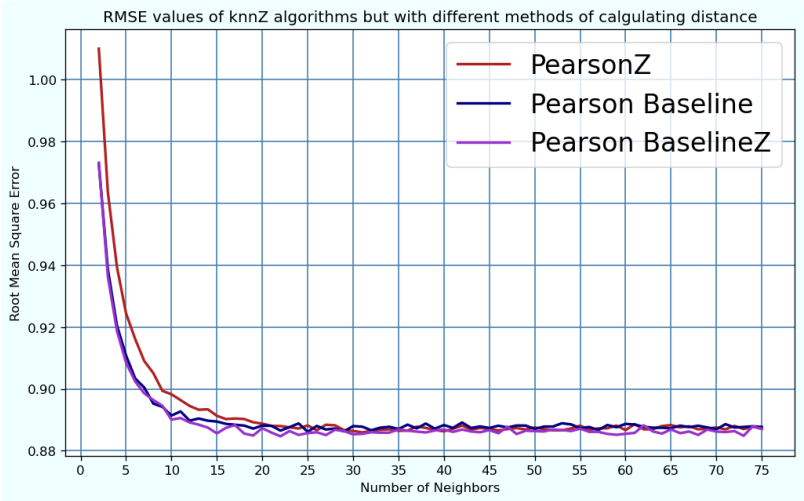


Figure 9.1.2: The comparing graph of RMSE between kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

Figure 9.1.2 is impossible to show clearly the best method as the differences are minuscule. For that reason, the algorithm makes a deeper analysis, calculating both the mean and the median of the RMSE mean values shown in Figure 9.1.3.

```
       Neighbors  Pearson_Baseline  Pearson_BaselineZ    Pearson
count  74.000000         74.000000          74.000000  74.000000
mean   38.500000          0.891118           0.889621   0.892986
std    21.505813          0.012368           0.012517   0.018414
min     2.000000          0.886093           0.884659   0.885931
25%    20.250000          0.887415           0.885776   0.886903
50%    38.500000          0.887933           0.886328   0.887532
75%    56.750000          0.888681           0.887250   0.888708
max    75.000000          0.972804           0.973180   1.010072

Regarding the knn  algorithm with Pearson Baseline the mean is:  0.8911180237479102
Regarding the knnZ algorithm with Pearson Baseline the mean is:  0.8896212804986628
Regarding the knnZ algorithm with Pearson the mean is:  0.8929861155460977

Regarding the knn  algorithm with Pearson Baseline the median is:  0.8879332965158567
Regarding the knnZ algorithm with Pearson Baseline the median is:  0.8863281622064509
Regarding the knnZ algorithm with Pearson the median is:  0.8875323354627418
```

Figure 9.1.3: Mean and Median of RMSE for the kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

From Figure 9.1.3, we can conclude that kNNZ with Pearson Baseline has by the tiniest margins the lowest values no matter the method of calculation. Especially for RMSE, all three are similar, and a different dataset might have different results as to which is the best.

MAE

Figure 9.1.4 presents the values of the RMSE means for the kNN PB, the kNNZ P, and the kNNZ PB algorithm for the different number of neighbors.

	Neighbors	Pearson_Baseline	Pearson_BaselineZ	Pearson
0	2	0.742765	0.734101	0.768041
1	3	0.714976	0.704312	0.732338
2	4	0.699669	0.690341	0.713260
3	5	0.691260	0.682547	0.701293
4	6	0.685218	0.677770	0.694903
..	...	...	...	...
69	71	0.673966	0.666516	0.670084
70	72	0.673078	0.666954	0.670462
71	73	0.672596	0.665582	0.669979
72	74	0.673019	0.668387	0.670724
73	75	0.673016	0.667519	0.670034

Figure 9.1.4: The dataframe of MAE values of the different similarity measures for the KNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

The means from the dataframe presented in Figure 9.1.4 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 9.1.5.

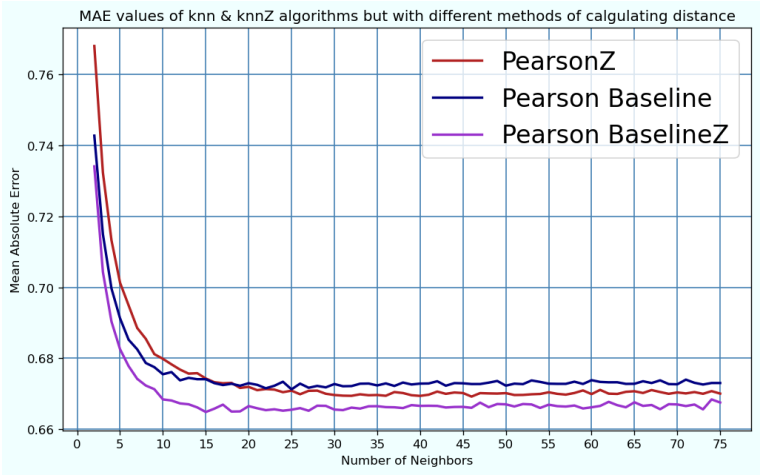
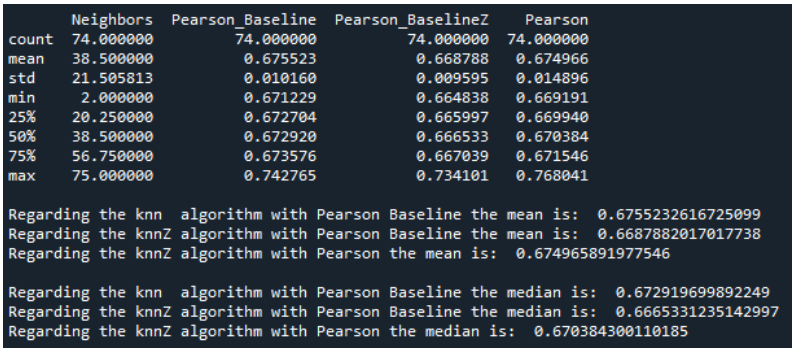


Figure 9.1.5: The comparing graph of MAE between kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

The differences are more pronounced for the MAE, something already observed in previous examples. To the degree that with a simple look is easy to prove that the kNNZ PB is the better solution. Pearson Baseline consistently offers a better error rate. However, it is unclear which of the kNN PB and kNNZ P have the lowest MAE.



	Neighbors	Pearson_Baseline	Pearson_BaselineZ	Pearson
count	74.000000	74.000000	74.000000	74.000000
mean	38.500000	0.675523	0.668788	0.674966
std	21.505813	0.010160	0.009595	0.014896
min	2.000000	0.671229	0.664838	0.669191
25%	20.250000	0.672704	0.665997	0.669940
50%	38.500000	0.672920	0.666533	0.670384
75%	56.750000	0.673576	0.667039	0.671546
max	75.000000	0.742765	0.734101	0.768041

```
Regarding the knn  algorithm with Pearson Baseline the mean is:  0.6755232616725099
Regarding the knnZ algorithm with Pearson Baseline the mean is:  0.6687882017017738
Regarding the knnZ algorithm with Pearson the mean is:  0.674965891977546

Regarding the knn  algorithm with Pearson Baseline the median is:  0.672919699892249
Regarding the knnZ algorithm with Pearson Baseline the median is:  0.6665331235142997
Regarding the knnZ algorithm with Pearson the median is:  0.670384300110185
```

Figure 9.1.6: Mean and Median of MAE for the kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

Figure 9.1.6 proves that kNNZ PB is the best and also that kNNZ P gives the second-best results even though its advantage starts at the 19th neighbor and before that had a clear negative difference from the kNN PB

Precision

Figure 9.2.1 presents the values of the Precision means for the kNN PB, the kNNZ P, and the kNNZ PB algorithm for the different number of neighbors.

	Neighbors	Pearson_Baseline	Pearson_BaselineZ	Pearson
0	2	0.624408	0.623656	0.541951
1	3	0.618054	0.621355	0.534427
2	4	0.612615	0.617098	0.530391
3	5	0.602726	0.608377	0.527608
4	6	0.602128	0.607573	0.528834
..	...	...	...	...
69	71	0.541882	0.545411	0.521421
70	72	0.544854	0.548239	0.522800
71	73	0.542364	0.545130	0.520777
72	74	0.543296	0.547202	0.521265
73	75	0.538470	0.547855	0.524638

Figure 9.2.1: The dataframe of MAE values of the different similarity measures for the kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

The means from the dataframe presented in Figure 9.2.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 9.2.2.

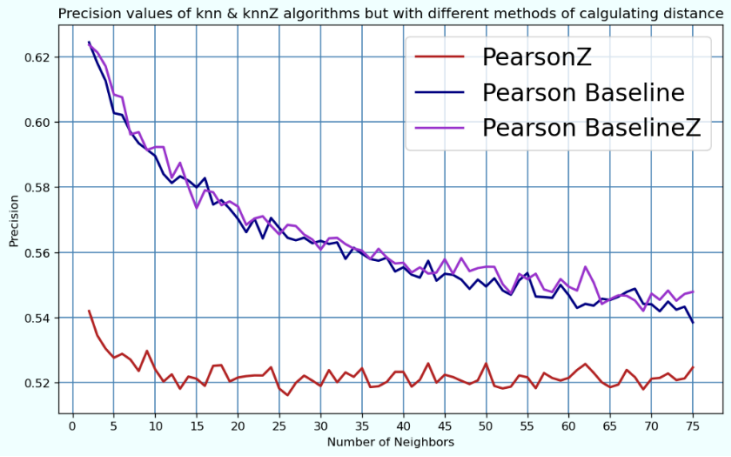


Figure 9.2.2: The comparison graph of Precision between kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

A fascinating graph, even though the kNNZ Pearson (red) and the kNNZ Pearson Baseline (purple) run the same algorithm, kNNZ PB (purple) shares its pattern with the kNN PB (blue). Figure 9.2.2 proves the degree of influence the similarity measure has for the algorithm. However, unfortunately, it cannot show if the kNN PB or the kNNZ PB is better than the other.

```
       Neighbors  Pearson_Baseline  Pearson_BaselineZ    Pearson
count  74.000000         74.000000          74.000000  74.000000
mean   38.500000          0.562664           0.564755   0.522348
std    21.505813          0.019491           0.019381   0.003935
min     2.000000          0.538470           0.541994   0.516094
25%    20.250000          0.547927           0.550992   0.520061
50%    38.500000          0.557390           0.558052   0.521625
75%    56.750000          0.570467           0.572899   0.523762
max    75.000000          0.624408           0.623656   0.541951

Regarding the knn  algorithm with Pearson Baseline the mean is:  0.5626638734655804
Regarding the knnZ algorithm with Pearson Baseline the mean is:  0.5647553041292949
Regarding the knnZ algorithm with Pearson the mean is:  0.5223480636836835

Regarding the knn  algorithm with Pearson Baseline the median is:  0.5573904042345617
Regarding the knnZ algorithm with Pearson Baseline the median is:  0.5580523347918231
Regarding the knnZ algorithm with Pearson the median is:  0.5216254532560529
```

Figure 9.2.3: Mean and Median of Precision for the kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

Figure 9.2.3 informs us how extraordinarily close they are. With an incredibly tiny difference, kNNZ PB comes on top. Though for such a negligible difference, they are practically identical for the metric of Precision.

Recall

Figure 9.2.4 shows the values of the Recall means for the kNN PB, the kNNZ P, and the kNNZ PB algorithm for the different number of neighbors.

	Neighbors	Pearson_Baseline	Pearson_BaselineZ	Pearson
0	2	0.269497	0.264394	0.207933
1	3	0.318362	0.318267	0.251880
2	4	0.349170	0.354689	0.281526
3	5	0.366113	0.368232	0.301441
4	6	0.382515	0.379757	0.315803
..	...	...	...	...
69	71	0.379153	0.376715	0.357741
70	72	0.380715	0.377650	0.360008
71	73	0.377755	0.374315	0.357137
72	74	0.377644	0.376706	0.355617
73	75	0.376531	0.376218	0.359024

Figure 9.2.4: The dataframe of Recall values of the different similarity measures for the kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

The means from the dataframe presented in Figure 9.2.4 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 9.2.5

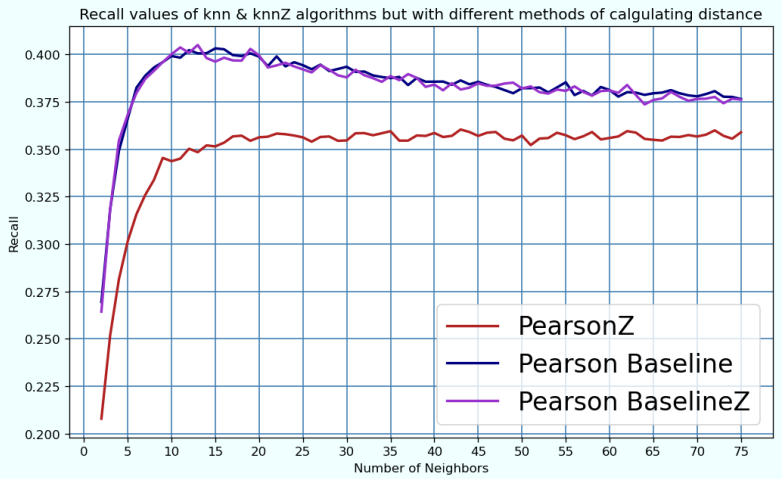
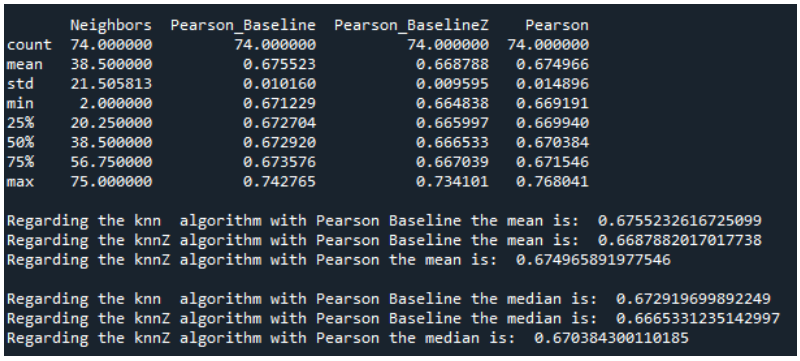


Figure 9.3.5: The comparing graph of Recall. Between kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

The kNN PB and kNNZ PB are almost identical for both recommendation measures, while the Pearson similarity measure has fallen behind. Again, is not possible to see which of the two is bests so that the algorithm will provide more information, presented in Figure 9.3.6.



	Neighbors	Pearson_Baseline	Pearson_BaselineZ	Pearson
count	74.000000	74.000000	74.000000	74.000000
mean	38.500000	0.675523	0.668788	0.674966
std	21.505813	0.010160	0.009595	0.014896
min	2.000000	0.671229	0.664838	0.669191
25%	20.250000	0.672704	0.665997	0.669940
50%	38.500000	0.672920	0.666533	0.670384
75%	56.750000	0.673576	0.667039	0.671546
max	75.000000	0.742765	0.734101	0.768041

Regarding the knn algorithm with Pearson Baseline the mean is: 0.6755232616725099
Regarding the knnZ algorithm with Pearson Baseline the mean is: 0.6687882017017738
Regarding the knnZ algorithm with Pearson the mean is: 0.674965891977546

Regarding the knn algorithm with Pearson Baseline the median is: 0.672919699892249
Regarding the knnZ algorithm with Pearson Baseline the median is: 0.6665331235142997
Regarding the knnZ algorithm with Pearson the median is: 0.670384300110185

Figure 9.3.6: Mean and Median of Recall for the kNN Pearson Baseline, kNNZ Pearson, and kNN Pearson Baseline.

Once again, kNNZ PB comes on top with higher Recall. All three methods follow the same pattern, but kNNZ PB and kNN PB are incredibly close. The Pearson Baseline similarity measure has proven it can achieve better results in both algorithms. As for the second question, choosing one or the other algorithm is not an easy decision. The kNN PB performed admirably especially compared to the kNNZ P, though, as the kNNZ PB proved superior to the kNN PB, we can hypothesize that the kNN Z-Score normalization algorithm carries a slight advantage. The combination selected for the final round of testing will be the kNNZ PB for its more impressive, compared to the recommendation measures, results for the RMSE and MAE against the kNN PB algorithm.

5.4 Comparing similarity measures between the best results of the kNN family comparison and the Matrix Factorization approach.

This part offers an in-depth comparison of the two matrix factorization algorithms and the best of the nearest neighbor. As in the previews parts, each metric shows the value of the process under different comparable optics.

RMSE

The final round of comparisons begins with the RMSE between the Non-Negative Matrix Factorization algorithm, the Singular Value Decomposition algorithm, and the k-Nearest Neighbor with Z-Score Normalization algorithm combined with the Pearson Baseline Similarity measure. Figure 10.1.1 presents the values of the RMSE means of the NMF, SVD, kNNZ PB.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
0	2	0.866789	0.973180	1.143125
1	3	0.866697	0.936264	1.114810
2	4	0.866758	0.918749	1.089415
3	5	0.866610	0.908474	1.068524
4	6	0.866611	0.902413	1.051414
..	...	...	...	...
69	71	0.866347	0.886035	0.958835
70	72	0.866369	0.886326	0.959432
71	73	0.866385	0.884823	0.960032
72	74	0.866394	0.887880	0.960643
73	75	0.866404	0.887058	0.961250

Figure 10.1.1: The dataframe of RMSE values of the different similarity measures for the SVD, kNNZ Pearson Baseline, and NMF.

The means from the dataframe presented in Figure 10.1.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 10.1.2.

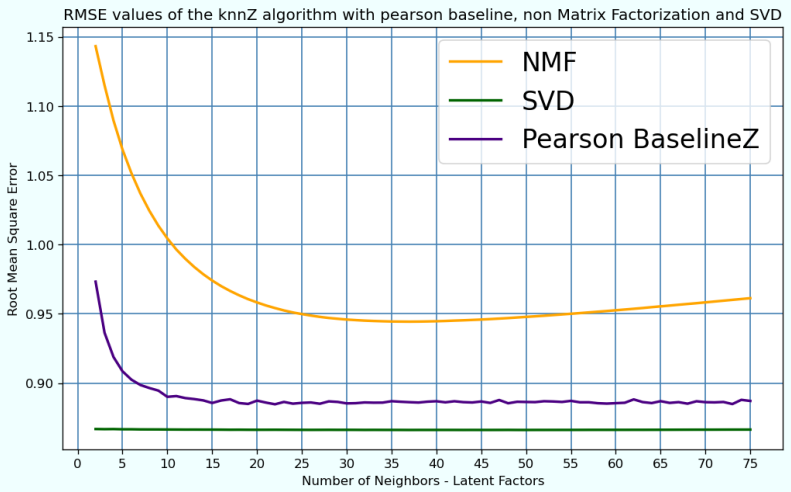


Figure 10.1.2: The comparing graph of RMSE between NMF, kNNZ Pearson Baseline, and SVD.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
count	74.000000	74.000000	74.000000	74.000000
mean	38.500000	0.866280	0.889621	0.966926
std	21.505813	0.000151	0.012517	0.039825
min	2.000000	0.866127	0.884659	0.944299
25%	20.250000	0.866167	0.885776	0.946609
50%	38.500000	0.866232	0.886328	0.952458
75%	56.750000	0.866344	0.887250	0.960619
max	75.000000	0.866789	0.973180	1.143125

Figure 10.1.3: Mean of RMSE for the SVD, kNNZ Pearson Baseline, and NMF.

The differences are apparent with every combination offering different error rates while retaining their ranking from best to worst throughout the number of neighbors and latent factors. The SVD offers a stable and low error rate for all the latent factors.

MAE

Figure 10.1.1 presents the values of the MAE means of the NMF, SVD, kNNZ PB.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
0	2	0.666273	0.734101	0.960885
1	3	0.666038	0.704312	0.930301
2	4	0.665942	0.690341	0.903080
3	5	0.665795	0.682547	0.880338
4	6	0.665844	0.677770	0.861627
..	...	...	...	...
69	71	0.665246	0.666516	0.731436
70	72	0.665257	0.666954	0.731807
71	73	0.665267	0.665582	0.732178
72	74	0.665278	0.668387	0.732566
73	75	0.665293	0.667519	0.732952

Figure 10.2.1: The dataframe of MAE values of the different similarity measures for the SVD, KNNZ Pearson Baseline, and NMF.

The means from the dataframe presented in Figure 10.1.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 10.1.2. Also, Figures 10.2.3 and 10.2.4 provide deeper insight, separating the kNNZ PB and SVD, showing the lowest MAE.

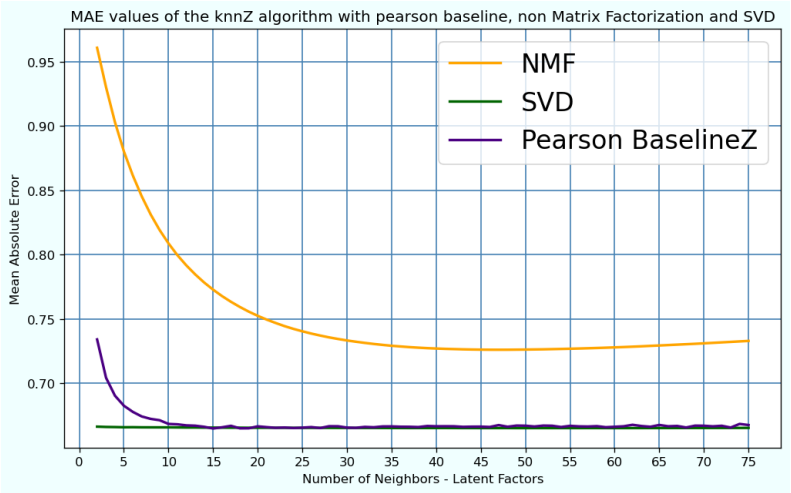


Figure 10.2.2: The comparing graph of MAE between NMF, KNNZ Pearson Baseline, and SVD.

While with the kNN algorithms, the values for the RMSE were closer, and the MAE offered a clear view; here, that pattern is reversed and necessary to prove which algorithm has the best results.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
count	74.000000	74.000000	74.000000	74.000000
mean	38.500000	0.665340	0.668788	0.753397
std	21.505813	0.000238	0.009595	0.050009
min	2.000000	0.665141	0.664838	0.726077
25%	20.250000	0.665171	0.665997	0.727351
50%	38.500000	0.665243	0.666533	0.730884
75%	56.750000	0.665402	0.667039	0.751724
max	75.000000	0.666273	0.734101	0.960885

Figure 10.2.3: Mean of MAE for the SVD, KNNZ Pearson Baseline, and NMF.

```
Regarding the NMF  algorithm the mean is:  0.7533966131221933
Regarding the knnZ algorithm with Pearson Baseline the mean is:  0.6687882017017738
Regarding the SVD algorithm the mean is:  0.6653402388827655

Regarding the NMF  algorithm the median is:  0.7308843015747236
Regarding the knnZ algorithm with Pearson Baseline the median is:  0.6665331235142997
Regarding the SVD algorithm the median is:  0.6652434070226245
```

Figure 10.2.4: Mean and Median of MAE for the SVD, KNNZ Pearson Baseline, and NMF.

Even after abating the effect of the kNNZ PB outliers appearing where the neighbor number is less than 10, the SVD algorithm shows a slightly lower error rate. For both prediction measures, the NMF algorithm had very high error rates.

Precision

Figure 10.2.1 presents the values of the Precision means for the kNN PB, the kNNZ P, and the kNNZ PB algorithm for the different number of neighbors.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
0	2	0.504753	0.623656	0.042827
1	3	0.503294	0.621355	0.101735
2	4	0.505903	0.617098	0.177394
3	5	0.505141	0.608377	0.241820
4	6	0.506699	0.607573	0.302135
..	...	...	...	...
69	71	0.520064	0.545411	0.610432
70	72	0.520543	0.548239	0.610848
71	73	0.521749	0.545130	0.610901
72	74	0.519562	0.547202	0.608181
73	75	0.517201	0.547855	0.610188

Figure 10.2.1: The dataframe of Precision values of the different similarity measures for the SVD, kNNZ Pearson Baseline, and NMF.

The means from the dataframe presented in Figure 10.2.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 10.1.2. Also, Figures 10.3.3 and 10.3.4 provide deeper insight between NMF and kNNZ PB.

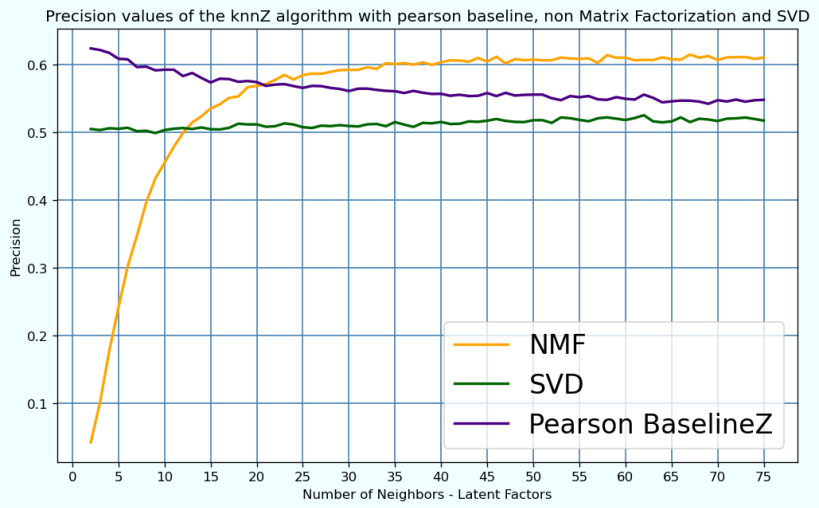


Figure 10.3.2: The comparing graph of Precision between NMF, SVD, and kNNZ Pearson Baseline.

The NMF, for the first time, shows a better metric at any point than the competition. Although from a first glance at the graph, it can not be proven that it offers a better Precision overall.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
count	74.000000	74.000000	74.000000	74.000000
mean	38.500000	0.512834	0.564755	0.553332
std	21.505813	0.006072	0.019381	0.117399
min	2.000000	0.498838	0.541994	0.042827
25%	20.250000	0.507979	0.550992	0.569086
50%	38.500000	0.513147	0.558052	0.601897
75%	56.750000	0.517648	0.572899	0.607829
max	75.000000	0.525153	0.623656	0.614653

Figure 10.3.3: Mean of Precision for the SVD, kNNZ Pearson Baseline, and NMF.

```
Regarding the NMF  algorithm the median is:  0.6018971970671196
Regarding the knnZ algorithm with Pearson Baseline the median is:  0.5580523347918231
Regarding the SVD algorithm the median is:  0.5131467754010746
```

Figure 10.3.4: Mean of MAE for the SVD, kNNZ Pearson Baseline, and NMF.

The highest mean belongs to the kNNZ PB, and the highest median to the NMF means, the reason being its extremely low Precision for a small number of latent factors. Overall the kNNZ PB retains the slightest edge over the NMF and has a higher Precision. Though if the implementation was on a higher number of latent factors, the NMF would be superior to the kNNZ PB.

Recall

Figure 10.4.1 presents the values of the Recall means for the kNN PB, the kNNZ P, and the kNNZ PB algorithm for the different number of neighbors.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
0	2	0.195845	0.264394	0.008931
1	3	0.234068	0.318267	0.024233
2	4	0.260841	0.354689	0.048901
3	5	0.277946	0.368232	0.078062
4	6	0.291200	0.379757	0.110019
..	...	...	...	...
69	71	0.331732	0.376715	0.660695
70	72	0.331241	0.377650	0.662262
71	73	0.336101	0.374315	0.662447
72	74	0.330807	0.376706	0.662189
73	75	0.332554	0.376218	0.664344

Figure 10.4.1: The dataframe of Recall values of the different similarity measures for the SVD, kNNZ Pearson Baseline, and NMF.

The means from the dataframe presented in Figure 10.4.1 are then visualized to assist in selecting the best similarity measuring method for the kNNZ algorithm Figure 10.4.2.

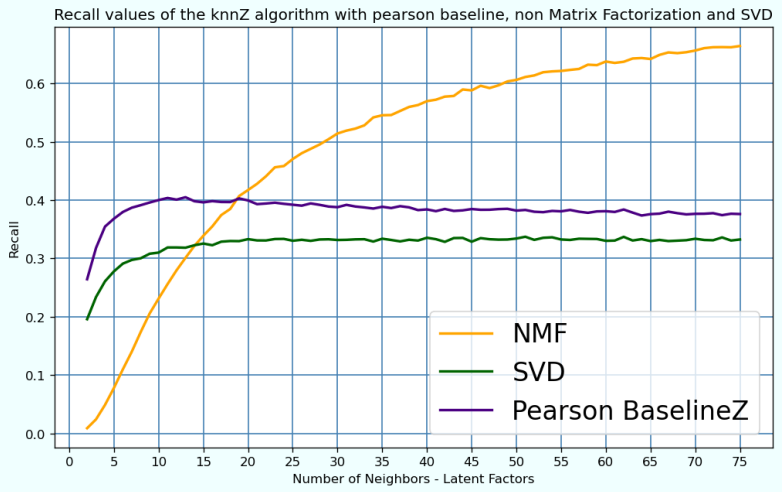


Figure 10.4.2: The comparing graph of Recall between NMF, SVD, and kNNZ Pearson Baseline.

The rise of the NMF for the recommender metrics continues. The SVD has the lowest of the three parallel to the superior kNNZ PB. Figure 10.4.2 shows that the Recall of the NMF is almost double for a high number of latent factors compared to the other two.

	NeighborsLF	SVD	Pearson_BaselineZ	NMF
count	74.000000	74.000000	74.000000	74.000000
mean	38.500000	0.324455	0.383231	0.494081
std	21.505813	0.022756	0.018356	0.176870
min	2.000000	0.195845	0.264394	0.008931
25%	20.250000	0.329097	0.379770	0.420090
50%	38.500000	0.331709	0.384058	0.561492
75%	56.750000	0.333130	0.391977	0.624602
max	75.000000	0.337259	0.405056	0.664344

Figure 10.4.3: Mean of Recall for the SVD, kNNZ Pearson Baseline, and NMF.

	kNNZ_Pearson_Baseline	SVD	NMF
MEAN RMSE	0.88962	0.86627	0.96692
MEAN RMSE % DIFFERENCE	-	-2.69444 %	+7.99489 %
MEAN MAE	0.66878	0.66534	0.75339
MEAN MAE % DIFFERENCE	-	-0.51822 %	+11.23026 %
MEAN PRECISION	0.56475	0.51283	0.55333
MEAN PRECISION % DIFFERENCE	-	-10.12441 %	-2.06442 %
MEAN RECALL	0.38323	0.32445	0.49408
MEAN RECALL % DIFFERENCE	-	-18.11538 %	+22.43559 %

Figure 10.5: Evaluation metrics and percentage differences between the kNNZ Pearson Baseline, SVD, and NMF.

The result from the different graphs and mainly Figure 10.5 shown an extreme gap for the Precision and the Recall between NMF and the other two methods. That deference creates the question of the NMF being perhaps a better solution despite its high error rate. Fortunately, that conversation does not have to take place. The main reason is its execution time.

5.6 Comparing execution times of the NMF, kNNZ PB, and SVD Algorithms

The execution time is an essential factor, and, as previously stated, one of the reasons with other algorithms such as SVD++ was not selected. NMF, even without specific measures, has veered significantly during testing. To such an extent, the top algorithms changed in order to calculate and provide their execution time. A small test group of 20 runs was implemented, and Figure 10.6 shows the findings.

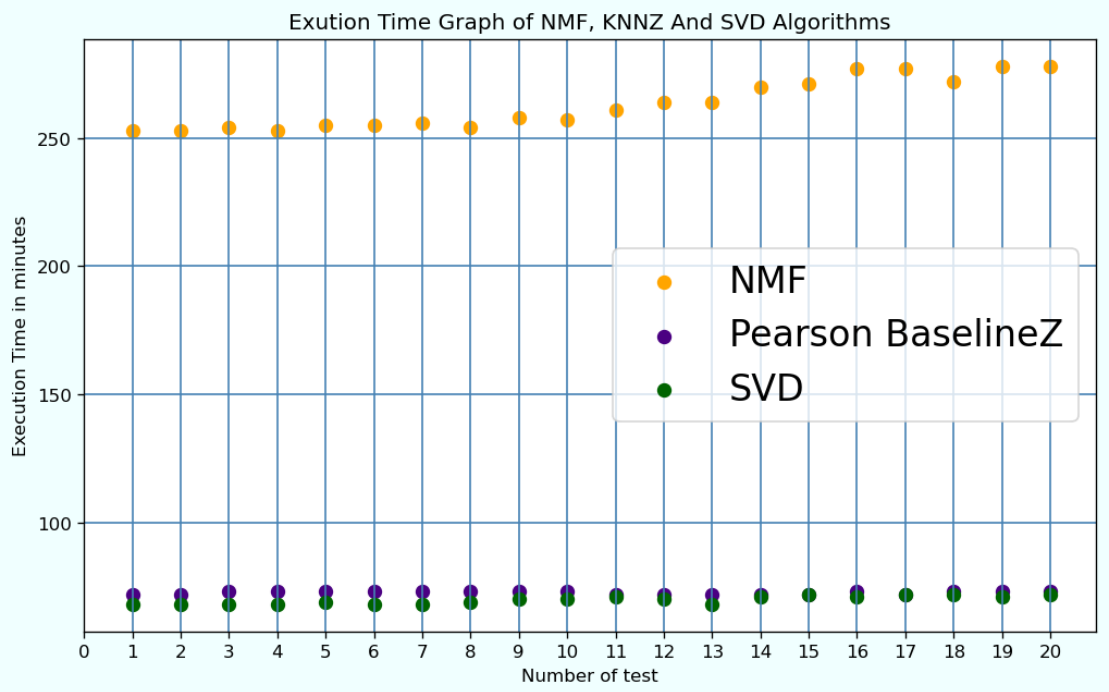


Figure 10.6: Results of the execution time of 20 runs of the NMF, SVD, and kNNZ with the Pearson Baseline similarity measure algorithms.

The kNNZ PB and the SVD need around 50 minutes to be run 74 times for such a number of neighbors and latent factors accordingly. In contrast to the NMF, which needs at least 250 minutes and also has a very high variance. Furthermore, through multiple experiments, it has been proven that the execution time for each factor constantly escalates compared to its number. Meaning that if the goal is those high numbers of Precision and Recall, its long execution time cannot be avoided or bypassed. With that in mind and the majority of the superior result coming at a higher number of latent factors, this method seems unfit for the task, especially if we consider more extensive datasets. Another reason is the high error rate. Even with the high Precision and Recall, the items recommended to the user come from a subpar pool produced with high RMSE and MAE.

As a result, kNNZ and SVD are the most fitting. The SVD offers slightly faster run times, Figure 10.6, and lower error rates than the kNNZ PB, Figure 10.5. While the kNNZ PB shows high Recall and Precision, Figure 10.5. These different methods come close, and if tested to a different dataset, it is possible that the result may differ. That fact makes it difficult to declare that one is superior to the other in every category.

In my opinion, probably the SVD will be preferred for bigger datasets for its low error rates. In contrast, the kNNZ PB for smaller but more diverse datasets where the movie ratings have increased fluctuation, making it more difficult to choose an element preferred by the users. An example that comes to mind is a smaller platform that offers more obscure cinematography like older productions, European cinema, indie directors, etc.

Chapter 6: Final Thoughts

The final chapter focuses on presenting the conclusions of all comparisons between the algorithms, any criticism both on the algorithms, methodologies of user similarity calculation, and on the methodology of testing as a whole. Finally, some concluding thoughts of the writer for the entire project, the limitations faced, and any ideas of improving upon this work.

	RMSE	MAE	Precision	Recall	F1
KNN COSINE	0.9019	0.6908	0,465602	0,303245	0.3673
KNN MSD	0.8973	0.6863	0,490197	0,318226	0.3859
KNN PEARSON	0.8961	0.6831	0,518756	0,348646	0.4170
KNN PEARSON BASELINE	0.8911	0.6755	0,562664	0,384115	0.4566
KNNZ COSINE	0.9018	0.6859	0,485498	0,311769	0.3797
KNNZ MSD	0.8987	0.6825	0,51188	0,328547	0.4002
KNNZ PEARSON	0.8930	0.6750	0,522348	0,349563	0.4188
KNNZ PEARSON BASELINE	0.8896	0.6688	0,564755	0,383231	0.4566
N-G MATRIX FACTORIZATION	0.9669	0.7534	0,553332	0,494081	0.5220
SVD	0.8663	0.6653	0,324455	0,324455	0.3245

Fig

ure 10.7: Total averages of the evaluation measures of all the algorithms

Overall, the experimentations have brought some interesting conclusions. The first and obvious is that the kNN with Z-Score Normalization algorithm, no matter the similarity measure, has achieved slightly better results in both qualities of prediction and recommendation. Moreover, concerning the similarity measures, the cosine has proved to be the less effective of the 4, closely followed by the MSD, both having subpar results than the other 2. On the other hand, both the Pearson and the PB give comparable results, and as seen in Figure 10.7, also they give their best when used in the kNNZ algorithm.

The 2 Matrix Factorization Algorithms are vastly different. NMF shows the worst quality of prediction out of all 12 but, in contrast, when overfitted with a high number of latent factors, offers the best F1 Figure 10.7. While the SVD has the best quality measures out of the test group, but it suffers from low recommendation quality.

The kNN, kNNZ, and the SVD have faster execution times than the NMF, so it is logical to say that if the use of the algorithms was on a periodically altered dataset, the legitimacy of the NMF algorithm in this context is questionable.

Due to the lack of a more fitting computing system, we are obliged to be content with smaller datasets in the context of this endeavor. For the same reason, other algorithms as the SVD++ were not tested, the reason being its prolonged execution time even if it is known for its high-quality measures.

If the continuation of this paper were possible, it would be beneficial if, under the same procedure, alternative and different sized datasets were used to evaluate the 12 combinations. Furthermore, the dataset used was relatively sparse. So it would be interesting to observe how the algorithms would react if the sparsity were different.

Intriguing would be the changes if the datasets would slowly gain new users and items and observing which algorithm would react better to the common cold-start problem. From there, one might want to try to create a hybrid approach trying to encapsulate the best of all worlds.

List of Abbreviations

RS	Recommender / Recommendation System
IFS	Information Filtering System
CF	Collaborative Filtering
CB	Content-Based
kNN	k-Nearest Neighbors Algorithm
kNNZ	k-Nearest Neighbors Algorithm with Z-Score Normalisation
IBCF	Item Based Collaborative Filtering
UBCF	User Based Collaborative Filtering
MF	Matrix Factorization
NMF	Non-Negative matrix factorization
SVD	Singular Value Decomposition
LSI	Latent Semantic Index
MCRS	Multi-Criteria Recommender Systems
MCDM	Multi-Criteria Decision Making
DRARS	Risk-Aware Recommender Systems
MSD	Mean Square Difference Similarity
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
NMAE	Normalised Mean Average Error
PB	Pearson Baseline
ROC	Receiver Operating Characteristic
DP	Differential Privacy
NCT/TF	Number of Common Terms/ Term Frequency
IDE	Integrated Development Environment
BSD	Berkeley Software Distribution
CSV	Comma Separated Values File

References

[1] Francesco Ricci and Lior Rokach and Bracha Shapira, Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35

[2] "Playboy Lead Rise of Recommendation Engines - TIME". TIME.com. 27 May 2010. Retrieved 1 June 2015

[3] Karlgren, Jussi. 1990. "An Algebra for Recommendations." Syslab Working Paper 179 (1990)

[4] Karlgren, Jussi. "Newsgroup Clustering Based On User Behavior-A Recommendation Algebra." SICS Research Report (1994)

[5] Karlgren, Jussi (October 2017). "A digital bookshelf: original work on recommender systems". Retrieved 27 October 2017

[6] Shardanand, Upendra, and Pattie Maes. "Social information filtering: algorithms for automating “word of mouth”." In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 210-217. ACM Press/Addison-Wesley Publishing Co., 1995

[7] Hill, Will, Larry Stead, Mark Rosenstein, and George Furnas. "Recommending and evaluating choices in a virtual community of use." In Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 194-201. ACM Press/Addison-Wesley Publishing Co., 1995

[8] Resnick, Paul, Neophytos Iacovou, Mitesh Suchak, Peter Bergström, and John Riedl. "GroupLens: an open architecture for collaborative filtering of netnews." In Proceedings of the 1994 ACM conference on Computer supported cooperative work, pp. 175-186. ACM, 1994

[9] Resnick, Paul, and Hal R. Varian. "Recommender systems." Communications of the ACM 40, no. 3 (1997): 56-58

[10] Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh WTF: The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web

[11] Baran, Remigiusz; Dziech, Andrzej; Zeja, Andrzej (2018-06-01). "A capable multimedia content discovery platform based on visual content analysis and intelligent data enrichment". Multimedia Tools and Applications. 77 (11): 14077–14091. doi:10.1007/s11042-017-5014-1. ISSN 1573-7721. S2CID 36511631

[12] H. Chen, A. G. Ororbia II, C. L. Giles ExpertSeer: a Keyphrase Based Expert Recommender for Digital Libraries, in arXiv preprint 2015

[13] H. Chen, L. Gou, X. Zhang, C. Giles Collabseer: a search engine for collaboration discovery, in ACM/IEEE Joint Conference on Digital Libraries (JCDL) 2011

[14] S.K. Lee, Y.H. Cho, S.H. Kim, Collaborative filtering with ordinal scale-based implicit ratings for mobile music recommendations, Information Sciences 180 (11) (2010) 2142–2155

[15] K. Choi, D. Yoo, G. Kim, Y. Suh, A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis. Electronic Commerce Research and Applications, in press, doi: 10.1016/j.elerap.2012.02.004

[16] E.R. Núñez-Valdéz, J.M. Cueva-Lovelle, O. Sanjuán-Martı´nez, V. Garcı´ a-Dı´ az, P. Ordoñez, C.E. Montenegro-Marı´ n, Implicit feedback techniques on recommender systems applied to electronic books, Computers in Human Behavior 28 (4) (2012) 1186–1193

[17] Hosein Jafarkarimi; A.T.H. Sim and R. Saadatdoost A Naïve Recommendation Model for Large Databases, International Journal of Information and Education Technology, June 2012

[18] Prem Melville and Vikas Sindhwani, Recommender Systems, Encyclopedia of Machine Learning, 2010.

[19] R. J. Mooney & L. Roy (1999). Content-based book recommendation using learning for text categorization. In Workshop Recom. Sys.: Algo. and Evaluation.

[20] ChenHung-Hsuan; ChenPu (2019-01-09). "Differentiating Regularization Weights -- A Simple Mechanism to Alleviate Cold Start in Recommender Systems". ACM Transactions on Knowledge Discovery from Data (TKDD). 13: 1–22. doi:10.1145/3285954. S2CID 59337456

[21] Rubens, Neil; Elahi, Mehdi; Sugiyama, Masashi; Kaplan, Dain (2016). "Active Learning in Recommender Systems". In Ricci, Francesco; Rokach, Lior; Shapira, Bracha (eds.). Recommender Systems Handbook (2 ed.). Springer US. doi:10.1007/978-1-4899-7637-6_24. ISBN 978-1-4899-7637-6

[22] Elahi, Mehdi; Ricci, Francesco; Rubens, Neil (2016). "A survey of active learning in collaborative filtering recommender systems". Computer Science Review. 20: 29–50. doi:10.1016/j.cosrev.2016.05.002

[23] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, David M. Pennock (2002). Methods and Metrics for Cold-Start Recommendations. Proceedings of the 25th Annual International ACMSIGIR Conference on Research and Development in Information Retrieval (SIGIR 2002). : ACM. pp. 253–260

[24] Bi, Xuan; Qu, Annie; Wang, Junhui; Shen, Xiaotong (2017). "A group-specific recommender system". Journal of the American Statistical Association. 112 (519): 1344–1353. doi:10.1080/01621459.2016.1219261. S2CID 125187672

[25] Mangalindan, JP. 2012. “Amazon’S Recommendation Secret Fortune”. http://fortune.com/2012/07/30/amazons-recommendation-secret/

[26] Koren, Yehuda.2009. “Collaborative Filtering with Temporal Dynamics.” 15th ACM SIGKDD Int’l Conf. Knowledge Discovery and Data Mining (KDD 09), ACM (2009): 447-455

[27] Liu, Jiahui, Dolan, Peter, Pedersen, Elin Rønby.2010. “ Personalized news recommendation based on click behavior”, In: Rich, et al. (eds.) In the 14th Int. Conf. on Intelligent User Interfaces (IUI), ACM, (2010): 31–40

[28] Terveen, Loren; Hill, Will (2001). "Beyond Recommender Systems: Helping People Help Each Other" (PDF). Addison-Wesley. p. 6. Retrieved 16 January 2012

[29] Sarwar, George Kaypi, Joseph Konstan, John Riedl.2001. “Item-based Collaborative Filtering Recommendation Algorithms.” In the 10th International World Wide Web Conference, 285-295

[30] G. Adomavicius, A. Tuzhilin, Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions, IEEE Transactions on Knowledge and Data Engineering 17 (6) (2005) 734–749

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

[32] J.L. Herlocker, J.A. Konstan, A.L. Borchers, J.T. Riedl, An algorithmic framework for performing collaborative filtering, in: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 1999, pp. 230–237

[33] F. Carmagnola, F. Vernero, P. Grillo, SoNARS: a social networks-based algorithm for social recommender systems, in: Proceedings of the 17th International Conference on User Modeling, Adaptation, and Personalization: Formerly UM and AH, 2009, pp. 223–234

[34] X. Su, T.M. Khoshgoftaar, A survey of collaborative filtering techniques, Advance in Artificial Intelligence 2009 (2009) 1–19

[35] R.R. Yager, Fuzzy logic methods in recommender systems, Fuzzy Sets and Systems 136 (2) (2003) 133–149

[36] J. Bobadilla, A. Hernando, F. Ortega, A. Gutiérrez, Collaborative filtering based on significances, Information Sciences 185 (1) (2012) 1–17

[37] J. Bobadilla, A. Hernando, F. Ortega, J. Bernal, A framework for collaborative filtering recommender systems, Expert Systems with Applications 38 (12) (2011) 14609–14623

[38] Owen, Sean, Anil, Robin, Dunning, Ted, Friedman, Ellen. 2011. Mahout in action. Shelter Island NY: Manning

[39] Verbert, Katrien, Drachsler, Hendrik, Manouselis, Nikos, Wolpers, Martin, Vuorikari, Vuorikari, Riina, Duval, Erik.2011. “Dataset-driven Research for Improving Recommender Systems for Learning.” 1st International Conference Learning Analytics & Knowledge, ACM (2011): 44-53

[40] Gong, Songjie. 2010. “A Collaborative Filtering Recommendation Algorithm Based On User Clustering And Item Clustering.” JSW 5 (7) doi:10.4304/jsw.5.7.745-752

[41] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filtering recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, 2007, pp. 291–324 (Chapter 9)

[42] R. Burke, Hybrid recommender systems: survey and experiments, User Modeling and User-Adapted Interaction 12 (4) (2002) 331–370

[43] C. Porcel, A. Tejeda-Lorente, M.A. Martı´ nez, E. Herrera-Viedma, A hybrid recommender system for the selective dissemination of research resources in a technology transfer office, Information Sciences 184 (1) (2012) 1–19

[44] M.G. Vozalis, K.G. Margaritis, Using SVD and demographic data for the enhancement of generalized collaborative filtering, Information Sciences 177 (2007) 3017–3037

[45] A.B. Barragáns-Martínez, E. Costa-Montenegro, J.C. Burguillo, M. Rey-López,F.A. Mikic-Fonte, A. Peleteiro, A hybrid content-based and item-based collaborative filtering approach to recommend TV programs enhanced with singular value decomposition, Information Sciences 180 (22) (2010) 4290–4311

[46] K. Choi, D. Yoo, G. Kim, Y. Suh, A hybrid online-product recommendation system: combining implicit rating-based collaborative filtering and sequential pattern analysis. Electronic Commerce Research and Applications, in press, doi: 10.1016/j.elerap.2012.02.004

[47] L. Candillier, F. Meyer, M. Boullé, Comparing state-of-the-art collaborative filtering systems, Lecture Notes in Computer Science 4571 (2007) 548–562

[48] F. Kong, X. Sun, S. Ye, A comparison of several algorithms for collaborative filtering in startup stage, IEEE Transactions on Networks, Sensing and Control (2005) 25–28

[49] K. Lang, NewsWeeder: learning to filter netnews, in: Proceedings 12th International Conference on Machine Learning, 1995, pp. 331–339

[50] N. Antonopoulus, J. Salter, Cinema screen recommender agent: combining collaborative and content-based filtering, IEEE Intelligent Systems (2006) 35–41

[51] R. Meteren, M. Someren, Using content-based filtering for recommendation, in: Proceedings of ECML 2000 Workshop: Machine Learning in Information Age, 2000, pp. 47–56

[52] M. Pazzani, A framework for collaborative, content-based, and demographic filtering, Artificial Intelligence Review-Special Issue on Data Mining on the Internet 13 (5-6) (1999) 393–408

[53] B. Krulwich, Lifestyle finder: intelligent user profiling using large-scale demographic data, Artificial Intelligence Magazine 18 (2) (1997) 37–45

[54] H. Ingoo, J.O. Kyong, H.R. Tae, The collaborative filtering recommendation based on SOM cluster-indexing CBR, Expert Systems with Applications 25 (2003) 413–423

[55] W. Yuan, D. Guan, Y.K. Lee, S. Lee, S.J. Hur, Improved trust-aware recommender system using small-worldness of trust networks, Knowledge Based Systems 23 (3) (2010) 232–238

[56] J. Zhong, X. Li, Unified collaborative filtering model based on combination of latent features, Expert Systems with Applications 37 (2010) 5666–5672

[57] X. Luo, Y. Xia, Q. Zhu, Incremental collaborative filtering recommender based on regularized matrix factorization, Knowledge-Based Systems 27 (2012) 271–280

[58] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Application of dimensionality reduction in recommender system – a case study, in: ACM WebKDD Workshop, 2000b, pp. 264–272

[59] Y. Koren, R. Bell, CH. Volinsky, Matrix factorization techniques for recommender systems, IEEE Computer 42 (8) (2009) 42–49

[60] X. Luo, Y. Xia, Q. Zhu, Applying the learning rate adaptation to the matrix factorization based collaborative filtering, Knowledge Based Systems 37 (2013) 154–164

[61] Koren, Yehuda; Bell, Robert; Volinsky, Chris (August 2009). "Matrix Factorization Techniques for Recommender Systems". *Computer*. **42** (8): 30–37.

[62] Funk, Simon. "Netflix Update: Try This at Home"

[63] G. Takács, I. Pilászy, B. Németh, D. Tikk, Scalable collaborative filtering approaches for large recommender systems, Journal of Machine Learning Research 10 (2009) 623–656

[64] S. Zhang, W. Wang, J. Ford, F. Makedon, Using singular value decomposition approximation for collaborative filtering, in: IEEE International Conference on E-Commerce Technology, 2005, pp. 1–8.

[65] F. Cacheda, V. Carneiro, D. Fernández, V. Formoso, Comparison of collaborative filtering algorithms: limitations of current techniques and proposals for scalable, high-performance recommender Systems, ACM Transactions on the Web 5 (1) (2011). Article 2.

[66] Jannach, Dietmar; Lerche, Lukas; Gedikli, Fatih; Bonnin, Geoffray (2013). *What Recommenders Recommend – An Analysis of Accuracy, Popularity, and Sales Diversity Effects*. *User Modeling, Adaptation, and Personalization*. Lecture Notes in Computer Science. **7899**. Springer Berlin Heidelberg. pp. 25–37

[67] *Zhu, Yunzhang; Shen, Xiaotong; Ye, Changqing (2016). "Personalized prediction and sparsity pursuit in latent factor models". Journal of the American Statistical Association.* ***111*** *(513): 241–252*

[68] S.K. Shinde, U. Kulkami, Hybrid personalizad recommender system using centering–bunching based clustering algorithm, Expert Systems with Applications 39 (1) (2012) 1381–1387

[69] Z. Yao, Q. Zhang, Item-based clustering collaborative filtering algorithm under high dimensional sparse data, in: International Joint Confeence on Computational Sciences and Optimization, 2009, pp. 787–790

[70] R.L. Zhu, S.J. Gong, Analyzing of collaborative filtering using clustering technology, international colloquium on computing, in: ISECS International Colloquium on Computing, Communication, Control, and Management, 2009, pp. 57–59

[71] T. George, S. Meregu, A scalable collaborative filtering Framework base don co-clustering, in: IEEE International Conference on Data Mining (ICDM), 2005, pp. 625–628

[72] A. Shepitsen, J. Gemmell, B. Mobasher, R. Burke, Personalized recommendation in social tagging systems using hierarchical clustering, in: Proceedings of the 2008 ACM Conference on Recommender Systems, 2008, pp. 259–266.

[73] M.C. Pham, Y. Cao, R. Klamma, M. Jarke, A clustering approach for collaborative filtering recommendation using social network analysis, Journal of Universal Computer Science 17 (4) (2011) 583–604

[74] G. Pitsilis, X. Zhang, W. Wang, Clustering recommenders in collaborative filtering using explicit trust information, Advances in Information and Communication Technology 358 (2011) 82–97

[75] T. Dubois, J. Golbeck, J. Kleint, A. Srinivasan, Improving recommendation accuracy by clustering social networks with trust, in: Proceedings of the 2009 ACM Conference on Recommender Systems, 2009, pp. 1–8

[76] L.Q. Gao, C. Li, Hybrid personalizad recommended model based on genetic algorithm, in: International Conference on Wireless Communication, Networks and Mobile Computing, 2008, pp. 9215–9218

[77] Y. Ho, S. Fong, Z. Yan, A hybrid ga-based collaborative filtering model for online recommenders, in: International Conference on e-Business, 2007, pp. 200–203

[78] M.Y.H. Al-Shamri, K.K. Bharadwaj, Fuzzy-genetic approach to recommender systems based on a novel hybrid user model, Expert Systems with Applications 35 (3) (2008) 1386–1399

[79] M. Lee, Y. Woo, A hybrid recommender system combining collaborative filtering with neural network, Lecture Notes on Computer Sciences 2347 (2002) 531–534

[80] C. Christakou, A. Stafylopatis, A hybrid movie recommender system based on neural networks, in: International Conference on Intelligent Systems Design and Applications, 2005, pp. 500–505

[81] L. Ren, L. HE, J. Gu, W. Xia, F. Wu, A hybrid recommender approach based on Widrow–Hoff learning, in: International Conference on Future Generation Communication and Networking, 2008, pp. 40–45

[82] L .M. Campos, J.M. Fernández-Luna, J.F. Huete, M.A. Rueda-Morales, Combining content-based and collaborative recommendations: a hybrid approach based on Bayesian Networks, International Journal of Approximate Reasoning 51 (7) (2010) 785–799

[83] Adomavicius, G.; Tuzhilin, A. (June 2005)."Toward the Next Generation of Recommender Systems: A Survey of the State-of-the-Art and Possible Extensions". *IEEE Transactions on Knowledge and Data Engineering*.**17**(6): 734–749

[84] *Gomez-Uribe, Carlos A.; Hunt, Neil (28 December 2015). "The Netflix Recommender System". ACM Transactions on Management Information Systems. **6** (4): 1–19*

[85] Robin Burke, Hybrid Web Recommender Systems Archived 2014-09-12 at the Wayback Machine, pp. 377-408

[86] *Li, Jing; Ren, Pengjie; Chen, Zhumin; Ren, Zhaochun; Lian, Tao; Ma, Jun (2017-11-06). "Neural Attentive Session-based Recommendation". Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. CIKM '17. Singapore, Singapore: Association for Computing Machinery: 1419–1428*

[87] Liu, Qiao; Zeng, Yifu; Mokhosi, Refuoe; Zhang, Haibin (2018-07-19). "STAMP: Short-Term Attention/Memory Priority Model for Session-based Recommendation". *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD '18. London, United Kingdom: Association for Computing Machinery: 1831–1839.

[88] Hidasi, Balázs; Karatzoglou, Alexandros; Baltrunas, Linas; Tikk, Domonkos (2016-03-29). "Session-based Recommendations with Recurrent Neural Networks"

[89] Chen, Minmin; Beutel, Alex; Covington, Paul; Jain, Sagar; Belletti, Francois; Chi, Ed (2018). "Top-K Off-Policy Correction for a REINFORCE Recommender System"

[90] Yifei, Ma; Narayanaswamy, Balakrishnan; Haibin, Lin; Hao, Ding (2020). "Temporal-Contextual Recommendation in Real-Time". *KDD '20: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. Association for Computing Machinery: 2291–2299

[91] *Hidasi, Balázs; Karatzoglou, Alexandros (2018-10-17). "Recurrent Neural Networks with Top-k Gains for Session-based Recommendations". Proceedings of the 27th ACM International Conference on Information and Knowledge Management. CIKM '18. Torino, Italy: Association for Computing Machinery: 843–852*

[92] *Lakiotaki, K.; Matsatsinis; Tsoukias, A (March 2011). "Multicriteria User Modeling in Recommender Systems". IEEE Intelligent Systems. **26** (2): 64–76*

[93] Gediminas Adomavicius, Nikos Manouselis, YoungOk Kwon. "Multi-Criteria Recommender Systems"

[94] Bouneffouf, Djallel (2013), *DRARS, A Dynamic Risk-Aware Recommender System*

[95] Yong Ge; Hui Xiong; Alexander Tuzhilin; Keli Xiao; Marco Gruteser; Michael J. Pazzani (2010). *An Energy-Efficient Mobile Recommender System*

[96] Pimenidis, Elias; Polatidis, Nikolaos; Mouratidis, Haralambos (3 August 2018). "Mobile recommender systems: Identifying the major concepts". *Journal of Information Science*. **45** (3): 387–397

[97] Zou, Lixin; Xia, Long; Ding, Zhuoye; Song, Jiaxing; Liu, Weidong; Yin, Dawei (2019). "Reinforcement Learning to Optimize Long-term User Engagement in Recommender Systems"

[98] Wikipedia, 2020. “Cosine Similarity.” https://en.wikipedia.org/wiki/Cosine_similarity

[99] Xingyuan Li.2011 “Collaborative Filtering Recommendation Algorithm Based on Cluster”, International Conference on Computer Science and network Technology(ICCSNT), IEEE, 4: 2682-2685

[100] Yan Shi, Xiao, HongWu Ye, and SongJie Gong. 2008. “A Personalized Recommender Integrating Item-Based And User-Based Collaborative Filtering.” ISBIM '08 International Seminar On Business And Information Management 1 (2008): 264-267

[101] B. Sarwar, G. Karypis, J. Konstan, J. Riedl, Analysis of recommendation algorithms for e-commerce, in: ACM Conference on Electronic Commerce, 2000a, pp. 158–167

[102] A. Gunawardana, G. Shani, A survey of accuracy evaluation metrics of recommender tasks, Journal of Machine Learning Reearch 10 (2009) 2935– 2962

[103] F. Hernández, E. Gaudioso, Evaluation of recommender systems: a new approach, Expert Systems with Applications 35 (2008) 790–804

[104] J. Bobadilla, F. Ortega, A. Hernando, J. Bernal, A collaborative filtering approach to mitigate the new user cold start problem, Knowledge Based Systems 26 (2012) 225–238

[105] P. Antunes, V. Herskovic, S.F. Ochoa, J.A. Pino, Structuring dimensions for collaborative systems evaluation, ACM Computing Surveys 44 (2) (2012). Article 8

[106] K. Goldberg, T. Roeder, D. Gupta, C. Perkins, Eigentaste: a constant time collaborative filtering algorithm, Information Retrieval 4 (2) (2001) 133–151

[107] J.S. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, in: 14th Conference on Uncertainty in Artificial Intelligence, 1998, pp. 43–52

[108] C.N. Ziegler, S.M. Mcnee, J.A. Konstan, G. Lausen, Improving recommendation lists through topic diversification, in: Proceedings of the 14th International Conference on World Wide Web, 2005, pp. 22–32

[109] J. O’donovan, Capturing trust in social web applications, in: J. Golbeck (Ed.), Computing with Social Trust, 2009, pp. 213–257

[110] S. Vargas, P. Castells, Rank and relevance in novelty and diversity metrics for recommender systems, in: Proceedings of the 2011 ACM Conference on Recommender Systems, 2011, pp. 109–116

[111] G. Koutrika, B. Bercovitz, H. Garcia, FlexRecs: expressing and combining flexible recommendations, in: Proceedings of the 35th SIGMOD International Conference on Management of Data, 2009, pp. 745–757

[112] L. Baltrunas, T. Makcinskas, F. Ricci, Group recommendation with rank aggregation and collaborative filtering, in: Proceedings of the 2010 ACM Conference on Recommender Systems, 2010, pp. 119–126

[113] K. Nehring, C. Puppe, A theory of diversity, Econometrica 70 (3) (2002) 1155– 1198

[114] N. Hurley, M. Zhang, Novelty and diversity in top-N recommendationsanalysis and evaluation, ACM Transactions on Internet Technology 10 (4) (2011) 1–29

[115] G. Adomavicius, J. Zhang, On the stability of recommendations algorithms, in: ACM Conference on Recommender Systems, 2010, pp. 47–54

[116] A. Hernando, J. Bobadilla, F. Ortega, J. Tejedor, Incorporating reliability measurements into the predictions of a recommender systems. Information Sciences

[117] Gemulla, E. Nijkamp, P. J. Haas, and Y. Sismanis. Large-scale matrix factorization with distributed stochastic gradient descent. In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining, pages 69–77

[118] Srebro, T. Jaakkola, et al. Weighted low-rank approximations. In ICML, volume 3, pages 720–727, 2003

[119] M. Jaggi and M. Sulovsk. A simple algorithm for nuclear norm regularized problems. In ICML, 2010

[120] Y. Xin and T. Jaakkola. Primal-dual methods for sparse constrained matrix completion. In AISTATS, 2012.

[121] J. Yang and X. Yuan. Linearized augmented lagrangian and alternating direction methods for nuclear norm minimization. Math. Comp. 82, 2013

[122] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In ICML, 2009.

[123] L. Sweeney. k-anonymity: A model for protecting privacy. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 10(05):557–570, 2002.

[124] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. l-diversity: Privacy beyond k-anonymity. ACM Transactions on Knowledge Discovery from Data (TKDD), 1(1):3, 2007

[125] S. Chawla, C. Dwork, F. McSherry, and K. Talwar. On the utility of privacypreserving histograms. In Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence, 2005

[126] M. Barbaro, T. Zeller, and S. Hansell. A face is exposed for aol searcher no. 4417749. New York Times, 9(2008):8For, 2006

[127] M. S. Alvim, M. E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi. Differential privacy: on the trade-off between utility and information leakage. In Formal Aspects of Security and Trust, pages 39–54. Springer, 2012

[128] C. Dwork. Differential privacy: A survey of results. In Theory and Applications of Models of Computation, 2008

[129] F. McSherry and I. Mironov. Differentially private recommender systems: Building privacy into the netflix prize contenders. In SIGKDD, 2009

[130] J.B. Schafer, D. Frankowski, J. Herlocker, S. Sen, Collaborative filltering recommender systems, in: P. Brusilovsky, A. Kobsa, W. Nejdl (Eds.), The Adaptive Web, 2007, pp. 291–324 (Chapter 9)

[131] X.N. Lam, T. Vu, T.D. Le, A.D. Duong, Addressing cold-start problem in recommendation systems, in: Conference On Ubiquitous Information Management And Communication, 2008, pp. 208–211

[132] Y.J. Park, A. Tuzhilin, The long tail of recommender systems and how to leverage it, in: Proceedings of the 2008 ACM Conference on Recommender Systems, 2008, pp. 11–18

[133] S.T. Park, W. Chu, Pairwise preference regression for cold-start recommendation, in: Proceedings of the 2009 ACM Conference on Recommender Systems, 2009, pp. 21–28

[134] A.M. Rashid, G. Karypis, J. Riedl, Learning preferences of new users in recommender systems: an information theoretic approach, in: ACM SIGKDD Explorations Newsletter, vol. 10, issue 2, 2008, pp. 90–100

[135] P.B. Ryan, D. Bridge, Collaborative recommending using formal concept analysis, Knowledge Based Systems 19 (5) (2006) 309–315

[136] C.W. Leung, S.C. Chan, F.L. Chung, An empirical study of a cross-level association rule mining approach to cold-start recommendations, Knowledge Based Systems 21 (7) (2008) 515–529

[137] H.N. Kim, A.T. Ji, I. Ha, G.S. Jo, Collaborative filtering based on collaborative tagging for enhancing the quality of recommendations, Electronic Commerce Research and Applications 9 (1) (2010) 73–83

[138] L.T. Weng, Y. Xu, Y. Li, R. Nayak, Exploiting item taxonomy for solving coldstart problem in recommendation making, in: Proceedings of the 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI2008), 2008, pp. 113–120

[139] S. Loh, F. Lorenzi, R. Granada, D. Lichtnow, L.K. Wives, J.P. Oliveira, Identifying similar users by their scientific publications to reduce cold start in recommender systems, in: Proceedings of the 5th International Conference on Web Information Systems and Technologies (WEBIST2009), 2009, pp. 593–600

[140] L. Martinez, L.G. Perez, M.J. Barranco, Incomplete preference relations to smooth out the cold-start in collaborative recommender systems, in: Proceedings of the 28th North American Fuzzy Information Processing Society Annual Conference (NAFIPS2009), 2009a, pp. 1–6

[141] T. Chen, L. He, Collaborative filtering based on demographic attribute vector, in: Proceedings of the International Conference on Future Computer and Communication, 2009, pp. 225–229

[142] M. Saranya, T. Atsuhiro, Hybrid recommender systems using latent features, in: Proceedings of the International Conference on Advanced Information Networking and Applications Workshops, 2009, pp. 661–666

[143] S.T. Park, D.M. Pennock, O. Madani, N. Good, D. Coste, Naïve filterbots for robust cold-start recommendations, in: Proceedings of Knowledge Discovery and Data Mining (KDD2006), 2006, pp. 699–705

[144] https://numpy.org/doc/stable/contents.html

[145] https://matplotlib.org/

[146] https://pandas.pydata.org/

[147] https://surprise.readthedocs.io/en/stable/knn_inspired.html#surprise.prediction_algorithms.knns.KNNWithZScore

[148] https://surprise.readthedocs.io/en/stable/matrix_factorization.html#surprise.prediction_algorithms.matrix_factorization.NMF

[149] https://surprise.readthedocs.io/en/stable/similarities.html/

[150] Herlocker, J., Konstan, J., Borchers, A., Riedl, J.. An Algorithmic Framework for Performing Collaborative Filtering. Proceedings of the 1999 Conference on Research and Development in Information Retrieval. Aug. 1999.

[151] https://grouplens.org/datasets/movielens/